

Michel Abdalla
Ricardo Dahab (Eds.)

LNCS 10769

Public-Key Cryptography – PKC 2018

21st IACR International Conference
on Practice and Theory of Public-Key Cryptography
Rio de Janeiro, Brazil, March 25–29, 2018
Proceedings, Part I

1
Part I



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

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

Michel Abdalla · Ricardo Dahab (Eds.)

Public-Key Cryptography – PKC 2018

21st IACR International Conference
on Practice and Theory of Public-Key Cryptography
Rio de Janeiro, Brazil, March 25–29, 2018
Proceedings, Part I

Editors

Michel Abdalla 
CNRS and École Normale Supérieure
Paris
France

Ricardo Dahab
University of Campinas
Campinas, SP
Brazil

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-76577-8 ISBN 978-3-319-76578-5 (eBook)
<https://doi.org/10.1007/978-3-319-76578-5>

Library of Congress Control Number: 2018934351

LNCS Sublibrary: SL4 – Security and Cryptology

© International Association for Cryptologic Research 2018

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, express 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.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 21st IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2018) was held March 25–29, 2018, in Rio de Janeiro, Brazil. The conference is sponsored by the International Association for Cryptologic Research (IACR) and focuses on all technical aspects of public-key cryptography.

These proceedings consist of two volumes including 49 papers that were selected by the Program Committee from 186 submissions. Each submission was assigned to at least three reviewers while submissions co-authored by Program Committee members received at least four reviews. Following the initial reviewing phase, the submissions were discussed over a period of five weeks. During this discussion phase, the Program Committee used quite intensively a recent feature of the review system, which allows Program Committee members to anonymously ask questions to the authors.

The reviewing and selection process was a challenging task and I am deeply grateful to the Program Committee members and external reviewers for their hard and thorough work. Many thanks also to Shai Halevi for his assistance with the Web submission and review software and for his constant availability.

The conference program also included invited talks by Elette Boyle (IDC Herzliya, Israel) and Hugo Krawczyk (IBM Research, USA). I would like to thank both of them as well as all the other speakers for their contributions to the program.

Finally, I would like to thank Ricardo Dahab, the general chair, for organizing a great conference and all the conference attendees for making this a truly intellectually stimulating event through their active participation.

March 2018

Michel Abdalla

PKC 2018

21st International Conference on Practice and Theory of Public-Key Cryptography

Rio de Janeiro, Brazil
March 25–29, 2018

Sponsored by
The International Association of Cryptologic Research

General Chair

Ricardo Dahab University of Campinas, Brazil

Program Chair

Michel Abdalla CNRS and École Normale Supérieure, France

Program Committee

Shweta Agrawal	Indian Institute of Technology, Madras, India
Prabhanjan Ananth	UCLA and MIT, USA
Diego Aranha	University of Campinas, Brazil
Mihir Bellare	University of California, San Diego, USA
Chris Brzuska	Hamburg University of Technology, Germany
Dario Catalano	Università di Catania, Italy
Jie Chen	East China Normal University, China
Yilei Chen	Boston University, USA
Céline Chevalier	Université Panthéon-Assas Paris 2, France
Kai-Min Chung	Academia Sinica, Taiwan
Dana Dachman-Soled	University of Maryland, USA
Bernardo David	Tokyo Institute of Technology, Japan
Léo Ducas	CWI Amsterdam, The Netherlands
Nico Döttling	FAU Erlangen-Nürnberg, Germany
Pierre-Alain Fouque	Rennes 1 University, France
Sergey Gorbunov	University of Waterloo, Canada
Aurore Guillevic	Inria, France
Carmit Hazay	Bar-Ilan University, Israel
Julia Hesse	Karlsruhe Institute of Technology, Germany
Zahra Jafargholi	Aarhus University, Denmark
Tibor Jager	Paderborn University, Germany
Bhavana Kanukurthi	Indian Institute of Science, India
Markulf Kohlweiss	Microsoft Research and University of Edinburgh, UK

Adeline Langlois	CNRS and Rennes 1 University, France
Payman Mohassel	Visa Research, USA
Ryo Nishimaki	NTT Secure Platform Labs, Japan
Alain Passelègue	UCLA, USA
Arpita Patra	Indian Institute of Science, India
Antigoni Polychroniadou	Cornell University, USA
Carla Ràfols Salvador	Universitat Pompeu Fabra, Spain
Alessandra Scafuro	North Carolina State University, USA
Christian Schaffner	University of Amsterdam & QuSoft, The Netherlands
Gil Segev	Hebrew University, Israel
Jae Hong Seo	Myongji University, South Korea
Qiang Tang	New Jersey Institute of Technology, USA
Mehdi Tibouchi	NTT Secure Platform Laboratories, Japan
Bogdan Warinschi	University of Bristol, UK
Mor Weiss	Northeastern University, USA

Additional Reviewers

Masayuki Abe	Binyi Chen
Shashank Agrawal	Long Chen
Erdem Alkim	Rongmao Chen
Nuttapong Attrapadung	Yu Chen
Saikrishna Badrinarayanan	Nai-Hui Chia
Shi Bai	Arka Rai Choudhuri
Christian Bardetscher	Ashish Choudhury
Hridam Basu	Peter Chvojka
Balthazar Bauer	Michele Ciampi
Carsten Baum	Ran Cohen
Pascal Bemmman	Sandro Coretti
Fabrice Benhamouda	Craig Costello
David Bernhard	Geoffroy Couteau
Pauline Bert	Jan Czajkowski
Olivier Blazy	Anders Dalskov
Guillaume Bonnoron	Luca De Feo
Niek Bouman	Jean Paul Degabriele
Florian Bourse	David Derler
Jacqueline Brendel	Apoorvaa Deshpande
Ran Canetti	Mario Di Raimondo
Guilhem Castagnos	Luis J. Dominguez Perez
Suvradip Chakraborty	Rafael Dowsley
Nishanth Chandran	Yfke Dulek
Sanjit Chatterjee	Lisa Eckey

Andrew Ellis
Lucas Enloe
Naomi Ephraim
Thomas Espitau
Leo Fan
Xiong Fan
Antonio Faonio
Prastudy Fauzi
Armando Faz-Hernández
Rex Fernando
Houda Ferradi
Claus Fieker
Dario Fiore
Marc Fischlin
Benjamin Fuller
Philippe Gaborit
Nicolas Gama
Chaya Ganesh
Romain Gay
Kai Gellert
Ran Gelles
Nicholas Genise
Paul Germouty
Essam Ghadafi
Satrajit Ghosh
Irene Giacomelli
Huijing Gong
Junqing Gong
Alonso González
Conrado Porto Lopes Gouvêa
Rishab Goyal
Paul Grubbs
Siyao Guo
Divya Gupta
Kyoohyung Han
Javier Herranz
Justin Holmgren
Kristina Hostakova
Zhengan Huang
Andreas Huelsing
Robin Hui
Shih-Han Hung
Aaron Hutchinson
Ilia Iliashenko
Sorina Ionica
Malika Izabachène
Michael Jacobson
Joseph Jaeger
Aayush Jain
Christian Janson
Stacey Jeffery
Saqib Kakvi
Shuichi Katsumata
Natasha Kharchenko
Sam Kim
Taechan Kim
Elena Kirshanova
Fuyuki Kitagawa
Susumu Kiyoshima
Konrad Kohbrok
Lisa Kohl
Ilan Komargodski
Stephan Krenn
Ashutosh Kumar
Rafael Kurek
Eyal Kushilevitz
Russell Lai
Kim Laine
Mario Larangeira
Changmin Lee
Hyung Tae Lee
Kwansu Lee
Moon Sung Lee
Nikos Leonardos
Iraklis Leontiadis
Qinyi Li
Benoît Libert
Weikai Lin
Feng-Hao Liu
Shengli Liu
Tianren Liu
Alex Lombardi
Vadim Lyubashevsky
Fermi Ma

Gilles Macario-Rat
Varun Madathil
Bernardo Magri
Monosij Maitra
Christian Majenz
Hemanta K. Maji
Giulio Malavolta
Mary Maller
Mark Manulis
Giorgia Azzurra Marson
Takahiro Matsuda
Sogol Mazaheri
Thierry Mefenza
Peihan Miao
Ian Miers
Ameer Mohammed
Paz Morillo
Fabrice Mouhartem
Pratyay Mukherjee
Pierrick Méaux
Gregory Neven
Khoa Nguyen
David Niehues
Luca Nizzardo
Sai Lakshmi Bhavana Obbattu
Cristina Onete
Michele Orrù
Emmanuela Orsini
Jheyne N. Ortiz
Daniel Escudero Ospina
Maris Ozols
Jiaxin Pan
Tapas Pandit
Dimitris Papadopoulos
Filip Pawlega
Thomas Peters
Doung Hieu Phan
Cecile Pierrot
Zaira Pindado
Oxana Poburinnaya
Chen Qian
Elizabeth Quaglia
Liz Quaglia
Ananth Raghunathan
Srinivasan Raghuraman
Somindu C. Ramanna

Divya Ravi
Guénaél Renault
Peter Rindal
Miruna Rosca
Lior Rotem
Kai Samelin
Pratik Sarkar
Sajin Sasy
John Schanck
Peter Scholl
Dominique Schröder
Adam Sealton
Sruthi Sekar
Nicolas Sendrier
Barak Shani
Abhishek Shetty
Javier Silva
Mark Simkin
Luisa Siniscalchi
Daniel Slamanig
Ben Smith
Fang Song
Eduardo Soria-Vazquez
Akshayaram Srinivasan
Ron Steinfeld
Mario Strefler
Christoph Striecks
Atsushi Takayasu
Benjamin Hong Meng Tan
Emmanuel Thomé
Sri Aravinda Thyagarajan
Ni Trieu
Rotem Tsabary
Jorge L. Villar
Dhinakaran Vinayagamurthy
Satyanarayana Vusirikala
Riad S. Wahby
Kun-Peng Wang
Mingyuan Wang
Xiao Wang
Yuyu Wang
Yohei Watanabe
Weiqiang Wen
Benjamin Wesolowski
David Wu
Keita Xagawa

Fan Xiong
Sophia Yakoubov
Shota Yamada
Takashi Yamakawa
Avishay Yanai
Rupeng Yang
Arkady Yerukhimovich
Eylon Yogev
Zuoxia Yu

Aaram Yun
Mohammad Zaheri
Mark Zhandry
Daode Zhang
Jiang Zhang
Kai Zhang
Ren Zhang
Linfeng Zhou

Sponsoring Institutions

Accenture Digital (<https://www.accenture.com/br-pt/digital-index>)

ERC CryptoCloud (<http://www.di.ens.fr/users/pointche/cryptocloud.php>)

Scyphir Unipessoal, LDA (<http://scyphir.pt>)

Contents – Part I

Key-Dependent-Message and Selective-Opening Security

New Constructions of Identity-Based and Key-Dependent Message Secure Encryption Schemes	3
<i>Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny</i>	
Key Dependent Message Security and Receiver Selective Opening Security for Identity-Based Encryption	32
<i>Fuyuki Kitagawa and Keisuke Tanaka</i>	
Tightly SIM-SO-CCA Secure Public Key Encryption from Standard Assumptions	62
<i>Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu</i>	

Searchable and Fully Homomorphic Encryption

Multi-Key Searchable Encryption, Revisited	95
<i>Ariel Hamlin, Abhi Shelat, Mor Weiss, and Daniel Wichs</i>	
Fully Homomorphic Encryption from the Finite Field Isomorphism Problem	125
<i>Yarkin Doröz, Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, Berk Sunar, William Whyte, and Zhenfei Zhang</i>	

Public-Key Encryption

Hybrid Encryption in a Multi-user Setting, Revisited	159
<i>Federico Giacon, Eike Kiltz, and Bertram Poettering</i>	
KEM Combiners	190
<i>Federico Giacon, Felix Heuer, and Bertram Poettering</i>	
Revisiting Proxy Re-encryption: Forward Secrecy, Improved Security, and Applications	219
<i>David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks</i>	

Encryption with Bad Randomness

Hedged Nonce-Based Public-Key Encryption: Adaptive Security
Under Randomness Failures 253
*Zhengan Huang, Junzuo Lai, Wenbin Chen, Man Ho Au, Zhen Peng,
and Jin Li*

Related Randomness Security for Public Key Encryption, Revisited 280
Takahiro Matsuda and Jacob C. N. Schuldt

Subversion Resistance

Subversion-Zero-Knowledge SNARKs. 315
Georg Fuchsbauer

Public-Key Encryption Resistant to Parameter Subversion
and Its Realization from Efficiently-Embeddable Groups 348
Benedikt Auerbach, Mihir Bellare, and Eike Kiltz

Cryptanalysis

A Practical Cryptanalysis of WalnutDSA™ 381
*Daniel Hart, DoHoon Kim, Giacomo Micheli, Guillermo Pascual-Perez,
Christophe Petit, and Yuxuan Quek*

Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving 407
Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven

Fast Lattice Basis Reduction Suitable for Massive Parallelization
and Its Application to the Shortest Vector Problem 437
Tadanori Teruya, Kenji Kashiwabara, and Goichiro Hanaoka

Composable Security

Reusing Tamper-Proof Hardware in UC-Secure Protocols 463
Jeremias Mechler, Jörn Müller-Quade, and Tobias Nilges

On Composable Security for Digital Signatures. 494
Christian Badertscher, Ueli Maurer, and Björn Tackmann

Oblivious Transfer

Equational Security Proofs of Oblivious Transfer Protocols 527
Baiyu Li and Daniele Micciancio

Extending Oblivious Transfer with Low Communication
 via Key-Homomorphic PRFs 554
Peter Scholl

Multiparty Computation

Committed MPC: Maliciously Secure Multiparty Computation
 from Homomorphic Commitments 587
Tore K. Frederiksen, Benny Pinkas, and Avishay Yanai

Fast Garbling of Circuits over 3-Valued Logic 620
Yehuda Lindell and Avishay Yanai

Efficient Covert Two-Party Computation 644
Stanislaw Jarecki

Towards Characterizing Securely Computable Two-Party
 Randomized Functions. 675
Deepesh Data and Manoj Prabhakaran

On the Message Complexity of Secure Multiparty Computation 698
Yuval Ishai, Manika Mittal, and Rafail Ostrovsky

Author Index 713

Contents – Part II

Signatures

SOFIA: \mathcal{MQ} -Based Signatures in the QROM	3
<i>Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe</i>	
A Unified Framework for Trapdoor-Permutation-Based Sequential Aggregate Signatures	34
<i>Craig Gentry, Adam O’Neill, and Leonid Reyzin</i>	
Constant-Size Group Signatures from Lattices	58
<i>San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu</i>	
Attribute-Based Signatures for Unbounded Circuits in the ROM and Efficient Instantiations from Lattices	89
<i>Ali El Kaafarani and Shuichi Katsumata</i>	

Structure-Preserving Signatures

Improved (Almost) Tightly-Secure Structure-Preserving Signatures	123
<i>Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy</i>	
Weakly Secure Equivalence-Class Signatures from Standard Assumptions . . .	153
<i>Georg Fuchsbauer and Romain Gay</i>	

Functional Encryption

Simple and Generic Constructions of Succinct Functional Encryption	187
<i>Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka</i>	
Making Public Key Functional Encryption Function Private, Distributively	218
<i>Xiong Fan and Qiang Tang</i>	
Full-Hiding (Unbounded) Multi-input Inner Product Functional Encryption from the k -Linear Assumption	245
<i>Pratish Datta, Tatsuki Okamoto, and Junichi Tomida</i>	

Foundations

Local Non-malleable Codes in the Bounded Retrieval Model	281
<i>Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi</i>	

Non-malleability vs. CCA-Security: The Case of Commitments 312
*Brandon Broadnax, Valerie Fetzer, Jörn Müller-Quade,
 and Andy Rupp*

Obfuscation-Based Cryptographic Constructions

Interactively Secure Groups from Obfuscation 341
Thomas Arikola and Dennis Hofheinz

Graded Encoding Schemes from Obfuscation 371
*Pooya Farshim, Julia Hesse, Dennis Hofheinz,
 and Enrique Larraia*

Protocols

Hashing Solutions Instead of Generating Problems: On the Interactive
 Certification of RSA Moduli 403
Benedikt Auerbach and Bertram Poettering

Two-Factor Authentication with End-to-End Password Security 431
*Stanislaw Jarecki, Hugo Krawczyk, Maliheh Shirvanian,
 and Nitesh Saxena*

Blockchain

Bootstrapping the Blockchain, with Applications to Consensus
 and Fast PKI Setup 465
*Juan A. Garay, Aggelos Kiayias, Nikos Leonardos,
 and Giorgos Panagiotakos*

Zero-Knowledge

Efficient Adaptively Secure Zero-Knowledge from Garbled Circuits 499
Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar

Compact Zero-Knowledge Proofs of Small Hamming Weight. 530
*Ivan Damgård, Ji Luo, Sabine Oechsner, Peter Scholl,
 and Mark Simkin*

Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials . . . 561
Jonathan Bootle and Jens Groth

On the Security of Classic Protocols for Unique Witness Relations 589
Yi Deng, Xuyang Song, Jingyue Yu, and Yu Chen

Lattices

<p>New (and Old) Proof Systems for Lattice Problems.</p> <p style="padding-left: 2em;"><i>Navid Alamati, Chris Peikert, and Noah Stephens-Davidowitz</i></p>	<p>619</p>
<p>Hash Proof Systems over Lattices Revisited</p> <p style="padding-left: 2em;"><i>Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach</i></p>	<p>644</p>
<p>Privately Constraining and Programming PRFs, the LWE Way.</p> <p style="padding-left: 2em;"><i>Chris Peikert and Sina Shiehian</i></p>	<p>675</p>
<p>Learning with Errors and Extrapolated Dihedral Cosets</p> <p style="padding-left: 2em;"><i>Zvika Brakerski, Elena Kirshanova, Damien Stehlé, and Weiqiang Wen</i></p>	<p>702</p>
<p>Rounded Gaussians: Fast and Secure Constant-Time Sampling for Lattice-Based Crypto</p> <p style="padding-left: 2em;"><i>Andreas Hülsing, Tanja Lange, and Kit Smeets</i></p>	<p>728</p>
<p>Author Index</p>	<p>759</p>

Key-Dependent-Message and Selective-Opening Security



New Constructions of Identity-Based and Key-Dependent Message Secure Encryption Schemes

Nico Döttling¹(✉), Sanjam Garg², Mohammad Hajiabadi²,
and Daniel Masny²(✉)

¹ Friedrich-Alexander-University Erlangen-Nürnberg, Erlangen, Germany
nico.doettling@fau.de

² University of California, Berkeley, USA
{sanjam,mdhajiabadi,daniel.masny}@berkeley.edu

Abstract. Recently, Döttling and Garg (CRYPTO 2017) showed how to build identity-based encryption (IBE) from a novel primitive termed *Chameleon Encryption*, which can in turn be realized from simple number theoretic hardness assumptions such as the computational Diffie-Hellman assumption (in groups without pairings) or the factoring assumption. In a follow-up work (TCC 2017), the same authors showed that IBE can also be constructed from a slightly weaker primitive called *One-Time Signatures with Encryption* (OTSE).

In this work, we show that OTSE can be instantiated from hard learning problems such as the Learning With Errors (LWE) and the Learning Parity with Noise (LPN) problems. This immediately yields the first IBE construction from the LPN problem and a construction based on a weaker LWE assumption compared to previous works.

Finally, we show that the notion of one-time signatures with encryption is also useful for the construction of key-dependent-message (KDM) secure public-key encryption. In particular, our results imply that a KDM-secure public key encryption can be constructed from any KDM-secure secret-key encryption scheme and any public-key encryption scheme.

1 Introduction

Identity-based encryption (IBE) is a form of public key encryption that allows a sender to encrypt messages to a user without knowing a user-specific public key, but only the user's name or identity and some global and succinct public

Research supported in part from 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

parameters. The public parameters are issued by a key authority which also provides identity-specific secret keys to the users.

The notion of IBE was originally proposed by Shamir [Sha84], and in two seminal results Boneh and Franklin [BF01] and Cocks [Coc01] provided the first candidate constructions of IBE in the random oracle model from groups with pairings and the quadratic residue problem respectively. Later works on IBE provided security proofs without random oracles [CHK04, BB04, Wat05, Wat09, LW10, BGH07] and realized IBE from hard lattice problems [GPV08, CHKP12, ABB10].

In a recent result, Döttling and Garg [DG17b] showed how to construct IBE from (presumably) qualitatively simpler assumptions, namely the computational Diffie-Hellman assumption in groups without pairings or the factoring assumption. In a follow-up work, the same authors [DG17a] provided a generalization of the framework proposed in [DG17b]. In particular, the authors show that identity-based encryption is equivalent to the seemingly simpler notion of *One-Time Signatures with Encryption* (OTSE) using a refined version of the tree-based IBE construction of [DG17b].

An OTSE-scheme is a one-time signature scheme with an additional encryption and decryption functionality. Informally, the encryption functionality allows anyone to encrypt a plaintext m to a tuple consisting of a public parameter \mathbf{pp} , a verification key \mathbf{vk} , an index i and a bit b , to obtain a ciphertext c . The plaintext m can be deciphered from c by using a pair of message-signature (x, σ) that is valid relative to \mathbf{vk} and which satisfies $x_i = b$. Security of the OTSE asserts that an adversary knowing a pair of message-signature (x, σ) and the underlying public parameter \mathbf{pp} and verification key \mathbf{vk} cannot distinguish between encryptions of two plaintexts encrypted to $(i, 1 - x_i)$ under $(\mathbf{pp}, \mathbf{vk})$, for any index i of the adversary's choice. (Note that this security property implies the one-time unforgeability of the signature.) The OTSE also needs to be compact, meaning the size of the verification key grows only with the security parameter, and does not depend on the size of messages allowed to be signed.

1.1 PKE and IBE from Learning with Errors

We will briefly review constructions of public-key encryption and identity-based encryption from the Learning with Errors (LWE) problem.

The hardness of LWE is determined by its dimension n , modulus q , noise magnitude parameter α and the amount of samples m . Regev [Reg05] showed that among the latter three parameters, in particular the noise magnitude parameter α is of major importance since it directly impacts the approximation factor of the underlying lattice problem.

Theorem 1 [Reg05]. *Let $\epsilon = \epsilon(n)$ be some negligible function of n . Also, let $\alpha = \alpha(n) \in (0, 1)$ be some real and let $p = p(n)$ be some integer such that $\alpha p > 2\sqrt{n}$. Assume there exists an efficient (possibly quantum) algorithm that solves $LWE_{p, \alpha}$. Then there exists an efficient quantum algorithm for solving the following worst-case lattice problems:*

1. Find a set of n linearly independent lattice vectors of length at most $\tilde{O}(\lambda_n(L) \cdot n/\alpha)$.
2. Approximate $\lambda_1(L)$ within $\tilde{O}(n/\alpha)$.

Here, λ_k is the minimal length of k linearly independent vectors in lattice L . To find such vectors within a constant or slightly sublinear approximation is known to be NP-hard under randomized reductions [ABSS93, Ajt98, Mic98, Kho04, HR07], while for an exponential approximation factor, they can be found in polynomial time using the LLL algorithm [LLL82]. Regev [Reg05] introduced the first PKE based on LWE for a choice of $\alpha = \tilde{O}(1/\sqrt{n})$, more precisely $\alpha = 1/(\sqrt{n} \log^2 n)$. The first lattice based IBEs, by Gentry et al. [GPV08], Cash et al. [CHKP10] and by Agrawal et al. [ABB10] require $\alpha = \tilde{O}(1/n)$, $\alpha = \tilde{O}(1/(\sqrt{kn}))$, where k is the output length of a hash function, and $\alpha = \tilde{O}(1/n^2)$.

The reason for this gap between PKE and IBE is that all the known IBE constructions use an additional trapdoor in order to sample short vectors as secret keys. This sampling procedure increases the norm of sampled vectors, such that the initial noise of a ciphertext must be decreased to maintain the correctness of the schemes. By losing a factor \sqrt{n} in the sampling procedure [MR04, GPV08, MP12, LW15], α needs to be chosen by a factor \sqrt{n} smaller. Therefore, this methodology unavoidably loses at least an additional \sqrt{n} factor. This explains why these techniques cause a gap compared to Regev’s PKE where α is at least a factor \sqrt{n} larger, which decreases the approximation factor by at least a factor of \sqrt{n} . This results in a stronger assumption with respect to the underlying short vector problem.

1.2 Our Results

As the main contribution of this work, we remove the requirement of the collision-tractability property of the hash function in the construction of [DG17a]. Specifically, we replace the notion of Chameleon Encryption with the simpler notion of *Hash Encryption*, for which no collision tractability property is required. The notion of Hash Encryption naturally arises from the notion of laconic Oblivious Transfer [CDG+17]. We provide simple and efficient constructions from the Learning With Errors (LWE) [Reg05] and (exponentially hard) Learning Parity with Noise (LPN) problem [YZ16].

Our overall construction of IBE from hash encryption proceeds as follows. We first show that we can use any CPA PKE to build a *non-compact* version of One-Time Signatures with Encryption (OTSE), in which, informally, the size of the verification key of the OTSE is bigger than the size of the messages allowed to be signed. We then show how to use hash encryption to boost non-compact OTSE into compact OTSE, under which arbitrarily large messages could be signed using a short public parameter and a short verification key, while preserving the associated encryption-decryption functionalities. Our transformation makes a non-black-box use of the non-compact OTSE primitive.

Using a recent result by Döttling and Garg [DG17a], we transform our compact OTSE to an IBE. Hence, we obtain the first constructions of IBE from the LWE assumption used by Regev’s PKE and the first construction from an LPN problem.

Further, we show how to use non-compact OTSE to transform key-dependent-message (KDM) secure *private* key encryption to KDM-secure *public* key encryption. Informally, a private-key encryption scheme is \mathcal{F} -KDM secure, for a function class \mathcal{F} , if the scheme remains semantically secure even if the adversary is allowed to obtain encryptions of $f(k)$, for $f \in \mathcal{F}$, under the secret key k itself. This notion is analogously defined for PKE. A large body of work, e.g., [BHHO08, ACPS09, BG10, BHHI10, App14, Döt15], shows how to build KDM-secure schemes from various specific assumptions. Briefly, in order to construct KDM-secure schemes for a large class of functions, they first show how to build KDM-secure schemes for a basic class of functions [BHHO08, BG10, ACPS09] (e.g., *projections*, *affine*) and then use KDM amplification procedures [BHHI10, App14] to obtain KDM security against richer functions families. We show that for any function family \mathcal{F} , an \mathcal{F} -KDM secure PKE can be obtained from a non-compact OTSE (and hence a CPA PKE) together with a \mathcal{G} -KDM secure private-key encryption scheme, where \mathcal{G} is a class of functions related to \mathcal{F} . (See Sect. 6 for a formal statement.) Using the result of [App14] we obtain that \mathcal{F} -KDM-secure PKE, for any \mathcal{F} , can be based on projection-secure private-key encryption and CPA PKE. We mention that prior to our work it was not known whether projection-secure PKE (which is sufficient for KDM PKE) could be constructed (in a black-box or a non-black-box way) from the combination of CPA PKE and projection-secure private-key encryption.

An overview of the contributions of this work is given in Fig. 1.

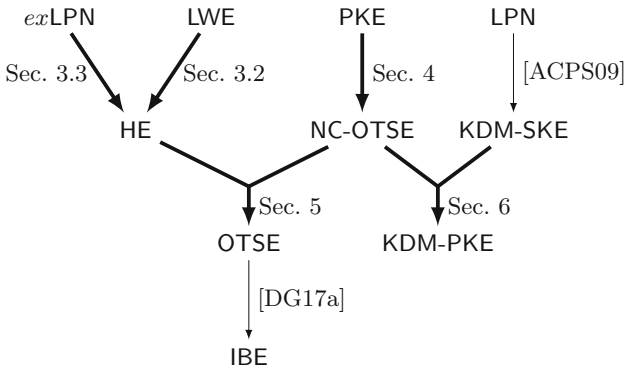


Fig. 1. Overview of the results in this work, bold arrows are contributions of this work.

1.3 Technical Outline

We will start by providing an outline of our construction of hash encryption from LWE. The LPN-based construction is similar in spirit, yet needs to account for additional subtleties that arise in the modulus 2 case. We will then sketch our construction of IBE from hash encryption.

Hash Encryption from LWE. The hashing key k of our hash function is given by a randomly chosen matrix $A \leftarrow \mathbb{Z}_p^{m \times \kappa}$. To hash a message, we encoded it as a vector $x \in \{0, 1\}^m \subseteq \mathbb{Z}^m$ and compute the hash value $h \leftarrow x^\top \cdot A$. It can be shown that under the short integer solution (SIS) problem [Reg05] this function is collision resistant.

We will now specify the encryption and decryption procedures. Our encryption scheme is a variant of the dual-Regev [GPV08] encryption scheme. For a matrix A , let A_{-i} denote the matrix obtained by removing the i -th row of A , and let a_i be the i -th row of A . Likewise, for a vector x let x_{-i} denote the vector obtained by dropping the i -th component of x . Given the hashing key $k = A$, a hash-value h , an index i and a bit b , we encrypt a message $m \in \{0, 1\}$ to a ciphertext $c = (c_1, c_2)$ via

$$\begin{aligned} c_1 &\leftarrow A_{-i} \cdot s + e_{-i} \\ c_2 &\leftarrow (h - b \cdot a_i) s + e_i + \lfloor p/2 \rfloor \cdot m, \end{aligned}$$

where $s \leftarrow \mathbb{Z}_p^\kappa$ is chosen uniformly at random and $e \in \mathbb{Z}_p^m$ is chosen from an appropriate discrete gaussian distribution.

To decrypt a ciphertext c using a preimage x , compute

$$\mu \leftarrow c_2 - x_{-i}^\top c_1,$$

output 0 if μ is closer to 0 and 1 if μ is closer to $p/2$. Correctness of this scheme follows similarly as in the dual Regev scheme [GPV08]. To argue security, we will show that a successful adversary against this scheme can be used to break the decisional extended LWE problem [AP12], which is known to be equivalent to standard LWE.

Compact OTSE from Non-compact OTSE and Hash Encryption. To obtain a compact OTSE scheme, we hash the verification keys of the non-compact OTSE-scheme using the hash function of the hash encryption primitive. While this resolves the compactness issue, it destroys the encryption-decryption functionalities of the non-compact OTSE. We overcome this problem through a non-blackbox usage of the encryption function of the base non-compact OTSE-scheme.

KDM Security. We sketch the construction of a KDM^{CPA} -secure PKE from a non-compact OTSE NC and a KDM^{CPA} -secure secret-key encryption scheme $\text{SKE} = (\text{Enc}, \text{Dec})$. We also need a garbling scheme (Garble, Eval), which can be built from SKE.

The public key $\mathbf{pk} = (\mathbf{pp}, \mathbf{vk})$ of the PKE is a public parameter \mathbf{pp} and a verification key \mathbf{vk} of NC and the secret key is $\mathbf{sk} = (k, \sigma)$, where k is a key of the secret-key scheme and σ is a valid signature of k w.r.t. \mathbf{vk} .

To encrypt m under $\mathbf{pk} = (\mathbf{pp}, \mathbf{vk})$ we first form a circuit C which on input k' returns $\text{Enc}(k', m)$. We then garble C to obtain a garbled circuit \tilde{C} and input labels $(X_{\iota,0}, X_{\iota,1})$ for every input index ι . For all ι and bit b , we OTSE-encrypt $X_{\iota,b}$ relative to the index ι and bit b (using \mathbf{pp} and \mathbf{vk}) to get $\mathbf{ct}_{\iota,b}$. The resulting ciphertext is then $\mathbf{ct} = (\tilde{C}, \{\mathbf{ct}_{\iota,b}\}_{\iota,b})$.

For decryption, using (k, σ) we can OTSE-decrypt the proper $\mathbf{ct}_{\iota,b}$'s to obtain a matching garbled input \tilde{k} for k . Then evaluating \tilde{C} on \tilde{k} we obtain $\mathbf{ct}' = \text{Enc}(k, m)$. We can then decrypt \mathbf{ct}' using k to recover m .

Using a series of hybrids we reduce the KDM security of the PKE to the stated security properties of the base primitives.

1.4 Concurrent Works

In a concurrent and independent work, Brakerski et al. [BLSV17] provided a construction of an IBE scheme from LPN with a very low noise rate of $\Omega(\log(\kappa)^2/\kappa)$, using techniques similar to the construction of OTSE from sub-exponentially hard LPN in this work. Also in a concurrent and independent work, Kitagawa and Tanaka [KT17] provided a construction of KDM-secure public key encryption from KDM-secure secret key encryption and IND-CPA secure public key encryption using techniques similar to ours.

2 Preliminaries

We use $\{0, 1\}_k^m$ to denote the set of binary vectors of length m with hamming weight k and $[m]$ to denote the set $\{1, \dots, m\}$. We use A_{-i} to denote matrix A where the i th row is removed. The same holds for a row vector x_{-i} , which denotes vector x where the i th entry is removed.

Lemma 1. *For $m \in \mathbb{N}$ and $1 \leq k \leq m$, the cardinality of set $\{0, 1\}_k^m$ is lower bounded by $\left(\frac{m}{k}\right)^k$ and upper bounded by $\left(\frac{em}{k}\right)^k$.*

Definition 1 (Bias). *Let $x \in \mathbb{F}_2$ be a random variable. Then the bias of x is defined by*

$$\text{bias}(x) = \Pr[x = 0] - \Pr[x = 1].$$

Remark 1. The bias of x is simply the second Fourier coefficient of the probability distribution of x , the first Fourier coefficient being 1 for all distributions. Thus, as $\Pr[x = 1] = 1 - \Pr[x = 0]$ it holds that $\Pr[x = 0] = \frac{1}{2} + \frac{1}{2}\text{bias}(x)$.

In the following, we summarize several useful properties of the bias of random variables.

- If $x \leftarrow B_\rho$, then $\text{bias}(x) = 1 - 2\rho$.
- Let $x_1, x_2 \in \mathbb{F}_2$ be independent random variables. Then it holds that $\text{bias}(x_1 + x_2) = \text{bias}(x_1) \cdot \text{bias}(x_2)$.
- Assume that the distribution of x is the convex combination of two distributions via $p_x = \alpha p_{x_1} + (1 - \alpha)p_{x_2}$. Then $\text{bias}(x) = \alpha \text{bias}(x_1) + (1 - \alpha)\text{bias}(x_2)$.

Proof. Convolution theorem

Lemma 2. *Let $v \in \mathbb{F}_2^n$ be a vector of weight t and $e \in \mathbb{F}_2^n$ a distribution for which each component is iid distributed with bias ϵ . Then it holds that $\Pr[\langle v, e \rangle = 0] = \frac{1}{2} + \frac{1}{2}\epsilon^t$.*

Proof. As v has weight t , it holds that

$$\text{bias}(\langle v, e \rangle) = \text{bias}\left(\sum_{i=1, \dots, n; v_i=1} e_i\right) = \epsilon^t,$$

where the second equality follows by the properties of the bias. Consequently, it holds that $\Pr[\langle v, e \rangle = 0] = \frac{1}{2} + \frac{1}{2}\epsilon^t$. \square

2.1 Hard Learning Problems

We consider variants of the learning problems LWE and LPN that are known to be as hard as the original problems. These variants are called extended LWE or LPN, since they leak some additional information about the noise term.

Definition 2 (Extended LWE). *A ppt algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ breaks extended LWE for noise distribution Ψ , m samples, modulus p and dimension κ if*

$$|\Pr[\mathcal{A}_2(\text{st}, A, As + e, x, x^T e) = 1] - \Pr[\mathcal{A}_2(\text{st}, A, B, x, x^T e) = 1]| \geq \epsilon,$$

where $(x, \text{st}) \leftarrow \mathcal{A}_1(1^\kappa)$ and the randomness is taken over $A \leftarrow \mathbb{Z}_p^{m \times \kappa}$, $B \leftarrow \mathbb{Z}_p^m$, $s \leftarrow \mathbb{Z}_p^\kappa$, $e \leftarrow \Psi$ and a non-negligible ϵ .

Lemma 3 [AP12, Theorem 3.1]. *For dimension κ , modulus q with smallest prime divisor p , $m \geq \kappa + \omega(\log(\kappa))$ samples and noise distribution Ψ , if there is an algorithm solving extended LWE with probability ϵ , then there is an algorithm solving LWE with advantage $\frac{\epsilon}{2p-1}$ as long as p is an upper bound on the norm of the hint $x^T e$.*

When $p = 2$ and the noise distribution $\Psi = B_\rho$ is the Bernoulli distribution, we call the problem LPN. The LPN problem was proposed by [BFKL94] for the private key setting. A series of works [Ale03, DMQN12, KMP14, Döt15] provided public key encryption schemes from the so-called *low-noise* LPN problem where the error term has a noise-rate of $O(1/\sqrt{\kappa})$. In a recent work, Yu and Zhang [YZ16] provided public key encryption schemes based on LPN with a constant

noise-rate but a sub-exponential number of samples $m = 2^{O(\sqrt{\kappa})}$. We refer to this variant as (sub-) exponentially hard LPN.

For our LPN based encryption scheme, we need to be able to embed a sufficiently strong binary error correction code such that decryption can recover a message. Therefore, we define a hybrid version of extended LPN that is able to hide a sufficiently large generator matrix of such a code.

Definition 3 (Extended Hybrid LPN). *A ppt algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ breaks extended LPN for noise distribution B_ρ , m samples, modulus p , dimension κ and ℓ hybrids if*

$$|\Pr[\mathcal{A}_2(\text{st}, A, AS + E, x, x^T E) = 1] - \Pr[\mathcal{A}_2(\text{st}, A, B, x, x^T E) = 1]| \geq \epsilon,$$

where $(x, \text{st}) \leftarrow \mathcal{A}_1(1^n)$ and the randomness is taken over $A \leftarrow \mathbb{Z}_p^{m \times \kappa}$, $B \leftarrow \mathbb{Z}_p^{m \times \ell}$, $S \leftarrow \mathbb{Z}_p^{\kappa \times \ell}$, $E \leftarrow B_\rho^{m \times \ell}$ and non-negligible ϵ .

A simple hybrid argument yields that if extended hybrid LPN can be broken with probability ϵ , then extended LPN can be broken with probability ϵ/ℓ . Therefore we consider extended hybrid LPN as hard as extended LPN.

2.2 Weak Commitments

In our LPN-based hash encryption scheme, we will use a list decoding procedure to receive a list of candidate messages during the decryption of a ciphertext. To determine which candidate message has been encrypted, we add a weak form of a commitment of the message to the ciphertext that hides the message. In order to derive the correct message from the list of candidates, we require that the commitment is binding with respect to the list of candidates, i.e. the list decoding algorithm.

Definition 4 (Weak Commitment for List Decoding). *A weak commitment scheme WC_D with respect to a list decoding algorithm D consists of three ppt algorithms Gen , Commit , and Verify , a message space $M \subset \{0, 1\}^*$ and a randomness space $R \subset \{0, 1\}^*$.*

- $\text{Gen}(1^\kappa)$: Outputs a key k .
- $\text{Commit}(k, m, r)$: Outputs a commitment $\text{wC}(m, r)$.
- $\text{Verify}(k, m, r, \text{wC})$: Outputs 1 if and only if $\text{wC}(m, r) = \text{wC}$.

For hiding, we require that for any ppt algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

$$|\Pr[\mathcal{A}_2(\text{st}, \text{wC}(m_0, r)) = 1] - \Pr[\mathcal{A}_2(\text{st}, \text{wC}(m_1, r)) = 1]| \leq \text{negl},$$

where $(m_0, m_1, \text{st}) \leftarrow \mathcal{A}_1(k)$ and the randomness is taken over the random coins of \mathcal{A} , $k \leftarrow \text{Gen}(1^\kappa)$ and $r \leftarrow R$. For binding with respect to D , we require that for any $m \in M$

$$\Pr[\text{Verify}(k, m, r, \text{wC}(m', r')) = 1 \wedge m \neq m'] \leq \text{negl},$$

where the randomness is taken over $(m', r') \leftarrow D(1^n, m, r)$, the random coins of Verify , D , $k \leftarrow \text{Gen}(1^\kappa)$ and $r \leftarrow R$.

Since D does not depend on the key k , a wC_D can be easily instantiated with a universal hash function. The key k corresponds to the hash function h and $wC(m, r) := h(m, r)$ is the hash of m and r . In the following we define universal hash functions and show with two lemmata that our construction of a weak commitment is hiding as well as binding.

Definition 5. For $n, m \in \mathbb{N}$, $m > n$, a family of functions H from $\{0, 1\}^m$ to $\{0, 1\}^n$ is called a family of universal hash functions if for any $x, x' \in \{0, 1\}^m$ with $x \neq x'$

$$\Pr_{h \leftarrow H}[h(x) = h(x')] \leq 2^{-n}.$$

Lemma 4. h is weakly binding with respect to D . In particular,

$$\Pr_{h \leftarrow H}[\exists i \in [\ell] : h(m, r) = h(m_i, r_i) \wedge m \neq m_i] \leq \ell 2^{-n},$$

where $\{(m_i, r_i)\}_{i \in [\ell]} \leftarrow D(1^n, m, r)$ and ℓ is the output list length of D .

Proof. D outputs a list of at most ℓ tuples of the form $(m_1, r_1), \dots, (m_\ell, r_\ell)$. For each of the tuples with $m_i \neq m$,

$$\Pr_{h \leftarrow H}[h(m, r) = h(m_i, r_i)] \leq 2^{-n}$$

holds. Using a union bound, we receive the statement of the lemma.

The work of Hastad et al. [HILL99] shows that for an r with sufficient entropy, for any m , $h(r, m)$ is statistical close to uniform. Therefore it statistically hides the message m .

Lemma 5 ([HILL99] Lemma 4.5.1). *Let h be a universal hash function from $\{0, 1\}^m$ to $\{0, 1\}^n$ and $r \leftarrow \{0, 1\}^{|r|}$ for $|r| \geq 2\kappa + n$, then for any m , $h(r, m)$ is statistically close to uniform given h .*

2.3 Secret- and Public-Key Encryption

We will briefly review the security notions for secret- and public-key encryption this work is concerned with.

Definition 6. A secret-key encryption scheme SKE consists of two algorithms Enc and Dec with the following syntax

- Enc(k, m): Takes as input a key $k \in \{0, 1\}^\kappa$ and a message $m \in \{0, 1\}^\ell$ and outputs a ciphertext c .
- Dec(k, ct): Takes as input a key $k \in \{0, 1\}^\kappa$ and a ciphertext ct and outputs a message m .

For correctness, for all $k \in \{0, 1\}^\kappa$ and $m \in \{0, 1\}^\ell$ we have:

$$\text{Dec}(k, \text{Enc}(k, m)) = m.$$

The standard security notion of secret-key encryption is indistinguishability under chosen plaintext attacks (IND-CPA). However, the notion of interest in this work is the stronger notion of key-dependent-message security under chosen-plaintext attacks. A secret-key encryption scheme $\text{SKE} = (\text{Enc}, \text{Dec})$ is called key-dependent-message secure under chosen plaintext attacks (KDM^{CPA}) if for every PPT-adversary \mathcal{A} the advantage

$$\text{Adv}_{\text{KDM}^{\text{CPA}}}(\mathcal{A}) = \left| \Pr[\text{KDM}^{\text{CPA}}(\mathcal{A}) = 1] - \frac{1}{2} \right|$$

is at most negligible advantage in the following experiment (Fig. 2):

Experiment $\text{KDM}^{\text{CPA}}(\mathcal{A})$:

1. $k \xleftarrow{\$} \{0, 1\}^\kappa$
2. $b^* \xleftarrow{\$} \{0, 1\}$
3. $b' \leftarrow \mathcal{A}^{\text{KDM}_{b^*, k}(\cdot)}(1^\kappa)$
 where the oracle KDM is defined by $\text{KDM}_{0, k}(f) = \text{SKE}.\text{Enc}(k, f(k))$
 and $\text{KDM}_{1, k}(f) = \text{SKE}.\text{Enc}(k, 0^\ell)$.
4. Output 1 if $b' = b^*$ and 0 otherwise.

Fig. 2. The $\text{KDM}^{\text{CPA}}(\mathcal{A})$ experiment

Definition 7. A public-key encryption scheme PKE consists of three (randomized) algorithms KeyGen , Enc and Dec with the following syntax.

- $\text{KeyGen}(1^\kappa)$: Takes as input the security parameter 1^κ and outputs a pair of public and secret keys (pk, sk) .
- $\text{Enc}(\text{pk}, m)$: Takes as input a public key pk and a message $m \in \{0, 1\}^\ell$ and outputs a ciphertext c .
- $\text{Dec}(\text{sk}, c)$: Takes as input a secret key sk and a ciphertext c and outputs a message m .

In terms of correctness, we require that for all messages $m \in \{0, 1\}^\ell$ and $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\kappa)$ that

$$\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, m)) = m.$$

A public-key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ is called IND^{CPA} -secure, if for every PPT-adversary \mathcal{A} the advantage

$$\text{Adv}_{\text{IND}^{\text{CPA}}}(\mathcal{A}) = \left| \Pr[\text{IND}^{\text{CPA}}(\mathcal{A}) = 1] - \frac{1}{2} \right|$$

is at most negligible in the following experiment (Fig. 3):

Experiment $\text{IND}^{\text{CPA}}(\mathcal{A})$:

1. $(\text{pk}, \text{sk}) \leftarrow \text{PKE.KeyGen}(1^\kappa)$
2. $(\text{m}_0, \text{m}_1) \leftarrow \mathcal{A}_1(\text{pk})$
3. $b^* \xrightarrow{\$} \{0, 1\}$
4. $\text{c}^* \leftarrow \text{PKE.Enc}(\text{pk}, \text{m}_{b^*})$
5. $b' \leftarrow \mathcal{A}_2(\text{pk}, \text{c}^*)$
6. Output 1 if $b' = b^*$ and 0 otherwise.

Fig. 3. The $\text{IND}^{\text{CPA}}(\mathcal{A})$ experiment

A public-key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$ is called key-dependent-message secure under chosen plaintext attacks (KDM^{CPA}), if for every PPT-adversary \mathcal{A} the advantage

$$\text{Adv}_{\text{KDM}^{\text{CPA}}}(\mathcal{A}) = \left| \Pr[\text{KDM}^{\text{CPA}}(\mathcal{A}) = 1] - \frac{1}{2} \right|$$

is at most negligible in the following experiment (Fig. 4):

Experiment $\text{KDM}^{\text{CPA}}(\mathcal{A})$:

1. $(\text{pk}, \text{sk}) \leftarrow \text{PKE.KeyGen}(1^\kappa)$
2. $b^* \xrightarrow{\$} \{0, 1\}$
3. $b' \leftarrow \mathcal{A}^{\text{KDM}_{b^*, \text{sk}}(\cdot)}(\text{pk})$
 where the oracle KDM is defined by $\text{KDM}_{0, \text{sk}}(f) = \text{PKE.Enc}(\text{pk}, f(\text{sk}))$ and $\text{KDM}_{1, \text{sk}}(f) = \text{PKE.Enc}(\text{pk}, 0^\ell)$.
4. Output 1 if $b' = b^*$ and 0 otherwise.

Fig. 4. The $\text{KDM}^{\text{CPA}}(\mathcal{A})$ experiment

2.4 One-Time Signatures with Encryption [DG17a]

Definition 8. A One-Time Signature Scheme with Encryption consists of five algorithms (SSetup, SGen, SSign, SEnc, SDec) defined as follows:

- SSetup($1^\kappa, \ell$): Takes as input a unary encoding of the security parameter 1^κ and a message length parameter ℓ and outputs public parameters pp .
- SGen(pp): Takes as input public parameters pp and outputs a pair (vk, sk) of verification and signing keys.
- SSign(sk, x): Takes as input a signing key sk and a message $x \in \{0, 1\}^\ell$ and outputs a signature σ .
- SEnc($\text{pp}, (\text{vk}, i, b), m$): Takes as input public parameters pp , a verification key vk , an index i , a bit b and a plaintext m and outputs a ciphertext c . We will generally assume that the index i and the bit b are included alongside.
- SDec($\text{pp}, (\text{vk}, \sigma, x), c$): Takes as input public parameters pp , a verification key vk , a signature σ , a message x and a ciphertext c and returns a plaintext m .

We require the following properties.

- **Compactness:** For $\text{pp} \leftarrow \text{SSetup}(1^\kappa, \ell)$ and $(\text{vk}, \text{sk}) \leftarrow \text{SGen}(\text{pp})$ it holds that $|\text{vk}| < \ell$, i.e. vk can be described with less than ℓ bits.
- **Correctness:** For all security parameters κ , message $x \in \{0, 1\}^\ell$, $i \in [\ell]$ and plaintext m : If $\text{pp} \leftarrow \text{SSetup}(1^\kappa, \ell)$, $(\text{vk}, \text{sk}) \leftarrow \text{SGen}(\text{pp})$ and $\sigma \leftarrow \text{SSign}(\text{sk}, x)$ then

$$\text{SDec}(\text{pp}, (\text{vk}, \sigma, x), \text{SEnc}(\text{pp}, (\text{vk}, i, x_i), m)) = m.$$
- **Selective Security:** For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function $\text{negl}(\cdot)$ such that the following holds:

$$\Pr[\text{IND}^{\text{OTSIG}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \text{negl}(\kappa)$$

where $\text{IND}^{\text{IBE}}(\mathcal{A})$ is shown in Fig. 5.

Experiment $\text{IND}^{\text{OTSIG}}(\mathcal{A})$:

1. $\text{pp} \leftarrow \text{SSetup}(1^\kappa, \ell)$
2. $(\text{vk}, \text{sk}) \leftarrow \text{SGen}(\text{pp})$
3. $x \leftarrow \mathcal{A}_1(\text{pp}, \text{vk})$
4. $\sigma \leftarrow \text{SSign}(\text{sk}, x)$
5. $(i, m_0, m_1) \leftarrow \mathcal{A}_2(\text{pp}, \text{vk}, \sigma)$
6. $b^* \xleftarrow{\$} \{0, 1\}$
7. $m^* \leftarrow m_{b^*}$
8. $c^* \leftarrow \text{SEnc}(\text{pp}, (\text{vk}, i, 1 - x_i), m^*)$
9. $b' \leftarrow \mathcal{A}_3(\text{pp}, \text{vk}, \sigma, c^*)$
10. Output 1 if $b' = b^*$ and 0 otherwise.

Fig. 5. The $\text{IND}^{\text{OTSIG}}(\mathcal{A})$ experiment

We remark that multi-challenge security (i.e. security in an experiment in which the adversary gets to see an arbitrary number of challenge-ciphertexts) follows via a simple hybrid argument. We also remark that in the definition of [DG17a], the message x was not allowed to depend on vk . The definition given here is stronger and readily implies the definition of [DG17a].

If an OTSE scheme does not fulfill the compactness property, then we refer to such a scheme as a non-compact OTSE-scheme or NC-OTSE.

Döttling and Garg [DG17a] showed that (compact) OTSE implies both fully secure IBE and selectively secure HIBE.

Theorem 2 (Informal). *Assume there exists an OTSE-scheme. Then there exists a fully secure IBE-scheme and a HIBE-scheme.*

2.5 Garbled Circuits

Garbled circuits were first introduced by Yao [Yao82] (see Lindell and Pinkas [LP09] and Bellare et al. [BHR12] for a detailed proof and further discussion). A projective circuit garbling scheme is a tuple of PPT algorithms (Garble, Eval) with the following syntax.

- $\text{Garble}(1^\kappa, C)$ takes as input a security parameter κ and a circuit C and outputs a *garbled circuit* \tilde{C} and labels $e_C = \{X_{\ell,0}, X_{\ell,1}\}_{\ell \in [n]}$, where n is the number of input wires of C .
- Projective Encoding: To encode an $x \in \{0,1\}^n$ with the input labels $e_C = \{X_{\ell,0}, X_{\ell,1}\}_{\ell \in [n]}$, we compute $\tilde{x} \leftarrow \{X_{\ell,x_\ell}\}_{\ell \in [n]}$.
- $\text{Eval}(\tilde{C}, \tilde{x})$: takes as input a garbled circuit \tilde{C} and a garbled input \tilde{x} , represented as a sequence of input labels $\{X_{\ell,x_\ell}\}_{\ell \in [n]}$, and outputs an output y .

We will denote hardwiring of an input s into a circuit C by $C[s]$. The garbling algorithm Garble treats the hardwired input as a regular input and additionally outputs the garbled input corresponding to s (instead of all the labels of the input wires corresponding to s). If a circuit C uses additional randomness, we will implicitly assume that appropriate random coins are hardwired in this circuit during garbling.

Correctness. For correctness, we require that for any circuit C and input $x \in \{0,1\}^n$ we have that

$$\Pr \left[C(x) = \text{Eval}(\tilde{C}, \tilde{x}) \right] = 1$$

where $(\tilde{C}, e_C = \{X_{\ell,0}, X_{\ell,1}\}_{\ell \in [n]}) \stackrel{\$}{\leftarrow} \text{Garble}(1^\kappa, C)$ and $\tilde{x} \leftarrow \{X_{\ell,x_\ell}\}$.

Security. For security, we require that there is a PPT simulator GCSim such that for any circuit C and any input x , we have that

$$(\tilde{C}, \tilde{x}) \approx_c \text{GCSim}(C, C(x))$$

where $(\tilde{C}, e_C = \{X_{l,0}, X_{l,1}\}_{l \in [n]}) \leftarrow \text{Garble}(1^\kappa, C)$ and $\tilde{x} \leftarrow \{X_{l,x_l}\}$.

3 Hash Encryption from Learning Problems

Intuitively, our hash encryption scheme can be seen as a witness encryption scheme that uses a hash value and a key to encrypt a message. The decryption procedure requires the knowledge of a preimage of the hash value to recover an encrypted message. Given key k , an algorithm `Hash` allows to compute a hash value for an input x . This hashing procedure is tied to the hash encryption scheme. More concretely, the encryption procedure encrypts a message with respect to a bit c for an index i . Given knowledge of a preimage, where the i th bit has the value c , one can successfully decrypt the initially encrypted message. Due to this additional constraint, a hash encryption is more restrictive than a witness encryption for the knowledge of the preimage of a hash value.

3.1 Hash Encryption

Definition 9 (Hash Encryption). *A hash encryption (HE) consists of four ppt algorithms `Gen`, `Hash`, `Enc` and `Dec` with the following syntax*

- `Gen`($1^\kappa, m$): Takes as input the security parameter κ , an input length m and outputs a key k .
- `Hash`(k, x): Takes a key k , an input $x \in \{0, 1\}^m$ and outputs a hash value h of κ bits.
- `Enc`($k, (h, i, c), m$): Takes a key k , a hash value h an index $i \in [m]$, $c \in \{0, 1\}$ and a message $m \in \{0, 1\}^*$ as input and outputs a ciphertext ct . We will generally assume that the index i and the bit c are included alongside.
- `Dec`(k, x, ct): Takes a key k , an input x and a ciphertext ct as input and outputs a value $m \in \{0, 1\}^*$ (or \perp).

Correctness. For correctness, we require that for any input $x \in \{0, 1\}^m$, index $i \in [m]$

$$\Pr[\text{Dec}(k, x, \text{Enc}(k, (\text{Hash}(k, x), i, x_i), m)) = m] \geq 1 - \text{negl},$$

where x_i denotes the i th bit of x and the randomness is taken over $k \leftarrow \text{Gen}(1^\kappa, m)$.

Security. We call a HE secure, i.e. selectively indistinguishable, if for any ppt algorithm \mathcal{A}

$$\Pr[\text{IND}^{\text{HE}}(1^\kappa, \mathcal{A}) = 1] \leq \frac{1}{2} + \text{negl},$$

where the game IND^{HE} is defined in Fig. 6.

Experiment $\text{IND}^{\text{HE}}(\mathcal{A})$:

1. $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$
2. $k \leftarrow \text{Gen}(1^\kappa, m)$
3. $(i \in [m], m_0, m_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, k)$
4. $b \leftarrow \{0, 1\}$
5. $\text{ct} \leftarrow \text{Enc}(k, (\text{Hash}(k, x), i, 1 - x_i), m_b)$
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct})$
7. Output 1 if $b' = b$ and 0 otherwise.

Fig. 6. The $\text{IND}^{\text{HE}}(\mathcal{A})$ experiment**3.2 Hash Encryption from LWE**

We use the same parameters as proposed by the PKE of [Reg05], i.e. Gaussian noise distribution $\Psi_{\alpha(\kappa)}$ for $\alpha(\kappa) = o\left(\frac{1}{\sqrt{\kappa} \log(\kappa)}\right)$, prime modulus $\kappa^2 \leq p \leq 2\kappa^2$, $m = (1 + \epsilon)(1 + \kappa) \log(\kappa)$ for $\epsilon > 0$. For hash domain $\{0, 1\}^m$ and message space $M = \{0, 1\}$, we define our LWE based HE as follows.

- $\text{Gen}(1^\kappa, m)$: Sample $A \leftarrow \mathbb{Z}_p^{m \times \kappa}$.
- $\text{Hash}(k, x)$: Output $x^T A$.
- $\text{Enc}(k, (h, i, c), m)$: Sample $s \leftarrow \mathbb{Z}_p^\kappa$, $e \leftarrow \Psi_{\alpha(\kappa)}^m$ and compute

$$\begin{aligned} c_1 &:= A_{-i}s + e_{-i} \\ c_2 &:= (h - c \cdot a_i)s + e_i + \lfloor p/2 \rfloor \cdot m. \end{aligned}$$

Output $\text{ct} = (c_1, c_2)$.

- $\text{Dec}(k, x, \text{ct})$: Output 1 if $c_2 - x_{-i}^T c_1$ is closer to $p/2$ than to 0 and otherwise output 0.

Depending on the concrete choice of $m = (1 + \epsilon)(1 + \kappa) \log(\kappa)$, the compression factor of the hash function is determined. For our purposes, the construction of an IBE, any choice of $\epsilon > 0$ is sufficient.

Lemma 6. *For the proposed parameters, the LWE based HE is correct.*

Proof. If $\text{ct} = (c_1, c_2)$ is an output of $\text{Enc}(k, (h, i, c), m)$, then for any x with $\text{Hash}(k, x) = h$, c_2 has the form

$$c_2 = (x^T A - c \cdot a_i)s + e_i + \lfloor p/2 \rfloor \cdot m.$$

Therefore, on input x , $c = x_i$, Dec computes

$$\begin{aligned} c_2 - x_{-i}^T c_1 &= (x^T A - c \cdot a_i)s + e_i + \lfloor p/2 \rfloor \cdot m - x_{-i}^T A_{-i}s - x_{-i}^T e_{-i} \\ &= (x_i - c) \cdot a_i s + e_i + \lfloor p/2 \rfloor \cdot m - x_{-i}^T e_{-i} \\ &= \lfloor p/2 \rfloor \cdot m + e_i - x_{-i}^T e_{-i}. \end{aligned}$$

By [Reg05, Claim 5.2], for any $x \in \{0, 1\}^m$, $|e_i - x_{-i}^T e_{-i}| < p/4$ holds with overwhelming probability. Hence, the noise is sufficiently small such that Dec outputs m . \square

Theorem 3. *The LWE based HE is IND^{HE} secure under the extended LWE assumption for dimension κ , Gaussian noise parameter $\alpha(n) = o(\frac{1}{\sqrt{n} \log(n)})$, prime modulus $\kappa^2 \leq p \leq 2\kappa^2$, and $m = (1 + \epsilon)(1 + \kappa) \log(n)$ samples.*

Proof. For proving the theorem, we will show that if there is an adversary \mathcal{A} that successfully breaks the IND^{HE} security of the proposed HE then there is an algorithm \mathcal{A}' that breaks the extended LWE assumption with the same probability.

We construct $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ as follows:

1. $\mathcal{A}'_1(1^\kappa)$: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$, Return x
2. $\mathcal{A}'_2(x, A, B, x^T e)$: $k := A$
3. $(i \in [m], m_0, m_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, k)$
4. $b \leftarrow \{0, 1\}$
5. $c_1 := B_{-i}$, $c_2 := (-1)^{x_i+1} B_i + \lfloor p/2 \rfloor \cdot m_b - x_{-i}^T e_{-i} + x_{-i}^T c_1$
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct} = (c_1, c_2))$
7. Return 1 if $b' = b$ and 0 otherwise.

In the LWE case, $B = As + e$. Therefore \mathcal{A}' creates ct with the same distribution as in game IND^{HE} . This is easy to see for $c_1 = B_{-i} = A_{-i}s + e_{-i}$. For c_2 , we have

$$\begin{aligned}
 c_2 &= (-1)^{x_i+1} B_i + \lfloor p/2 \rfloor \cdot m_b - x_{-i}^T e_{-i} + x_{-i}^T c_1 \\
 &= (-1)^{x_i+1} a_i s + (-1)^{x_i+1} e_i + \lfloor p/2 \rfloor \cdot m_b - x_{-i}^T e_{-i} + x_{-i}^T A_{-i} s + x_{-i}^T e_{-i} \\
 &= (-1)^{x_i+1} a_i s + (-1)^{x_i+1} e_i + \lfloor p/2 \rfloor \cdot m_b + x_{-i}^T A_{-i} s \\
 &= (h - ((-1)^{x_i} + x_i) a_i) s + (-1)^{x_i+1} e_i + \lfloor p/2 \rfloor \cdot m_b \\
 &= (h - (1 - x_i) a_i) s + (-1)^{x_i+1} e_i + \lfloor p/2 \rfloor \cdot m_b.
 \end{aligned}$$

Notice since e_i is Gaussian with mean 0, $-e_i$ and e_i have the same distribution.

In the uniform case, B is uniform and therefore \mathcal{A}' 's guess b' is independent of b . Hence, \mathcal{A}'_2 outputs 1 with probability $\frac{1}{2}$. \mathcal{A}' breaks extended LWE with advantage

$$\begin{aligned}
 &|\Pr[\mathcal{A}_3(\text{st}', A, As + e, x, x^T e) = 1] - \Pr[\mathcal{A}_3(\text{st}', A, B, x, x^T e) = 1]| \\
 &= \left| \Pr[\text{IND}^{\text{HE}}(\mathcal{A}) = 1] - \frac{1}{2} \right|.
 \end{aligned}$$

\square

3.3 Hash Encryption from Exponentially Hard LPN

For LPN, we use a Bernoulli noise distribution B_ρ with Bernoulli parameter $\rho = c_\rho$ and hash domain $x \in \{0, 1\}_k^m$, where $k = c_k \log(\kappa)$ for constants c_ρ and c_k . $G \in \mathbb{Z}_2^{(|m|+\kappa) \times \ell}$ is the generator matrix of a binary, list decodeable error correction code that corrects an error with $1/\text{poly}$ bias, where $|m|$ is the message length and ℓ the dimension of the codewords. For this task, we can use the error correction code proposed by Guruswami and Rudra [GR11]. Further, we use a weak commitment scheme WC with respect to the list decoding algorithm of G .

- $\text{Gen}(1^\kappa, m)$: Sample $A \leftarrow \mathbb{Z}_2^{m \times \log^2(\kappa)}$, output $k := A$.
- $\text{Hash}(k, x)$: Output $x^T A$.
- $\text{Enc}(k, (h, i, c), m)$: Sample $S \leftarrow \mathbb{Z}_2^{\log^2(\kappa) \times \ell}$, $E \leftarrow B_\rho^{m \times \ell}$, and a random string $r \leftarrow \text{R}_{\text{WC}}$ and compute

$$\begin{aligned} c_0 &:= k_{\text{WC}} \leftarrow \text{Gen}_{\text{WC}}(1^\kappa) \\ c_1 &:= A_{-i} S + E_{-i} \\ c_2 &:= (h - c \cdot a_i) S + E_i + (m || r) \cdot G \\ c_3 &:= \text{wC}(m, r) \leftarrow \text{Commit}(k_{\text{WC}}, m, r). \end{aligned}$$

Output $\text{ct} = (c_0, c_2, c_3)$.

- $\text{Dec}(k, x, \text{ct})$: Use code G to list decode $c_2 - x^T_i c_1$. Obtain from the list of candidates the candidate $(m || r)$ that fits $\text{Verify}(c_0, m, r, c_3) = 1$. Output this candidate.

The choice of the constant c_k will determine the compression factor of the hash function Hash . The compression is determined by the ratio between $|\{0, 1\}_k^m|$ and the space of the LPN secret $2^{\log^2(\kappa)}$. By Lemma 1, the cardinality of $|\{0, 1\}_k^m|$ is lower bounded by $(\frac{m}{c_k \log(\kappa)})^{c_k \log(\kappa)}$. $m := c\kappa$ yields a compression factor of at least $c_k(c - \frac{\log(c_k \log(\kappa))}{\log \kappa})$, which allows any constant compression factor for a proper choice of the constants c and c_k .

For the correctness, we need to rely on the error correction capacity of code G and the binding property of the weak commitment scheme. For properly chosen constants c_ρ and k , the proposed HE is correct.

Lemma 7. *For $\rho = c_\rho \leq \frac{1}{4}$, $k = c_k \log(\kappa)$, and an error correction code G that corrects an error with a bias of $2^{-4c_\rho \kappa^{-4c_\rho c_k}}$ and let WC be a weak commitment that is binding with respect to the list decoding of G , then the LPN based HE is correct.*

Proof. $\text{ct} = (c_0, c_1, c_2, c_3)$ is an output of $\text{Enc}(k, (h, i, c), m)$, then for any x with $\text{Hash}(k, x) = h$, c_2 has the form

$$c_2 = (x^T A - c \cdot a_i) S + E_i + (m || r) \cdot G.$$

Therefore, on input \mathbf{x} , $c = x_i$, Dec computes

$$\begin{aligned} c_2 - \mathbf{x}_{-i}^T c_1 &= (\mathbf{x}^T A - c \cdot a_i)S + E_i + (\mathbf{m} \parallel \mathbf{r}) \cdot G - \mathbf{x}_{-i}^T A_{-i} S - \mathbf{x}_{-i}^T E_{-i} \\ &= (x_i - c) \cdot a_i S + E_i + (\mathbf{m} \parallel \mathbf{r}) \cdot G - \mathbf{x}_{-i}^T E_{-i} \\ &= (\mathbf{m} \parallel \mathbf{r}) \cdot G + E_i - \mathbf{x}_{-i}^T E_{-i}. \end{aligned}$$

By Lemma 2, for each component e_j , $j \in [\ell]$ of $e := E_i - \mathbf{x}_{-i}^T E_{-i}$ and $\rho \leq \frac{1}{4}$,

$$\begin{aligned} \rho' &:= \Pr[e_j = 1] = \frac{1}{2}(1 - (1 - 2\rho)^{k+1}) \leq \frac{1}{2} \left(1 - 2^{-4c_\rho(c_k \log(\kappa)+1)}\right) \\ &= \frac{1}{2} \left(1 - 2^{-4c_\rho} \kappa^{-4c_\rho c_k}\right). \end{aligned}$$

This lower bounds the bias of each component of the noise term $E_i - \mathbf{x}_{-i}^T E_{-i}$ by bound $2^{-4c_\rho} \kappa^{-4c_\rho c_k}$. This bound is polynomial in κ and therefore correctable by a suitable error correction code with list decoding. Hence, $(\mathbf{m} \parallel \mathbf{r})$ is contained in the output list of candidates of the list decoding. By the binding of WC, there is with overwhelming probability only a single candidate of the polynomially many candidates that fits $\text{Verify}(c_0, \mathbf{m}, \mathbf{r}, c_3) = 1$, which corresponds to the initially encrypted message \mathbf{m} . Otherwise, the list decoding of G would break the binding property of WC. \square

The security analysis is simliar to the one of the LWE based scheme with the difference that now a ciphertext also contains a commitment which depends on the encrypted message. In a first step, we use the LPN assumption to argue that all parts of the ciphertext are computationally independent of the message. In a second step, we use the hiding property of the commitment scheme to argue that now the whole ciphertext is independent of the encrypted message and therefore an adversary cannot break the scheme.

Theorem 4. *Let WC be a weak commitment scheme that is hiding, then the LPN based HE is IND^{HE} secure under the extended hybrid LPN assumption for dimension $\log^2(\kappa)$, m samples, ℓ hybrids and noise level ρ .*

Proof. Consider the following hybrid experiments:

Hybrid \mathcal{H}_1 :

1. $(\mathbf{x}, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$
2. $\mathbf{k} := A \leftarrow \text{Gen}(1^\kappa, m)$
3. $(i \in [m], \mathbf{m}_0, \mathbf{m}_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, \mathbf{k})$
4. $b \leftarrow \{0, 1\}$
5. $S \leftarrow \mathbb{Z}_2^{\log^2(\kappa) \times \ell}$, $E \leftarrow B_\rho^{m \times \ell}$, $\mathbf{r} \leftarrow \text{R}_{\text{WC}}$,
 $c_0 := \text{k}_{\text{WC}} \leftarrow \text{Gen}_{\text{WC}}(1^\kappa)$, $c_1 := A_{-i} S + E_{-i}$, $c_2 := (\mathbf{h} - (1 - x_i) \cdot a_i)S + E_i + (\mathbf{m}_b \parallel \mathbf{r}) \cdot G$, $c_3 := \text{wC}(\mathbf{m}_b, \mathbf{r}) \leftarrow \text{Commit}(\text{k}_{\text{WC}}, \mathbf{m}_b, \mathbf{r})$,
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct} = (c_0, c_1, c_2, c_3))$
7. Return 1 if $b' = b$ and 0 otherwise.

Hybrid \mathcal{H}_2 :

1. $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$
2. $k := A \leftarrow \text{Gen}(1^\kappa, m)$
3. $(i \in [m], m_0, m_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, k)$
4. $b \leftarrow \{0, 1\}$
5. $B \leftarrow \mathbb{Z}_2^{m \times \ell}$, $r \leftarrow \text{R}_{\text{WC}}$,
 $c_0 := k_{\text{WC}} \leftarrow \text{Gen}_{\text{WC}}(1^\kappa)$, $c_1 := B_{-i}$, $c_2 := B_i$, $c_3 := \text{wC}(m_b, r) \leftarrow$
 $\text{Commit}(k_{\text{WC}}, m_b, r)$,
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct} = (c_0, c_1, c_2, c_3))$
7. Return 1 if $b' = b$ and 0 otherwise.

Lemma 8. *Let \mathcal{A} be an adversary that distinguishes \mathcal{H}_1 and \mathcal{H}_2 with advantage ϵ . Then there is an algorithm \mathcal{A}' that breaks the extended hybrid LPN assumption with advantage ϵ .*

Proof. We construct $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ as follows:

1. $\mathcal{A}'_1(1^\kappa)$: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$ Return x
2. $\mathcal{A}'_2(\text{st}_1, x, A, B, x^T E)$: $k := A$
3. $(i \in [m], m_0, m_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, k)$
4. $b \leftarrow \{0, 1\}$
5. $r \leftarrow \text{R}_{\text{WC}}$,
 $c_0 := k_{\text{WC}} \leftarrow \text{Gen}_{\text{WC}}(1^\kappa)$, $c_1 := B_{-i}$, $c_2 := B_i + (m_b || r) \cdot G - x_{-i}^T E_{-i} + x_{-i}^T c_1$,
 $c_3 := \text{wC}(m_b, r) \leftarrow \text{Commit}(k_{\text{WC}}, m_b, r)$,
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct} = (c_0, c_1, c_2, c_3))$
7. Return 1 if $b' = b$ and 0 otherwise.

In the LPN case, $B = AS + E$. Therefore \mathcal{A}' creates ct with the same distribution as in game IND^{HE} . This is easy to see for c_0 , c_3 and $c_1 = B_{-i} = A_{-i}S + E_{-i}$. For c_2 , we have

$$\begin{aligned}
 c_2 &= B_i + (m_b || r) \cdot G - x_{-i}^T E_{-i} + x_{-i}^T c_1 \\
 &= a_i S + E_i + (m_b || r) \cdot G - x_{-i}^T E_{-i} + x_{-i}^T A_{-i} S + x_{-i}^T E_{-i} \\
 &= a_i S + E_i + (m_b || r) \cdot G + x_{-i}^T A_{-i} S \\
 &= (h + (1 - x_i) a_i) S + E_i + (m_b || r) \cdot G,
 \end{aligned}$$

which results in the same distribution over \mathbb{Z}_2 .

In the uniform case, B and hence c_2 are uniform. Therefore \mathcal{A}' simulates \mathcal{H}_2 . \mathcal{A}' breaks extended hybrid LPN with advantage

$$\begin{aligned}
 &|\Pr[\mathcal{A}_2(\text{st}_1, x, A, AS + E, x, x^T E) = 1] - \Pr[\mathcal{A}_2(\text{st}_1, x, A, B, x, x^T E) = 1]| \\
 &= |\Pr[\mathcal{H}_1(1^\kappa, \mathcal{A}) = 1] - \Pr[\mathcal{H}_2(1^\kappa, \mathcal{A}) = 1]|.
 \end{aligned}$$

□

Lemma 9. *If there is an adversary \mathcal{A} with $\Pr[\mathcal{H}_2(1^\kappa, \mathcal{A}) = 1] = \frac{1}{2} + \epsilon$, then there is an algorithm \mathcal{A}' that breaks the hiding property of WC with advantage 2ϵ .*

Proof. We construct $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ as follows.

1. $\mathcal{A}'_1(k_{\text{WC}})$: $(x, \text{st}_1) \leftarrow \mathcal{A}_1(1^\kappa)$
2. $k := A \leftarrow \text{Gen}(1^\kappa, m)$
3. $(i \in [m], m_0, m_1, \text{st}_2) \leftarrow \mathcal{A}_2(\text{st}_1, k)$, Return (m_0, m_1)
4. $\mathcal{A}'_2(k_{\text{WC}}, \text{st}_2, \text{wC})$: $b \leftarrow \{0, 1\}$
5. $B \leftarrow \mathbb{Z}_2^{m \times \ell}$,
 $c_0 := k_{\text{WC}}, c_1 := B_{-i}, c_2 := B_i, c_3 := \text{wC}$
6. $b' \leftarrow \mathcal{A}_3(\text{st}_2, \text{ct} = (c_0, c_1, c_2, c_3))$
7. Return 1 if $b' = b$ and 0 otherwise.

It is easy to see that \mathcal{A}' correctly simulates \mathcal{H}_2 . When \mathcal{A} guesses b with his guess b' correctly, then also \mathcal{A}' does. Therefore

$$\begin{aligned} & \frac{1}{2} \Pr[\mathcal{A}'_2(k_{\text{WC}}, \text{st}_2, \text{wC}(m_1, r)) = 1] + \frac{1}{2} \Pr[\mathcal{A}'_2(k_{\text{WC}}, \text{st}_2, \text{wC}(m_0, r)) = 0] \\ &= \Pr[\mathcal{H}_2(1^\kappa, \mathcal{A}) = 1] = \frac{1}{2} + \epsilon. \end{aligned}$$

Hence,

$$\Pr[\mathcal{A}'_2(k_{\text{WC}}, \text{st}_2, \text{wC}(m_1, r)) = 1] - \Pr[\mathcal{A}'_2(k_{\text{WC}}, \text{st}_2, \text{wC}(m_0, r)) = 1] = 2\epsilon.$$

□

□

4 Non-compact One-Time Signatures with Encryption

In this Section we will construct a *non-compact* OTSE scheme NC from any public-key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$.

- $\text{SSetup}(1^\kappa, \ell)$: Output $\text{pp} \leftarrow (1^\kappa, \ell)$.
- $\text{SGen}(\text{pp})$: For $j = \{1, \dots, \ell\}$ and $b \in \{0, 1\}$ compute $(\text{pk}_{j,b}, \text{sk}_{j,b}) \leftarrow \text{PKE.KeyGen}(1^\kappa)$. Set $\text{vk} \leftarrow \{\text{pk}_{j,0}, \text{pk}_{j,1}\}_{j \in [\ell]}$ and $\text{sgk} \leftarrow \{\text{sk}_{j,0}, \text{sk}_{j,1}\}_{j \in [\ell]}$. Output (vk, sgk) .
- $\text{SSign}(\text{pp}, \text{sgk} = \{\text{sk}_{j,0}, \text{sk}_{j,1}\}_{j \in [\ell]}, x)$: Output $\sigma \leftarrow \{\text{sk}_{j,x_j}\}_{j \in [\ell]}$.
- $\text{SEnc}(\text{pp}, (\text{vk} = \{\text{pk}_{j,0}, \text{pk}_{j,1}\}_{j \in [\ell]}, i, b), m)$: Output $c \leftarrow \text{PKE.Enc}(\text{pk}_{i,b}, m)$.
- $\text{SDec}(\text{pp}, (\text{vk}, \sigma = \{\text{sk}_{j,x_j}\}_{j \in [\ell]}, x), c)$: Output $m \leftarrow \text{PKE.Dec}(\text{sk}_{i,x_i}, c)$.

Correctness of this scheme follows immediately from the correctness of PKE.

Security. We will now establish the $\text{IND}^{\text{OTSIG}}$ -security of NC from the IND^{CPA} -security of PKE.

Theorem 5. *Assume that PKE is IND^{CPA} -secure. Then NC is $\text{IND}^{\text{OTSIG}}$ -secure.*

Proof. Let \mathcal{A} be a PPT-adversary against $\text{IND}^{\text{OTSIG}}$ with advantage ϵ . We will construct an adversary \mathcal{A}' against the IND^{CPA} experiment with advantage $\frac{\epsilon}{2\ell}$. \mathcal{A}' gets as input a public key pk of the PKE and will simulate the $\text{IND}^{\text{OTSIG}}$ -experiment to \mathcal{A} . \mathcal{A}' first guesses an index $i^* \xleftarrow{\$} [\ell]$ and a bit $b^* \xleftarrow{\$} \{0, 1\}$, sets $\text{pk}_{i^*, b^*} \leftarrow \text{pk}$ and generates $2\ell - 1$ pairs of public and secret keys $(\text{pk}_{j,b}, \text{sk}_{j,b}) \leftarrow \text{KeyGen}(1^\kappa)$ for $j \in [\ell]$ and $b \in \{0, 1\}$ with the restriction that $(j, b) \neq (i^*, b^*)$. \mathcal{A}' then sets $\text{vk} \leftarrow \{\text{pk}_{j,0}, \text{pk}_{j,1}\}_{j \in [\ell]}$ and runs \mathcal{A} on input vk . If it holds for the message x output by \mathcal{A} that $x_{i^*} = b^*$, then \mathcal{A}' aborts the simulation and outputs a random bit. Once \mathcal{A} outputs (m_0, m_1, i) , \mathcal{A}' checks if $(i, b) = (i^*, b^*)$ and if not aborts and outputs a random bit. Otherwise, \mathcal{A}' sends the message-pair (m_0, m_1) to the IND^{CPA} -experiment and receives a challenge-ciphertext c^* . \mathcal{A}' now forwards c^* to \mathcal{A} and outputs whatever \mathcal{A} outputs.

First notice that the verification key vk computed by \mathcal{A}' is identically distributed to the verification key in the $\text{IND}^{\text{OTSIG}}$ experiment. Thus, vk does not reveal the index i^* and the bit b^* , and consequently it holds that $(i, b) = (i^*, b^*)$ with probability $\frac{1}{2\ell}$. Conditioned on the event that $(i, b) = (i^*, b^*)$, it holds that the advantage of \mathcal{A}' is identical to the advantage of \mathcal{A} . Therefore, it holds that

$$\text{Adv}_{\text{IND}^{\text{CPA}}}(\mathcal{A}') = \frac{\text{Adv}_{\text{IND}^{\text{OTSIG}}}(\mathcal{A})}{2\ell},$$

which concludes the proof. \square

5 Compact One-Time-Signatures with Encryption via Hash-Encryption

In this Section, we will show how a non-compact OTSE scheme NC can be bootstrapped to a compact OTSE scheme OTSE using hash-encryption. Let $\text{NC} = (\text{SSetup}, \text{SGen}, \text{SSign}, \text{SEnc}, \text{SDec})$ be a non-compact OTSE scheme, $\text{HE} = (\text{HE.Gen}, \text{HE.Hash}, \text{HE.Enc}, \text{HE.Dec})$ be a hash-encryption scheme and $(\text{Garble}, \text{Eval})$ be a garbling scheme. The scheme OTSE is given as follows.

- $\text{SSetup}(1^\kappa, \ell)$: Compute $\bar{\text{pp}} \leftarrow \text{NC.SSetup}(1^\kappa, \ell)$, $k \leftarrow \text{HE.Gen}(1^\kappa, \ell')$ (where ℓ' is the size of the verification keys vk generated using $\bar{\text{pp}}$) and output $\text{pp} \leftarrow (\bar{\text{pp}}, k)$.
- $\text{SGen}(\text{pp} = (\bar{\text{pp}}, k))$: Compute $(\bar{\text{vk}}, \bar{\text{sgk}}) \leftarrow \text{NC.SGen}(\bar{\text{pp}})$. Compute $h \leftarrow \text{HE.Hash}(k, \bar{\text{vk}})$, set $\text{vk} \leftarrow h$, $\text{sgk} \leftarrow (\bar{\text{vk}}, \bar{\text{sgk}})$ and output (vk, sgk) .
- $\text{SSign}(\text{pp} = (\bar{\text{pp}}, k), \text{sgk} = (\bar{\text{vk}}, \bar{\text{sgk}}), x)$: Compute the signature $\sigma' \leftarrow \text{NC.SSign}(\bar{\text{pp}}, \bar{\text{sgk}}, x)$. Output $\sigma \leftarrow (\text{vk}, \sigma')$.
- $\text{SEnc}(\text{pp} = (\bar{\text{pp}}, k), (\text{vk} = h, i, b), m)$: Let C be the following circuit. $C[\bar{\text{pp}}, i, b, m](\bar{\text{vk}})$: Compute and output $\text{NC.SEnc}(\bar{\text{pp}}, (\bar{\text{vk}}, i, b), m)$ ¹.

¹ We also need to hardcode the randomness for NC.SEnc into C , but for ease of notation we omit this parameter.

$(\tilde{C}, e_C) \leftarrow \text{Garble}(1^\kappa, C[\bar{p}\bar{p}, i, b, m])$
 Parse $e_C = \{Y_{j,0}, Y_{j,1}\}_{j \in [\ell']}$
 $f_C \leftarrow \{\text{HE.Enc}(k, (h, j, b'), Y_{j,b'})\}_{j \in [\ell], b' \in \{0,1\}}$
 Output $\text{ct} \leftarrow (\tilde{C}, f_C)$.

– $\text{SDec}(\text{pp} = (\bar{p}\bar{p}, k), (\text{vk} = h, \sigma = (\bar{\text{v}}k, \sigma'), x), \text{ct} = (\tilde{C}, f_C))$:

Parse $f_C = \{c_{j,b'}\}_{j \in [\ell], b' \in \{0,1\}}$
 $y \leftarrow \bar{\text{v}}k$
 $\tilde{y} \leftarrow \{\text{HE.Dec}(k, y, c_{j,y_j})\}_{j \in [\ell]}$
 $c' \leftarrow \text{Eval}(\tilde{C}, \tilde{y})$
 $m \leftarrow \text{NC.SDec}(\bar{p}\bar{p}, (\bar{\text{v}}k, \sigma', x), c')$
 Output m

Compactness and Correctness. By construction, the size of the verification key $\text{vk} = \text{HE.Hash}(k, \bar{\text{v}}k)$ depends on κ , but not on ℓ' or ℓ . Therefore, OTSE is compact.

To see that the scheme is correct, note first that since it holds that $h = \text{HE.Hash}(k, \bar{\text{v}}k)$ and $f_C = \{\text{HE.Enc}(k, (h, j, b'), Y_{j,b'})\}_{j \in [\ell], b' \in \{0,1\}}$, by correctness of the hash-encryption scheme HE we have

$$\tilde{y} = \{\text{HE.Dec}(k, y, c_{j,y_j})\}_{j \in [\ell]} = \{Y_{j,y_j}\}_{j \in [\ell]}.$$

Thus, as $(\tilde{C}, e_C) = \text{Garble}(1^\kappa, C[\bar{p}\bar{p}, i, b, m])$ and by the definition of C , it holds by the correctness of the garbling scheme $(\text{Garble}, \text{Eval})$ that

$$c' = \text{Eval}(\tilde{C}, \tilde{y}) = C[\bar{p}\bar{p}, i, b, m](\bar{\text{v}}k) = \text{NC.SEnc}(\bar{p}\bar{p}, (\bar{\text{v}}k, i, b), m),$$

as $y = \bar{\text{v}}k$. Finally, as $\sigma' = \text{NC.SSign}(\bar{p}\bar{p}, \text{sgk}, x)$ it holds by the correctness of the non-compact OTSE-scheme NC that

$$\text{NC.SDec}(\bar{p}\bar{p}, (\bar{\text{v}}k, \sigma', x), c') = m,$$

which concludes the proof of correctness.

Security. We will now establish the $\text{IND}^{\text{OTSIG}}$ -security of OTSE from the security of the hash-encryption scheme HE, the security of the garbling scheme $(\text{Garble}, \text{Eval})$ and the $\text{IND}^{\text{OTSIG}}$ -security of NC.

Theorem 6. *Assume that HE is an IND^{HE} -secure hash-encryption scheme, $(\text{Garble}, \text{Eval})$ is a secure garbling scheme and NC is $\text{IND}^{\text{OTSIG}}$ -secure. Then OTSE is an $\text{IND}^{\text{OTSIG}}$ -secure OTSE-scheme.*

Proof. Let \mathcal{A} be a PPT-adversary against the $\text{IND}^{\text{OTSIG}}$ -security of OTSE. Consider the following hybrid experiments.

Hybrid \mathcal{H}_0 . This experiment is identical to $\text{IND}^{\text{OTSIG}}(\mathcal{A})$.

Hybrid \mathcal{H}_1 . This experiment is identical to \mathcal{H}_0 , except that f_C is computed by $f_C \leftarrow \{\text{HE.Enc}(k, (h, j, b'), Y_{j,y_j})\}_{j \in [\ell'], b' \in \{0,1\}}$, i.e. for all $j \in [\ell']$ the message Y_{j,y_j} is encrypted, regardless of the bit b' . Computational indistinguishability between \mathcal{H}_0 and \mathcal{H}_1 follows from the IND^{HE} -security of HE. The reduction R first generates the public parameters $\bar{pp} \leftarrow \text{NC.SSetup}(1^\kappa, \ell)$, the keys $(\bar{vk}, \text{sgk}) \leftarrow \text{NC.SGen}(\bar{pp})$ and sends \bar{vk} as its selectively chosen message to the IND^{HE} -experiment. It then obtains k , computes $h \leftarrow \text{HE.Hash}(k, \bar{vk})$ and sets $pp \leftarrow (\bar{pp}, k)$, $vk \leftarrow h$, $\text{sgk} \leftarrow (\bar{vk}, \text{sgk})$ and then simulates \mathcal{H}_0 with these parameters with \mathcal{A} . Instead of computing the ciphertexts f_C by itself, R sends the labels $\{Y_{j,0}, Y_{j,1}\}_{j \in [\ell']}$ to the multi-challenge IND^{HE} -experiment and obtains the ciphertexts f_C . R continues the simulation and outputs whatever \mathcal{A} outputs. Clearly, if the challenge-bit of R 's IND^{HE} -experiment is 0, then from the view of \mathcal{A} the reduction R simulates \mathcal{H}_0 perfectly. On the other hand, if the challenge-bit is 1, then R simulates \mathcal{H}_1 perfectly. Thus R 's advantage is identical to \mathcal{A} 's distinguishing advantage between \mathcal{H}_0 and \mathcal{H}_1 . It follows that \mathcal{H}_0 and \mathcal{H}_1 are computationally indistinguishable, given the IND^{HE} -security of NC.

Hybrid \mathcal{H}_2 . This experiment is identical to \mathcal{H}_1 , except that we compute \tilde{C} by $(\tilde{C}, \tilde{y}) \leftarrow \text{GCSim}(C, C[\bar{pp}, i, b, m](\bar{vk}))$ and the value and f_C by $f_C \leftarrow \{\text{HE.Enc}(k, (h, j, b'), \tilde{y}_j)\}_{j \in [\ell'], b' \in \{0,1\}}$. Computational indistinguishability of \mathcal{H}_1 and \mathcal{H}_2 follows by the security of the garbling scheme (Garble, Eval).

Notice that $C[\bar{pp}, i, b, m](\bar{vk})$ is identical to $\text{NC.SEnc}(\bar{pp}, (\bar{vk}, i, b), m^*)$. Thus, by the security of the non-compact OTSE-scheme NC, we can argue that \mathcal{A} 's advantage in \mathcal{H}_2 is negligible. \square

6 KDM-Secure Public-Key Encryption

In this section, we will build a KDM^{CPA} -secure public-key encryption scheme from a KDM^{CPA} -secure secret-key encryption scheme and a non-compact OTSE-scheme. The latter can be constructed from any public-key encryption scheme by the results of Sect. 4.

Let $\text{NC} = (\text{SSetup}, \text{SGen}, \text{SSign}, \text{SEnc}, \text{SDec})$ be a non-compact OTSE scheme, $\text{SKE} = (\text{Enc}, \text{Dec})$ be a KDM^{CPA} -secure secret-key encryption scheme and (Garble, Eval) be a garbling scheme. The public-key encryption scheme PKE is given as follows.

- $\text{KeyGen}(1^\kappa)$: Sample $k \xleftarrow{\$} \{0,1\}^\kappa$, compute $pp \leftarrow \text{NC.SSetup}(1^\kappa, \kappa)$, compute $(vk, \text{sgk}) \leftarrow \text{NC.SGen}(pp)$ and $\sigma \leftarrow \text{NC.SSign}(pp, \text{sgk}, k)$. Output $pk \leftarrow (pp, vk)$ and $sk \leftarrow (k, \sigma)$.
- $\text{Enc}(pk = (pp, vk), m)$: Let C be the following circuit: $C[m](k)$: Compute and output $\text{SKE.Enc}(k, m)$.²

² We also need to hardcode the randomness for SKE.Enc into C , but for ease of notation we omit this parameter.

$(\tilde{C}, e_C) \leftarrow \text{Garble}(1^\kappa, C[m])$
 Parse $e_C = \{K_{j,0}, K_{j,1}\}_{j \in [\kappa]}$
 $f_C \leftarrow \{\text{NC.SEnc}(\text{pp}, (\text{vk}, j, b), K_{j,b})\}_{j \in [\kappa], b \in \{0,1\}}$
 Output $\text{ct} \leftarrow (\tilde{C}, f_C)$.

– $\text{Dec}(\text{sk} = (k, \sigma), \text{ct} = (\tilde{C}, f_C))$:

$\tilde{k} \leftarrow \{\text{NC.SDec}(\text{pp}, (\text{vk}, \sigma, k), f_{C_{j,k_j}})\}_{j \in [\kappa]}$
 $c' \leftarrow \text{Eval}(\tilde{C}, \tilde{k})$
 $m \leftarrow \text{SKE.Dec}(k, c')$
 Output m

Note in particular that the secret key sk does not include the signing key sgk .

6.1 Correctness

We will first show that the scheme PKE is correct. Let therefore $(\text{pk}, \text{sk}) \leftarrow \text{PKE.KeyGen}(1^\kappa)$ and $\text{ct} \leftarrow \text{PKE.Enc}(\text{pk}, m)$. By the correctness of the OTSE-scheme NC it holds that $\tilde{k} = \{K_{j,k_j}\}$. Thus, by the correctness of the garbling scheme it holds that $\text{ct}' = \tilde{C}[m](k) = \text{SKE.Enc}(k, m)$. Finally, by the correctness of SKE it holds that $\text{SKE.Dec}(k, \text{ct}') = m$.

6.2 Security

We will now show that PKE is KDM^{CPA} -secure.

Theorem 7. *Assume that NC is an $\text{IND}^{\text{OTSIG}}$ -secure OTSE-scheme and $(\text{Garble}, \text{Eval})$ is a secure garbling scheme. Let \mathcal{F} be a class of KDM-functions and assume that the function $g_{\text{pp}, \text{sgk}} : x \mapsto (x, \text{NC.SSign}(\text{pp}, \text{sgk}, x))$ is in a class \mathcal{G} (e.g. affine functions). Assume that SKE is a KDM^{CPA} -secure secret-key encryption scheme for the class $\mathcal{F} \circ \mathcal{G}$. Then PKE is a KDM^{CPA} -secure public key encryption scheme for the class \mathcal{F} .*

Note that if both \mathcal{F} and \mathcal{G} are the class of affine functions, e.g. over \mathbb{F}_2 , then $\mathcal{F} \circ \mathcal{G}$ is again the class of affine functions (over \mathbb{F}_2). Thus, every function in $\mathcal{F} \circ \mathcal{G}$ can also be implemented as an affine function, i.e. by a matrix-vector product followed by an addition.

Proof. Let \mathcal{A} be a PPT-adversary against the KDM^{CPA} -security of PKE. Consider the following hybrids, in which we will change the way the KDM-oracle is implemented. For sake of readability, we only provide 3 hybrids, where in actuality each hybrid consists of q sub-hybrids, where q is the number of KDM-queries of \mathcal{A} .

Hybrid \mathcal{H}_1 : This hybrid is identical to the KDM^{CPA} -experiment.

Hybrid \mathcal{H}_2 : This hybrid is identical to \mathcal{H}_1 , except that f_C is computed by $f_C \leftarrow \{\text{NC.SEnc}(\text{pp}, (\text{vk}, j, b), K_{j,k_j})\}_{j \in [\kappa], b \in \{0,1\}}$, i.e. for each $j \in [\kappa]$ we encrypt K_{j,k_j}

twice, instead of $K_{j,0}$ and $K_{j,1}$. By the $\text{IND}^{\text{OTSIG}}$ -security of NC the hybrids \mathcal{H}_1 and \mathcal{H}_2 are computationally indistinguishable.

Hybrid \mathcal{H}_3 : This hybrid is identical to \mathcal{H}_2 , except that we compute \tilde{C} and f_C by $(\tilde{C}, \tilde{k}) \leftarrow \text{GCSim}(\mathcal{C}, \mathcal{C}[\mathbf{m}](k))$. Computational indistinguishability between \mathcal{H}_2 and \mathcal{H}_3 follows by the security of the garbling scheme (*Garble, Eval*). Notice that it holds that $\mathcal{C}[\mathbf{m}^*](k) = \text{SKE.Enc}(k, \mathbf{m}^*)$.

We will now show that the advantage of \mathcal{A} is negligible in \mathcal{H}_3 , due to the KDM^{CPA} -security of SKE. We will provide a reduction R such that $R^{\mathcal{A}}$ has the same advantage against the KDM^{CPA} -security of SKE as \mathcal{A} 's advantage against \mathcal{H}_3 .

Before we provide the reduction R , notice that R does not have access to its own challenge secret key k , which is part of the secret key $\text{sk} = (k, \sigma)$ of the resulting PKE. Also, since σ is a signature on k , R does not know the value of σ either. Thus, R cannot on its own simulate encryptions of messages that depend on (k, σ) . We overcome this problem by using the KDM-oracle provided to R which effectively allows R to obtain encryptions of key-dependent messages $\text{sk} = (k, \sigma)$. Details follow.

The reduction R first samples $\text{pp} \leftarrow \text{NC.SSetup}(1^\kappa, \kappa)$ and $(\text{vk}, \text{sgk}) \leftarrow \text{NC.SGen}(\text{pp})$ and invokes \mathcal{A} on $\text{pk} = (\text{pp}, \text{vk})$. Then R simulates \mathcal{H}_3 for \mathcal{A} with the following differences. Whenever \mathcal{A} queries the KDM-oracle with a function $f \in \mathcal{F}$, the reduction R programs a new function $f' \in \mathcal{F} \circ \mathcal{G}$ which is defined by

$$f'(k) = f(k, \text{NC.SSign}(\text{pp}, \text{sgk}, k)).$$

We assume for simplicity that the signing procedure NC.SSign is deterministic, if not we require that the same randomness r is used for NC.SSign at each KDM-query³.

We claim that R simulates \mathcal{H}_3 perfectly from the view of \mathcal{A} . If the challenge-bit in R 's KDM^{CPA} -experiment is 0, then the outputs of \mathcal{A} 's KDM-oracle on input f are encryptions of $f'(k) = f(\text{sk})$, and therefore, from the view of \mathcal{A} the challenge-bit in \mathcal{H}_3 is also 0. On the other hand, if the challenge-bit in R 's KDM^{CPA} -experiment is 1, then the outputs of \mathcal{A} 's KDM-oracle on input f are encryptions of 0^ℓ , and therefore, from \mathcal{A} 's view the challenge-bit in \mathcal{H}_3 is 1. We conclude that the advantage of $R^{\mathcal{A}}$ is identical to the advantage of \mathcal{A} against \mathcal{H}_3 . It follows from the KDM^{CPA} -security of SKE that the latter is negligible, which concludes the proof. \square

³ This does not pose a problem as we always sign the same message k at each KDM-query.

References

- [ABB10] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
- [ABSS93] Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The hardness of approximate optima in lattices, codes, and systems of linear equations. In: 34th FOCS, Palo Alto, California, 3–5 November 1993, pp. 724–733. IEEE Computer Society Press (1993)
- [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
- [Ajt98] Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In: 30th ACM STOC, Dallas, TX, USA, 23–26 May 1998, pp. 10–19. ACM Press (1998)
- [Ale03] Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS, Cambridge, MA, USA, 11–14 October 2003, pp. 298–307. IEEE Computer Society Press (2003)
- [AP12] Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_20
- [App14] Applebaum, B.: Key-dependent message security: generic amplification and completeness. *J. Cryptol.* **27**(3), 429–451 (2014)
- [BB04] Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_27
- [BF01] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
- [BFKL94] Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_24
- [BG10] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_1
- [BGH07] Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th FOCS, Providence, RI, USA, 20–23 October 2007, pp. 647–657. IEEE Computer Society Press (2007)
- [BHHI10] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_22

- [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
- [BHR12] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, Raleigh, NC, USA, 16–18 October 2012, pp. 784–796. ACM Press (2012)
- [BLSV17] Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous IBE, leakage resilience and circular security from new assumptions. Cryptology ePrint Archive, Report 2017/967 (2017). <http://eprint.iacr.org/2017/967>
- [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
- [CHK04] Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_13
- [CHKP10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
- [CHKP12] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. *J. Cryptol.* **25**(4), 601–639 (2012)
- [Coc01] Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45325-3_32
- [DG17a] Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 372–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_13
- [DG17b] Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
- [DMQN12] Döttling, N., Müller-Quade, J., Nascimento, A.C.A.: IND-CCA secure cryptography based on a variant of the LPN problem. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 485–503. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_30
- [Döt15] Döttling, N.: Low noise LPN: KDM secure public key encryption and sample amplification. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 604–626. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_27
- [GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, 17–20 May 2008, pp. 197–206. ACM Press (2008)
- [GR11] Guruswami, V., Rudra, A.: Soft decoding, dual BCH codes, and better list-decodable ε -biased codes. *IEEE Trans. Information Theory* **57**(2), 705–717 (2011)

- [HILL99] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
- [HR07] Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC, San Diego, CA, USA, 11–13 June 2007, pp. 469–477. ACM Press (2007)
- [Kho04] Khot, S.: Hardness of approximating the shortest vector problem in lattices. In: 45th FOCS, Rome, Italy, 17–19 October 2004, pp. 126–135. IEEE Computer Society Press (2004)
- [KMP14] Kiltz, E., Masny, D., Pietrzak, K.: Simple chosen-ciphertext security from low-noise LPN. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 1–18. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_1
- [KT17] Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. *Cryptology ePrint Archive, Report 2017/987* (2017). <http://eprint.iacr.org/2017/987>
- [LLL82] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**(4), 515–534 (1982)
- [LP09] Lindell, Y., Pinkas, B.: A proof of security of Yao’s protocol for two-party computation. *J. Cryptol.* **22**(2), 161–188 (2009)
- [LW10] Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
- [LW15] Lyubashevsky, V., Wichs, D.: Simple lattice trapdoor sampling from a broad class of distributions. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 716–730. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_32
- [Mic98] Micciancio, D.: The shortest vector in a lattice is hard to approximate to within some constant. In: 39th FOCS, Palo Alto, CA, USA, 8–11 November 1998, pp. 92–98. IEEE Computer Society Press (1998)
- [MP12] Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
- [MR04] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS, Rome, Italy, 17–19 October 2004, pp. 372–381. IEEE Computer Society Press (2004)
- [Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC, Baltimore, MA, USA, 22–24 May 2005, pp. 84–93. ACM Press (2005)
- [Sha84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
- [Wat05] Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7

- [Wat09] Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
- [Yao82] Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: 23rd FOCS, Chicago, Illinois, 3–5 November 1982, pp. 160–164. IEEE Computer Society Press (1982)
- [YZ16] Yu, Y., Zhang, J.: Cryptography with auxiliary input and trapdoor from constant-noise LPN. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 214–243. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_9



Key Dependent Message Security and Receiver Selective Opening Security for Identity-Based Encryption

Fuyuki Kitagawa^(✉) and Keisuke Tanaka

Tokyo Institute of Technology, Tokyo, Japan
{kitagaw1, keisuke}@is.titech.ac.jp

Abstract. We construct two identity-based encryption (IBE) schemes. The first one is IBE satisfying key dependent message (KDM) security for user secret keys. The second one is IBE satisfying simulation-based receiver selective opening (RSO) security. Both schemes are secure against adaptive-ID attacks and do not have any a-priori bound on the number of challenge identities queried by adversaries in the security games. They are the first constructions of IBE satisfying such levels of security.

Our constructions of IBE are very simple. We construct KDM secure IBE by transforming KDM secure secret-key encryption using IBE satisfying only ordinary indistinguishability against adaptive-ID attacks (IND-ID-CPA security). Our simulation-based RSO secure IBE is based only on IND-ID-CPA secure IBE.

We also demonstrate that our construction technique for KDM secure IBE is used to construct KDM secure public-key encryption. More precisely, we show how to construct KDM secure public-key encryption from KDM secure secret-key encryption and public-key encryption satisfying only ordinary indistinguishability against chosen plaintext attacks.

Keywords: Identity-based encryption
Key dependent message security · Receiver selective opening security

1 Introduction

1.1 Background

Identity-based encryption (IBE) proposed by Shamir [30] is an extension of public-key encryption (PKE). In IBE, we can use an identity of a recipient as a public-key. The secret-key corresponding to an identity is generated only by the trusted authority who has the master secret-key. Users can obtain secret-keys corresponding to their identities by authenticating themselves to the trusted authority. By using IBE, we can avoid distributing public-key certificates that is one of the major issues with public-key cryptography.

Security notions for IBE capture corruptions and collusions of users. In other words, we require that IBE guarantee confidentiality of a message encrypted under an identity id^* even if an adversary obtains a secret-key corresponding to any identity other than id^* .

Security notions for IBE are classified into two categories, that is, adaptive security and selective security. An IBE scheme is said to be secure against adaptive-ID attacks [12] if it is secure even when an adversary adaptively chooses the challenge identity id^* . On the other hand, an IBE scheme is said to be secure against selective-ID attacks [16] if it is secure when an adversary declares the challenge identity id^* before seeing public parameters.

Security against adaptive-ID attacks is a desirable security notion for IBE when we use it in practical situations. However, since IBE has an advanced functionality compared to PKE, attack scenarios that ordinary indistinguishability against adaptive-ID attacks does not capture can naturally occur in practical situations of IBE. As such attack scenarios, in this work, we focus on the situations of encrypting secret-keys and the selective opening attacks.

Black et al. [11] introduced the notion of *key dependent message (KDM) security* which guarantees confidentiality even in situations of encrypting secret-keys. Informally, an encryption scheme is said to be KDM secure if it is secure when an adversary can obtain encryptions of $f(\text{sk}_1, \dots, \text{sk}_\ell)$, where $\text{sk}_1, \dots, \text{sk}_\ell$ are secret-keys that exist in the system and f is a function.

Alperin-Sheriff and Peikert [3] pointed out that KDM security with respect to user secret-keys is well-motivated by some natural usage scenarios for IBE such as key distribution in a revocation system. They constructed the first IBE satisfying KDM security for user secret-keys assuming the hardness of the learning with errors (LWE) problem. Galindo et al. [22] proposed an IBE scheme that satisfies KDM security for master secret-keys based on the hardness of a rank problem on bilinear groups. However, both of these schemes are secure only against selective-ID attacks. Moreover, both schemes have some a-priori bound on the number of queries made by an adversary.¹

In the selective opening attack, an adversary, given some ciphertexts, adaptively corrupts some fraction of users and tries to break confidentiality of ciphertexts of uncorrupted users.

There are both sender corruption case and receiver corruption case in this attack scenario. Bellare et al. [8] formalized *sender selective opening (SSO) security* for PKE that captures situations where there are many senders and a single receiver, and an adversary can obtain messages and random coins of corrupted senders. Hazay et al. [24] later formalized *receiver selective opening (RSO) security* for PKE that captures situations where there are many receivers and a single sender, and an adversary can obtain messages and secret-keys of corrupted receivers.

¹ The scheme by Alperin-Sheriff and Peikert has an a-priori bound on the number of challenge identities in the security game. The scheme by Galindo et al. has an a-priori bound on the number of KDM encryption queries made by an adversary.

Selective opening attacks originally considered in the context of multi-party computation are natural and motivated in the context of IBE since it also considers situations where there are many users and some fraction are corrupted. Bellare et al. [9] defined SSO security for IBE and proposed SSO secure IBE schemes under the decisional linear assumption and a subgroup decision assumption in composite order bilinear groups. Their definition of SSO security for IBE captures adaptive-ID attacks in addition to sender selective opening attacks. However, it does not take receiver selective opening attacks into account.

It is known that the standard notions of indistinguishability imply neither KDM security [1, 10, 18, 28] nor selective opening security [7, 25, 26]. From this fact, we know very little about the possibility of IBE satisfying these stronger security notions than standard indistinguishability though there have been many works on the study of IBE.

Especially, it is open whether we can construct IBE that is KDM secure against adaptive-ID attacks and there is no a-priori bound on the number of queries made by an adversary. For selective opening security, we have no construction of IBE satisfying RSO security even if we require only security against selective-ID attacks.

As mentioned above, attack scenarios captured by both KDM security and selective opening security are natural and motivated for IBE. We thus think it is important to clarify these issues.

1.2 Our Results

Based on the above background, we propose KDM secure IBE and RSO secure IBE. Both schemes satisfy security against adaptive-ID attacks. They are the first schemes satisfying such levels of security.

Our constructions of IBE are very simple. We construct KDM secure IBE by transforming KDM secure secret-key encryption (SKE) using IBE satisfying ordinary indistinguishability against adaptive-ID attacks (IND-ID-CPA security) and garbled circuits. Somewhat surprisingly, our RSO secure IBE is based only on IND-ID-CPA secure IBE.

We show the details of each result below.

Key dependent message secure IBE. In this work, we focus on KDM security for user secret-keys similarly to Alperin-Sheriff and Peikert [3], and let KDM security indicate KDM security for user secret-keys. We show the following theorem.²

Theorem 1 (Informal). *Assuming there exist IND-ID-CPA secure IBE and SKE that is KDM secure with respect to projection functions (resp. functions computable by a-priori bounded size circuits). Then, there exists IBE that is KDM secure with respect to projection functions (resp. functions computable by a-priori bounded size circuits) against adaptive-ID attacks.*

² We also use garbled circuits, but it is implied by one-way functions [31]. Thus, it is not explicitly appeared in the statement of Theorem 1.

Projection function is a function each of whose output bits depends on at most one bit of an input. KDM security with respect to projection functions is a generalization of circular security [15]. We can construct IBE satisfying KDM security with respect to any function computable by circuits of a-priori bounded size [6] by requiring the same KDM security for the underlying SKE.

As noted above, KDM secure IBE proposed by Alperin-Sheriff and Peikert is only secure against selective-ID attacks. Moreover, their scheme has an a-priori bound on the number of challenge identities in the security game. Our KDM secure IBE is secure against adaptive-ID attacks and does not have any a-priori bound on the number of queries made by an adversary in the security game.

To achieve KDM security for an a-priori unbounded number of challenge identities, in our construction, the size of instances of the underlying KDM secure SKE needs to be independent of the number of users in the security game.³

We can construct SKE that is KDM secure with respect to projection functions and satisfies this efficiency requirement based on the decisional diffie-hellman (DDH) assumption [13] and LWE assumption [5].⁴ In addition, Applebaum [4] showed how to transform SKE that is KDM secure with respect to projection functions into SKE that is KDM secure with respect to functions computable by a-priori bounded size circuits.

We can construct IND-ID-CPA secure IBE under the LWE assumption [2]. Moreover, Döttling and Garg [21] recently showed how to construct IND-ID-CPA secure IBE based on the computational diffie-hellman (CDH) assumption.

Thus, from Theorem 1, we obtain the following corollary.

Corollary 1. *There exists IBE that is KDM secure with respect to functions computable by a-priori bounded size circuits against adaptive-ID attacks under the DDH assumption or LWE assumption.*

In addition to these results, based on the construction techniques above, we also show that we can transform KDM secure SKE into KDM secure PKE by using PKE satisfies ordinary indistinguishability against chosen plaintext attacks (IND-CPA security). Specifically, we show the following theorem.

Theorem 2 (Informal). *Assuming there exist IND-CPA secure PKE and SKE that is KDM secure with respect to projection functions (resp. functions computable by a-priori bounded size circuits). Then, there exists PKE that is KDM secure with respect to projection functions (resp. functions computable by a-priori bounded size circuits).*

It seems that we cannot construct KDM secure PKE from KDM secure SKE via the straightforward hybrid encryption methodology. It leads to a dead-lock of secret-keys of the underlying primitives and thus it is difficult to prove the security of hybrid encryption construction. Thus, this result is of independent interest.

³ For more details, see Remark 1 in Sect. 2.2.

⁴ More precisely, these works showed how to construct PKE that is KDM secure with respect to projection functions and satisfies the efficiency requirement.

Receiver selective opening secure IBE. Before our work, RSO security for IBE has never been studied while an IBE scheme that is SSO secure was proposed by Bellare et al. [9]. Therefore, we first define RSO security for IBE formally. Our definition is a natural extension of simulation-based RSO security for PKE proposed by Hazay et al. [24]. We then show the following theorem.

Theorem 3 (Informal). *Assuming there exists IND-ID-CPA secure IBE. Then, there exists IBE that satisfies simulation-based RSO security against adaptive-ID attacks.*

Somewhat surprisingly, the above theorem says that all we need is IND-ID-CPA secure IBE to achieve simulation-based RSO secure IBE. We can obtain the result via a simple double encryption paradigm [29].

The reason we can obtain the above result via a simple double encryption paradigm is that in receiver selective opening attacks for IBE, we have to consider the revelation of secret-keys themselves but not the random coins for key generation since secret-keys are generated by the trusted authority in IBE.

We also observe that if we allow only revelations of secret-keys and not the random coins for key generation, we can construct PKE satisfying such simulation-based RSO security using any PKE satisfying ordinary IND-CPA security. This fact is somewhat obvious from some previous results [17, 24] though these works did not explicitly state it. For self-containment, we show the following theorem.

Theorem 4 (Informal). *Assuming there exists IND-CPA secure PKE. Then, there exists PKE that satisfies simulation-based RSO security with respect to the revelation of only secret-keys.*

To prove simulation-based RSO security against the revelation of random coins for key generation, it seems that the underlying PKE needs to be *key simulatable* [19, 24] in some sense. In this case, it is difficult to construct simulation-based RSO secure PKE without relying on some specific algebraic or lattice assumptions.

We summarize our results in Fig. 1.

1.3 Overview of Our Techniques

We first give an intuition for our KDM secure IBE.

KDM secure IBE from KDM secure SKE. Our construction methodology for KDM secure IBE is somewhat based on the recent beautiful construction of IBE proposed by Döttling and Garg [20, 21] using new primitives called *chameleon encryption* or *one-time signatures with encryption*. The essence of their constructions is the mechanism that an encryptor who does not know the exact value of a public-key ek of PKE can generate an “encoding” of a PKE’s ciphertext under the public-key ek . Moreover, in their construction, the security of IBE is directly reduced to that of PKE in the last step of the security proof.

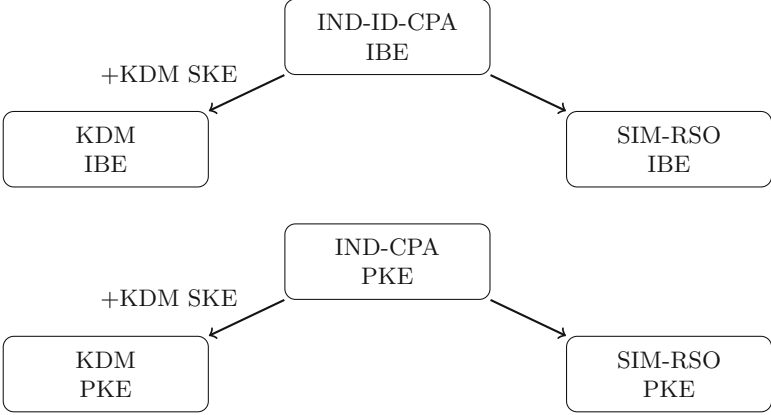


Fig. 1. Our results.

Based on this idea, by realizing the mechanism that an encryptor who does not know the value of the key K of SKE can generate an encoding of an SKE’s ciphertext under the key K of SKE, we try to transform SKE into public-key primitives such as PKE and IBE shifting the security level of SKE to them. We then show that we can construct KDM secure IBE (resp. PKE) based on KDM secure SKE and IND-ID-CPA secure IBE (resp. IND-CPA secure PKE).

We emphasize that we need neither chameleon encryption nor one-time signatures with encryption. IND-ID-CPA secure IBE is sufficient for our KDM secure IBE.

Our constructions are very simple and use garbled circuits. For simplicity, we focus on constructing KDM secure PKE to give an intuition. Suppose that we construct a KDM secure PKE scheme KdmPKE from a KDM secure SKE scheme SKE and IND-CPA secure PKE scheme PKE .

The encryption algorithm of KdmPKE first garbles an encryption circuit of SKE that has a message to be encrypted hardwired, that is, $E_{\text{ske}}(\cdot, m)$, and then encrypts labels of the garbled circuit by PKE under different keys. This process can be done without any secret-key of SKE and thus we achieve the “encoding” mechanism mentioned above. This construction is similar to that of “semi-adaptively” secure functional encryption based on selectively secure one proposed by Goyal et al. [23], but our techniques for the security proof explained below are different from theirs.

Why IND-CPA security of the underlying PKE is sufficient? One might wonder why IND-CPA security of the underlying PKE scheme PKE is sufficient to construct the KDM secure PKE scheme KdmPKE . To see the answer for this question, we closer look at the construction of KdmPKE .

Let the length of a secret-key K of SKE be len_K . A public-key Kdm.ek of KdmPKE consists of $2 \cdot \text{len}_K$ PKE ’s public-keys $\{\text{ek}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}$, where $[\text{len}_K]$ denotes $\{1, \dots, \text{len}_K\}$. The secret-key Kdm.dk corresponding to Kdm.ek consists

of a secret-key K of SKE and len_K secret-keys of PKE corresponding to the bit representation of $K = K[1] \dots K[\text{len}_K]$, that is, $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$. We note that secret-keys of PKE that do not correspond to the bit representation of K are not included in Kdm.dk .

As mentioned above, when encrypting a message m under the public-key $\text{Kdm.ek} := \{\text{ek}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}$, the encryption algorithm of KdmPKE first garbles an encryption circuit of SKE in which m is hardwired, that is, $E_{\text{ske}}(\cdot, m)$. This results in a single garbled circuit \tilde{E} and $2 \cdot \text{len}_K$ labels $\{\text{lab}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}$. Then, the encryption algorithm of KdmPKE encrypts $\text{lab}_{j,\alpha}$ by $\text{ek}_{j,\alpha}$ for every $j \in [\text{len}_K]$ and $\alpha \in \{0,1\}$. The resulting ciphertext of KdmPKE consists of \tilde{E} and these $2 \cdot \text{len}_K$ ciphertexts of PKE.

When decrypting this ciphertext with $\text{Kdm.dk} := (K, \{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]})$, we first obtain labels corresponding to K from len_K out of $2 \cdot \text{len}_K$ ciphertexts of PKE using $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$ and evaluate \tilde{E} with those labels. This results in an SKE's ciphertext $E_{\text{ske}}(K, m)$. Then, by decrypting it with K , we obtain m .

In this construction, secret-keys of PKE corresponding to K , that is, $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$ are included in Kdm.dk , but the rest of secret-keys $\{\text{dk}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$ are not included in Kdm.dk . Thus, even if an adversary for KdmPKE obtains encryptions of key dependent messages, they cannot get information of $\{\text{dk}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$ while they potentially get information of $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$ from those encryptions. In addition, in the security proof, we use the security of PKE of instances related to $\{\text{dk}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$, but not $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$. This is the reason the IND-CPA security of PKE is sufficient to construct a KDM secure PKE scheme KdmPKE . To see the fact, we show the outline of the proof below.

In the proof, by using the security of garbled circuits, we change the security game without affecting the behavior of an adversary so that we generate a challenge ciphertext under the key pair $(\text{Kdm.ek}, \text{Kdm.dk})$ with simulated garbled circuits computed from an SKE's ciphertext of the challenge key dependent message m^* under the key K , that is, $E_{\text{ske}}(K, m^*)$, where K is the secret-key of SKE contained in Kdm.dk . By this change, we do not need m^* itself, and the ciphertext $E_{\text{ske}}(K, m^*)$ is sufficient to simulate the security game. Thus, at this point, we can reduce the KDM security of KdmPKE to that of the underlying SKE.

More precisely, in the above proof, before using the security of garbled circuits, we have to eliminate the labels of garbled circuits that do not correspond to the bit representation of K , that is, $\{\text{lab}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$ from the view of the adversary. This can be done by using the IND-CPA security of PKE of only instances related to $\{\text{dk}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$. Therefore, we can complete the proof by using IND-CPA security of PKE of instances related to $\{\text{dk}_{j,1-K[j]}\}_{j \in [\text{len}_K]}$, but not $\{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]}$.

Conversions of functions. One additional non-trivial point is the conversion of functions by reductions.

In the security game of KDM security, an adversary queries a function and obtain an encryption of the function of secret-keys. Thus, KDM security is parameterized by function classes indicating functions that an adversary can query.

In the above construction, a secret-key Kdm.dk of KdmPKE contains some secret-keys of PKE in addition to a secret-key of SKE. Therefore, a function queried by an adversary for KdmPKE is a function of secret-keys of PKE and secret-keys of SKE. On the other hand, a function that a reduction algorithm can query is a function of only secret-keys of SKE. This means that the reduction algorithm needs to convert a function queried by an adversary for KdmPKE .

Such conversion is clearly possible if we do not care classes of functions. However, when considering KDM security, classes of functions are important since they determine the level of KDM security. It is not clear how such conversions affect a class of functions. Especially, it is not clear whether we can perform such conversions for functions without changing the class of functions.

We show that such conversions are possible for projection functions and functions computable by a-priori bounded size circuits. Thus, we can reduce the KDM security for those function classes of KdmPKE to that of SKE.

These arguments hold if we replace the underlying IND-CPA secure PKE with IND-ID-CPA secure IBE. The above construction can be seen as a special case where the size of instances of the underlying IBE linearly depends on the size of identity space. Thus, we can obtain KDM secure IBE from KDM secure SKE and IND-ID-CPA secure IBE.

RSO secure IBE from IND-ID-CPA secure IBE. Our starting point of the construction of RSO secure IBE is the above KDM secure IBE based on KDM secure SKE. It seems that the above construction can be used to carry over strong security notions of SKE to IBE that we need to simulate secret-keys in some sense in the security game. One such example, we focus on RSO security.⁵ Actually, in the above construction, if the underlying SKE has non-committing property (such as one-time pad), the resulting IBE seems to gain simulation-based RSO security.

However, it turns out that the construction is redundant and a simple double encryption paradigm [29] is sufficient to achieve RSO security. The reason we can construct RSO secure IBE via simple constructions is related to whether we allow the revelation of the random coins for key generation in addition to secret-keys or not.

Secret key vs random coins for the key generation. Hazay et al. [24] considered the revelation of both secret-keys and random coins for key generation when they defined RSO security for PKE. It is better to take the revelation of random

⁵ We observe that another example is leakage resilience. We do not focus on it in this paper.

coins of the key generation into account for many applications of PKE. However, for IBE, it is sufficient to take the revelation of only secret-keys into account.

In IBE, the trusted authority generates user secret-keys and distributes them to users. Thus, if an adversary corrupts a user, the adversary cannot obtain the random coin used to generate the secret-key of the user since the user does not know it. For this reason, we do not have to take the revelation of random coins of key generation in IBE into account.⁶

Construction based on a double encryption paradigm. When we do not take the revelation of random coins of key generation in IBE into account, we can construct simulation-based RSO secure IBE via a simple double encryption paradigm [29] without using garbled circuits.

More precisely, using an IBE scheme IBE whose identity space is $\mathcal{ID} \times \{0, 1\}$, we construct the following new IBE scheme RsolBE whose message space and identity space are $\{0, 1\}$ and \mathcal{ID} , respectively.

The setup algorithm of RsolBE is the same as that of IBE . When generating a secret-key $\text{Rso.sk}_{\text{id}}$ for identity $\text{id} \in \mathcal{ID}$, the key generation algorithm of RsolBE generates an IBE 's secret-key $\text{sk}_{\text{id},r}$ for the identity (id, r) , where r is a freshly generated random bit, and outputs $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id},r})$. When encrypting a message $m \in \{0, 1\}$ for identity $\text{id} \in \mathcal{ID}$, the encryption algorithm of RsolBE generates a pair of ciphertexts $(\text{CT}_0, \text{CT}_1)$, where CT_α is an encryption of m under the identity (id, α) for every $\alpha \in \{0, 1\}$. The decryption algorithm of RsolBE , given a pair of ciphertexts $(\text{CT}_0, \text{CT}_1)$ and a secret-key $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id},r})$, outputs the decryption result of CT_r with $\text{sk}_{\text{id},r}$.

This construction achieves a non-committing property. Suppose that we generate CT_r as an encryption of 0 under the identity (id, r) and CT_{1-r} as an encryption of 1 under the identity $(\text{id}, 1 - r)$ when generating a ciphertext $(\text{CT}_0, \text{CT}_1)$ for the identity id , where r is the random bit contained in the secret-key $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id},r})$ for id . We can open this ciphertext to any $m \in \{0, 1\}$ by pretending as if the secret-key $\text{Rso.sk}_{\text{id}}$ for id is $(r \oplus m, \text{sk}_{\text{id},r \oplus m})$. Due to this non-committing property, we prove the simulation-based RSO security of RsolBE .

From this result, we observe that if we take the revelation of only secret-keys into account, we can also construct SIM-RSO secure PKE based on any IND-CPA secure PKE. Our results on simulation-based RSO secure IBE and PKE highlight the gap of difficulties between achieving RSO security against revelation of only secret-keys and that against both secret-keys and random coins for key generation. To achieve the latter RSO security for PKE, it seems that the underlying scheme needs to be *key simulatable* [19, 24] in some sense.

1.4 Organization

In Sect. 2, we introduce some notations and review definitions of cryptographic primitives that we use as building blocks. In Sect. 3, we define IBE, and introduce

⁶ One additional reason is that we can always make a key generation algorithm of IBE deterministic by using pseudorandom functions.

KDM security and RSO security for it. In Sect. 4, we show how to construct KDM secure IBE from KDM secure SKE and IND-ID-CPA secure IBE. In Sect. 5, we show the construction of simulation-based RSO secure IBE based on IND-ID-CPA secure IBE. In Sect. 6, we show how to construct KDM secure PKE from KDM secure SKE and IND-CPA secure PKE. In Sect. 7, we show how to construct simulation-based RSO secure PKE based on IND-CPA secure PKE.

2 Preliminaries

We define some cryptographic primitives after introducing some notations.

Notations. $x \xleftarrow{r} X$ denotes choosing an element from a finite set X uniformly at random, and $y \leftarrow A(x; r)$ denotes assigning y to the output of an algorithm A on an input x and a randomness r . When there is no need to write the randomness clearly, we omit it and simply write $y \leftarrow A(x)$. For strings x and y , $x||y$ denotes the concatenation of x and y . For an integer ℓ , $[\ell]$ denote the set of integers $\{1, \dots, \ell\}$. For a string x and positive integer $j \leq |x|$, $x[j]$ denotes the j -th bit of x .

λ denotes a security parameter. PPT stands for probabilistic polynomial time. A function $f(\lambda)$ is a negligible function if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \text{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function.

2.1 Garbled Circuits

We define garbled circuits. We can realize garbled circuits for all efficiently computable circuits based on one-way functions [31].

Definition 1 (Garbled circuits). Let $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ be a family of circuits where each circuit in \mathcal{C}_n takes n -bit inputs. A circuit garbling scheme GC is a two tuple $(\text{Garble}, \text{Eval})$ of PPT algorithms.

The garbling algorithm Garble , given a security parameter 1^λ and circuit $C \in \mathcal{C}_n$, outputs a garbled circuit \tilde{C} , together with $2n$ labels $\{\text{lab}_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}$. The evaluation algorithm, given a garbled circuit \tilde{C} and n labels $\{\text{lab}_j\}_{j \in [n]}$, outputs y . As correctness, we require $\text{Eval}\left(\tilde{C}, \{\text{lab}_{j,x[j]}\}_{j \in [n]}\right) = C(x)$ for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, where $\left(\tilde{C}, \{\text{lab}_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}\right) \leftarrow \text{Garble}(1^\lambda, C)$.

We define its security. Let Sim be a PPT simulator. We define the following game between a challenger and an adversary \mathcal{A} .

1. First, the challenger chooses a bit $b \xleftarrow{r} \{0,1\}$ and sends a security parameter 1^λ to \mathcal{A} . Then, \mathcal{A} sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ for the challenger. Next, if $b = 1$, the challenger computes

$(\tilde{C}, \{\text{lab}_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, C)$ and returns $(\tilde{C}, \{\text{lab}_{j,x[j]}\}_{j \in [n]})$ to \mathcal{A} . Otherwise, the challenger returns $(\tilde{C}, \{\text{lab}_j\}_{j \in [n]}) \leftarrow \text{Sim}(1^\lambda, |C|, C(x))$ to \mathcal{A} .

2. \mathcal{A} outputs $b' \in \{0, 1\}$.

We require that there exists a PPT simulator Sim such that for any PPT adversary \mathcal{A} , we have $\text{Adv}_{\text{GC}, \mathcal{A}, \text{Sim}}^{\text{GC}}(\lambda) = \text{negl}(\lambda)$.

2.2 Public Key Encryption

A public-key encryption (PKE) scheme PKE is a three tuple $(\text{KG}, \text{Enc}, \text{Dec})$ of PPT algorithms. Below, let \mathcal{M} be the message space of PKE. The key generation algorithm KG , given a security parameter 1^λ , outputs a public key ek and a secret key dk . The encryption algorithm Enc , given a public key ek and message $m \in \mathcal{M}$, outputs a ciphertext CT . The decryption algorithm Dec , given a secret key dk and ciphertext c , outputs a message $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. As correctness, we require $\text{Dec}(\text{dk}, \text{Enc}(\text{ek}, m)) = m$ for every $m \in \mathcal{M}$ and $(\text{ek}, \text{dk}) \leftarrow \text{KG}(1^\lambda)$.

We introduce indistinguishability against chosen plaintext attacks (IND-CPA security) for PKE.

Definition 2 (IND-CPA security). Let PKE be a PKE scheme. We define the IND-CPA game between a challenger and an adversary \mathcal{A} as follows. We let \mathcal{M} be the message space of PKE.

1. First, the challenger chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Next, the challenger generates a key pair $(\text{ek}, \text{dk}) \leftarrow \text{KG}(1^\lambda)$ and sends ek to \mathcal{A} .
2. \mathcal{A} sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger. We require that $|m_0| = |m_1|$. The challenger computes $\text{CT} \leftarrow \text{Enc}(\text{ek}, m_b)$ and returns CT to \mathcal{A} .
3. \mathcal{A} outputs $b' \in \{0, 1\}$.

In this game, we define the advantage of the adversary \mathcal{A} as $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{indcpa}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$. We say that PKE is IND-CPA secure if for any PPT adversary \mathcal{A} , we have $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{indcpa}}(\lambda) = \text{negl}(\lambda)$.

Next, we define key dependent message (KDM) security for PKE [11].

Definition 3 (KDM-CPA security). Let PKE be a PKE scheme, \mathcal{F} function family, and ℓ the number of users. We define the \mathcal{F} -KDM-CPA game between a challenger and an adversary \mathcal{A} as follows. Let \mathcal{DK} and \mathcal{M} be the secret key space and message space of PKE, respectively.

1. First, the challenger chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Next, the challenger generates ℓ key pairs $(\text{ek}^{(k)}, \text{dk}^{(k)}) \leftarrow \text{KG}(1^\lambda)$ ($k \in [\ell]$). The challenger sets $\text{dk} := (\text{dk}^{(1)}, \dots, \text{dk}^{(\ell)})$ and sends $(\text{ek}^{(1)}, \dots, \text{ek}^{(\ell)})$ to \mathcal{A} .

2. \mathcal{A} may adaptively make polynomially many KDM queries.

KDM queries. \mathcal{A} sends $(k, f) \in [\ell] \times \mathcal{F}$ to the challenger. We require that f be a function such that $f : \mathcal{DK}^\ell \rightarrow \mathcal{M}$. If $b = 1$, the challenger returns $\text{CT} \leftarrow \text{Enc}(\text{ek}^{(k)}, f(\text{dk}))$ to \mathcal{A} . Otherwise, the challenger returns $\text{CT} \leftarrow \text{Enc}(\text{ek}^{(k)}, 0^{|f(\cdot)|})$ to \mathcal{A} .

3. \mathcal{A} outputs $b' \in \{0, 1\}$.

We say that PKE is \mathcal{F} -KDM-CPA secure if for any PPT adversary \mathcal{A} and polynomial $\ell = \ell(\lambda)$, we have $\text{Adv}_{\text{PKE}, \mathcal{F}, \mathcal{A}, \ell}^{\text{kdmcpa}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}| = \text{negl}(\lambda)$.

Remark 1 (Flexibility of the number of users). The above definition implicitly requires that the size of instances such as public keys, secret keys, and ciphertexts be independent of the number of users ℓ . We require the same condition for KDM secure SKE. This requirement is necessary for our constructions of KDM secure IBE (and PKE) based on KDM secure SKE.

When we reduce the KDM security of our IBE to that of the underlying SKE, the number of users ℓ in the security game of SKE corresponds to the number of challenge identities queried by an adversary for IBE. If the size of instances of SKE depends on ℓ , we can prove the KDM security of the resulting IBE only when the number of challenge identities is a-priori bounded.

Function families. As we can see, KDM security is defined with respect to function families. In this paper, we focus on KDM security with respect to the following function families.

Projection functions. A projection function is a function in which each output bit depends on at most a single bit of an input. Let f be a function and $y = y_1 \dots y_m$ be the output of the function f on an input $x = x_1 \dots x_n$, that is $f(x) = y$. We say that f is a projection function if for any $j \in [m]$, there exists $i \in [n]$ such that $y_j \in \{0, 1, x_i, 1 - x_i\}$.

In this paper, we let \mathcal{P} denote the family of projection functions, and we say that PKE is \mathcal{P} -KDM-CPA secure if it is KDM-CPA secure with respect to projection functions.

Functions computable by a-priori bounded size circuits. In the security game of KDM-CPA security with respect to this function family, an adversary can query a function computable by a circuit of a-priori bounded size and input and output length. We allow the size of instances of a scheme to depend on these a-priori bounds on functions while we do not allow it to depend on the number of total users as we noted in Remark 1.

In this paper, we say that PKE is \mathcal{B} -KDM-CPA secure if it is KDM-CPA secure with respect to functions computable by a-priori bounded size circuits.

\mathcal{P} -KDM-CPA security is a generalization of circular security [15] and strong enough for many applications. Boneh et al. [13] and Applebaum et al. [5] showed

how to construct \mathcal{P} -KDM-CPA secure PKE under the decisional diffie-hellman (DDH) assumption and learning with errors (LWE) assumption, respectively.⁷

Barak et al. [6] showed how to construct \mathcal{B} -KDM-CPA secure PKE under the DDH assumption or LWE assumption. Applebaum [4] showed how to transform \mathcal{P} -KDM-CPA secure PKE into \mathcal{B} -KDM-CPA secure one using garbled circuits.

We next introduce the definition of receiver selective opening (RSO) security for PKE. We adopt the simulation-based definition proposed by Hazay et al. [24].

Definition 4 (SIM-RSO security). *Let PKE be a PKE scheme, and ℓ the number of users. Let \mathcal{A} and \mathcal{S} be a PPT adversary and simulator, respectively. We define the following pair of games.*

Real game

1. First, the challenger generates ℓ key pairs $(\text{ek}^{(k)}, \text{dk}^{(k)}) \leftarrow \text{KG}(1^\lambda)$ ($k \in [\ell]$) and sends $(\text{ek}^{(1)}, \dots, \text{ek}^{(\ell)})$ to \mathcal{A} .
2. \mathcal{A} sends a message distribution Dist to the challenger. The challenger generates $\{m^{(k)}\}_{k \in [\ell]} \leftarrow \text{Dist}$, computes $\text{CT}^{(k)} \leftarrow \text{Enc}(\text{ek}^{(k)}, m^{(k)})$ for every $k \in [\ell]$, and sends $\{\text{CT}^{(k)}\}_{k \in [\ell]}$ to \mathcal{A} .
3. \mathcal{A} sends a subset \mathcal{I} of $[\ell]$ to the challenger. The challenger sends $\{(\text{dk}^{(k)}, m^{(k)})\}_{k \in \mathcal{I}}$ to \mathcal{A} .
4. \mathcal{A} sends a string **out** to the challenger.
5. The challenger outputs $\text{out}_{\text{real}} := (\{m^{(k)}\}_{k \in [\ell]}, \text{Dist}, \mathcal{I}, \text{out})$.

Simulated game

1. First, the challenger sends 1^λ to \mathcal{S} .
2. \mathcal{S} sends a message distribution Dist to the challenger. The challenger generates $\{m^{(k)}\}_{k \in [\ell]} \leftarrow \text{Dist}$.
3. \mathcal{S} sends a subset \mathcal{I} of $[\ell]$ to the challenger. The challenger sends $\{m^{(k)}\}_{k \in \mathcal{I}}$ to \mathcal{S} .
4. \mathcal{S} sends a string **out** to the challenger.
5. The challenger outputs $\text{out}_{\text{sim}} := (\{m^{(k)}\}_{k \in [\ell]}, \text{Dist}, \mathcal{I}, \text{out})$.

We say that PKE is SIM-RSO secure if for any PPT adversary \mathcal{A} and polynomial $\ell = \ell(\lambda)$, there exists a PPT simulator \mathcal{S} such that for any PPT distinguisher \mathcal{D} with binary output we have $\text{Adv}_{\text{PKE}, \mathcal{A}, \ell, \mathcal{S}, \mathcal{D}}^{\text{simrso}}(\lambda) = |\Pr[\mathcal{D}(\text{out}_{\text{real}}) = 1] - \Pr[\mathcal{D}(\text{out}_{\text{sim}}) = 1]| = \text{negl}(\lambda)$.

The above definition considers non-adaptive corruptions by an adversary. Namely, an adversary needs to corrupt users in one go.

⁷ Brakerski and Goldwasser [14] proposed \mathcal{P} -KDM-CPA secure PKE under the quadratic residuosity (QR) assumption and decisional composite residuosity (DCR) assumption, but their schemes do not satisfy the flexibility of the number of users in the sense of Remark 1.

We note that our construction of RSO secure PKE based on IND-CPA secure PKE works well even if we consider adaptive corruptions by an adversary. For simplicity, we define RSO security for PKE against non-adaptive corruptions in this paper.

Secret key vs key generation randomness. We define SIM-RSO security taking only the revelation of secret keys into account throughout the paper. Namely, we assume that an adversary gets only a secret key itself of a corrupted user and not the random coin used to generate the secret key.

Hazay et al. [24] considered the revelation of both secret keys and random coins for key generation when they defined RSO security for PKE. It is better to take the revelation of random coins of key generation into account for some applications.

We show that by requiring only security against the revelation of secret keys, we can obtain RSO secure PKE from IND-CPA secure PKE. If we consider RSO security against the revelation of random coins for key generation, it seems difficult to construct RSO secure PKE based only on IND-CPA secure PKE without assuming that secure erasure is possible or the underlying scheme is *key simulatable* [19, 24] in some sense.

2.3 Secret Key Encryption

A secret-key encryption (SKE) scheme SKE is a three tuple $(\text{KG}, \text{Enc}, \text{Dec})$ of PPT algorithms. Below, let \mathcal{M} be the message space of SKE . The key generation algorithm KG , given a security parameter 1^λ , outputs a secret key \mathbf{K} . The encryption algorithm Enc , given a secret key \mathbf{K} and a message $m \in \mathcal{M}$, outputs a ciphertext CT . The decryption algorithm Dec , given a secret key \mathbf{K} and a ciphertext CT , outputs a message $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. As correctness, We require $\text{Dec}(\mathbf{K}, \text{Enc}(\mathbf{K}, m)) = m$ for every $m \in \mathcal{M}$ and $\mathbf{K} \leftarrow \text{KG}(1^\lambda)$.

Next, we define KDM-CPA security for SKE.

Definition 5 (KDM-CPA security for SKE). *Let SKE be an SKE scheme whose key space and message space are \mathcal{K} and \mathcal{M} , respectively. Let \mathcal{F} be a function family, and ℓ the number of users. We define the \mathcal{F} -KDM-CPA game between a challenger and an adversary \mathcal{A} as follows.*

1. *First, the challenger chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Next, the challenger generates ℓ secret keys $\mathbf{K}^{(k)} \leftarrow \text{KG}(1^\lambda)(k \in [\ell])$, sets $\mathbf{K} := (\mathbf{K}^{(1)}, \dots, \mathbf{K}^{(\ell)})$, and sends 1^λ to \mathcal{A} .*
2. *\mathcal{A} may adaptively make polynomially many KDM queries.*

KDM queries. *\mathcal{A} sends $(k, f) \in [\ell] \times \mathcal{F}$ to the challenger. We require that f be a function such that $f : \mathcal{K}^\ell \rightarrow \mathcal{M}$. If $b = 1$, the challenger returns $\text{CT} \leftarrow \text{Enc}(\mathbf{K}^{(k)}, f(\mathbf{K}))$. Otherwise, the challenger returns $\text{CT} \leftarrow \text{Enc}(\mathbf{K}^{(k)}, 0^{|f(\cdot)|})$.*
3. *\mathcal{A} outputs $b' \in \{0, 1\}$.*

We say that SKE is \mathcal{F} -KDM-CPA secure if for any PPT adversary \mathcal{A} and polynomial $\ell = \ell(\lambda)$, we have $\text{Adv}_{\text{SKE}, \mathcal{F}, \mathcal{A}, \ell}^{\text{kdmcpa}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}| = \text{negl}(\lambda)$.

As we noted at Remark 1 after the definition of KDM security for PKE, we require that the size of instances of a KDM-CPA secure SKE scheme be independent of the number of users ℓ . This requirement is necessary for our construction of KDM secure IBE (and PKE) based on KDM secure SKE.

Similarly to KDM security for PKE, we focus on KDM security for SKE with respect to projection functions and that with respect to functions computable by a-priori bounded size circuits. We say that SKE is \mathcal{P} -KDM-CPA secure if it is KDM-CPA secure with respect to projection functions. We say that SKE is \mathcal{B} -KDM-CPA secure if it is KDM-CPA secure with respect to functions computable by a-priori bounded size circuits.

3 Identity-Based Encryption

We define identity-based encryption (IBE). Then, we introduce KDM security and RSO security for IBE.

An IBE scheme IBE is a four tuple (Setup, KG, Enc, Dec) of PPT algorithms. Below, let \mathcal{M} be the message space of IBE. The setup algorithm Setup, given a security parameter 1^λ , outputs a public parameter PP and a master secret key MSK. The key generation algorithm KG, given a master secret key MSK and identity $\text{id} \in \mathcal{ID}$, outputs a user secret key sk_{id} . The encryption algorithm Enc, given a public parameter PP, identity $\text{id} \in \mathcal{ID}$, and message $m \in \mathcal{M}$, outputs a ciphertext CT. The decryption algorithm Dec, given a user secret key sk_{id} and ciphertext CT, outputs a message $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. As correctness, we require $\text{Dec}(\text{KG}(\text{MSK}, \text{id}), \text{Enc}(\text{PP}, \text{id}, m)) = m$ for every $m \in \mathcal{M}$, $\text{id} \in \mathcal{ID}$, and $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$.

We define indistinguishability against adaptive-ID attacks (IND-ID-CPA security [12]) for IBE.

Definition 6 (IND-ID-CPA security for IBE). *Let IBE be an IBE scheme whose identity space and message space are \mathcal{ID} and \mathcal{M} , respectively. We define the IND-ID-CPA game between a challenger and an adversary \mathcal{A} as follows.*

1. *First, the challenger chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Next, the challenger generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . Finally, the challenger prepares a list L_{ext} which is initially empty.*

At any step of the game, \mathcal{A} can make key extraction queries.

Extraction queries. *\mathcal{A} sends $\text{id} \in \mathcal{ID}$ to the challenger. The challenger returns $\text{sk}_{\text{id}} \leftarrow \text{KG}(\text{MSK}, \text{id})$ to \mathcal{A} and adds id to L_{ext} .*

2. *\mathcal{A} sends $(\text{id}^*, m_0, m_1) \in \mathcal{ID} \times \mathcal{M} \times \mathcal{M}$ to the challenger. We require that $|m_0| = |m_1|$ and $\text{id}^* \notin L_{\text{ext}}$. The challenger computes $\text{CT} \leftarrow \text{Enc}(\text{PP}, \text{id}, m_b)$ and returns CT to \mathcal{A} .*

Below, \mathcal{A} is not allowed to make an extraction query for id^ .*

3. *\mathcal{A} outputs $b' \in \{0, 1\}$.*

We say that IBE is IND-ID-CPA secure if for any PPT adversary \mathcal{A} , we have $\text{Adv}_{\text{IBE}, \mathcal{A}}^{\text{indidcpa}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}| = \text{negl}(\lambda)$.

3.1 KDM Security for IBE

Next, we define KDM security for IBE. Alperin-Sheriff and Peikert [3] defined KDM security for IBE by extending selective security for IBE [16]. The following definition is an extension of adaptive security for IBE [12]. For the difference between the definition of Alperin-Sheriff and Peikert and ours, see Remark 2 after Definition 7.

Definition 7 (KDM-CPA security for IBE). Let IBE be an IBE scheme, and \mathcal{F} a function family. We define the \mathcal{F} -KDM-CPA game between a challenger and an adversary \mathcal{A} as follows. Let SK, \mathcal{ID} , and \mathcal{M} be the user secret key space, identity space, and message space of IBE, respectively.

1. First, the challenger chooses a challenge bit $b \leftarrow \{0, 1\}$. Next, the challenger generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . Finally, the challenger prepares lists $L_{\text{ext}}, L_{\text{ch}}$, and \mathbf{sk} all of which are initially empty.
2. \mathcal{A} may adaptively make the following three types of queries.

Extraction queries. \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$ to the challenger. The challenger returns $\text{sk}_{\text{id}} \leftarrow \text{KG}(\text{MSK}, \text{id})$ to \mathcal{A} and adds id to L_{ext} .

Registration queries. \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$ to the challenger. The challenger generates $\text{sk}_{\text{id}} \leftarrow \text{KG}(\text{MSK}, \text{id})$ and adds id and sk_{id} to L_{ch} and \mathbf{sk} , respectively.

KDM queries. \mathcal{A} sends $(\text{id}, f) \in L_{\text{ch}} \times \mathcal{F}$ to the challenger. We require that f be a function such that $f : SK^{|L_{\text{ch}}|} \rightarrow \mathcal{M}$. If $b = 1$, the challenger returns $\text{CT} \leftarrow \text{Enc}(\text{PP}, \text{id}, f(\mathbf{sk}))$ to \mathcal{A} . Otherwise, the challenger returns $\text{CT} \leftarrow \text{Enc}(\text{PP}, \text{id}, 0^{|\mathcal{M}|})$ to \mathcal{A} .

3. \mathcal{A} outputs $b' \in \{0, 1\}$.

We say that IBE is \mathcal{F} -KDM-CPA secure if for any PPT adversary \mathcal{A} , we have $\text{Adv}_{\text{IBE}, \mathcal{F}, \mathcal{A}}^{\text{kdmcpa}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}| = \text{negl}(\lambda)$.

Similarly to KDM security for PKE, we focus on KDM security for IBE with respect to projection functions and that with respect to functions computable by a-priori bounded size circuits. We say that IBE is \mathcal{P} -KDM-CPA secure if it is KDM-CPA secure with respect to projection functions. We say that IBE is \mathcal{B} -KDM-CPA secure if it is KDM-CPA secure with respect to functions computable by a-priori bounded size circuits.

Remark 2 (Difference with [3]). Alperin-Sheriff and Peikert [3] defined KDM security for IBE. Their definition is a natural extension of selective security for IBE [16]. In their definition, an adversary must declare the set of challenge identities L_{ch} at the beginning of the security game. On the other hand, our definition

of KDM security for IBE is an extension of adaptive security for IBE [12]. In our definition, an adversary can adaptively declare challenge identities through registration queries.⁸

One additional difference between our definition and that of Alperin-Sheriff and Peikert is whether the size of instances of IBE such as a public parameter is allowed to depend on the number of challenge identities or not. In the definition of Alperin-Sheriff and Peikert, the setup algorithm of IBE takes the upper bound on the number of challenge identities as an input, and the size of instances of IBE depends on the number of challenge identities. In our definition, there is no a-priori bound on the number of challenge identities, and thus the size of instances of IBE is required to be independent of the number of challenge identities.

3.2 RSO Security for IBE

We next define RSO security for IBE. We extend the simulation-based definition for PKE proposed by Hazay et al. [24].

Definition 8 (SIM-RSO security for IBE). *Let IBE be an IBE scheme whose identity space and message space are \mathcal{ID} and \mathcal{M} , respectively. Let \mathcal{A} and \mathcal{S} be a PPT adversary and simulator, respectively. We define the following pair of games.*

Real game

1. *The challenger generates public parameter and master secret key $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . The challenger then prepares a list L_{ext} which is initially empty.*

At any step of the game, \mathcal{A} can make key extraction queries.

Extraction queries. *\mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus L_{\text{ext}}$ to the challenger. The challenger returns $\text{sk}_{\text{id}} \leftarrow \text{KG}(\text{MSK}, \text{id})$ to \mathcal{A} and adds id to L_{ext} .*

2. *\mathcal{A} sends q identities $\{\text{id}^{(k)} \in \mathcal{ID} \setminus L_{\text{ext}}\}_{k \in [q]}$ and a message distribution*

Dist on \mathcal{M}^q to the challenger, where q is an a-priori unbounded polynomial of λ . The challenger generates $\{m^{(k)}\}_{k \in [q]} \leftarrow \text{Dist}$, computes $\text{CT}^{(k)} \leftarrow$

$\text{Enc}(\text{PP}, \text{id}^{(k)}, m^{(k)})$ for every $k \in [q]$, and sends $\{\text{CT}^{(k)}\}_{k \in [q]}$ to \mathcal{A} .

Below, \mathcal{A} is not allowed to make extraction queries for $\{\text{id}^{(k)}\}_{k \in [q]}$.

3. *\mathcal{A} sends a subset \mathcal{I} of $[q]$ to the challenger. The challenger computes $\text{sk}_{\text{id}^{(k)}} \leftarrow \text{KG}(\text{MSK}, \text{id}^{(k)})$ for every $k \in \mathcal{I}$ and sends $\{(\text{sk}_{\text{id}^{(k)}}, m^{(k)})\}_{k \in \mathcal{I}}$ to \mathcal{A} .*

⁸ One might think it is a restriction to force an adversary to register challenge identities before making KDM queries. This is not the case since the adversary is allowed to adaptively make registration and KDM queries. Our definition with registration queries makes the security proof of our IBE simple.

4. \mathcal{A} sends a string out to the challenger.

5. The challenger outputs $\text{out}_{\text{real}} = \left(\left\{ \text{id}^{(k)} \right\}_{k \in [q]}, \left\{ m^{(k)} \right\}_{k \in [q]}, \text{Dist}, \mathcal{I}, \text{out} \right)$.

Simulated game

1. First, the challenger sends 1^λ to \mathcal{S} .

2. \mathcal{S} sends q identities $\left\{ \text{id}^{(k)} \in \mathcal{ID} \right\}_{k \in [q]}$ and a message distribution Dist on \mathcal{M}^q to the challenger, where q is an a-priori unbounded polynomial of λ . The challenger generates $\left\{ m^{(k)} \right\}_{k \in [q]} \leftarrow \text{Dist}$.

3. \mathcal{S} sends a subset \mathcal{I} of $[q]$ to the challenger. The challenger sends $\left\{ m^{(k)} \right\}_{k \in \mathcal{I}}$ to \mathcal{S} .

4. \mathcal{S} sends a string out to the challenger.

5. The challenger outputs $\text{out}_{\text{sim}} := \left(\left\{ \text{id}^{(k)} \right\}_{k \in [q]}, \left\{ m^{(k)} \right\}_{k \in [q]}, \text{Dist}, \mathcal{I}, \text{out} \right)$.

Then, we say that IBE is SIM-RSO secure if for any PPT adversary \mathcal{A} , there exists a PPT simulator \mathcal{S} such that for any PPT distinguisher \mathcal{D} with binary output we have $\text{Adv}_{\text{IBE}, \mathcal{A}, \mathcal{S}, \mathcal{D}}^{\text{simrso}}(\lambda) = |\Pr[\mathcal{D}(\text{out}_{\text{real}}) = 1] - \Pr[\mathcal{D}(\text{out}_{\text{sim}}) = 1]| = \text{negl}(\lambda)$.

As we noted after defining SIM-RSO security for PKE, for simplicity, we consider non-adaptive corruptions by an adversary in this paper. We note that our construction of RSO secure IBE based on IND-ID-CPA secure IBE works well if we consider adaptive corruptions by an adversary.

Remark 3 (On the syntax of simulators). In the above definition, not only an adversary but also a simulator is required to output challenge identities with a message distribution, and these identities are given to a distinguisher of games. One might think this is somewhat strange since these identities output by a simulator are never used in the simulated game. This syntax of simulators is similar to that used by Bellare et al. [9] when they defined simulation-based sender selective opening security for IBE.

It does not seem to be a big issue whether we require a simulator to output identities or not. This intuition comes from the fact that we allow an adversary and simulator to output arbitrary length strings, and thus they can always include challenge identities into the output strings.

However, this subtle issue might divide notions of selective opening security for IBE. Especially, it looks hard to prove that the definition with simulators without outputting identities implies that with simulators outputting identities, while it is easy to prove the opposite implication. This means that the former definition is possibly weaker than the latter.

From these facts, similarly to Bellare et al. [9], we adopt the definition with simulators explicitly outputting identities in this work.

4 KDM Secure IBE from KDM Secure SKE and IND-ID-CPA Secure IBE

We show how to construct KDM secure IBE based on KDM secure SKE and IND-ID-CPA secure IBE. The construction also uses a circuit garbling scheme.

Let $\text{SKE} = (\text{G}, \text{E}, \text{D})$ be an SKE scheme whose message space is \mathcal{M} . Let len_K and len_r denote the length of a secret key and encryption randomness of SKE, respectively. Let $\text{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ be an IBE scheme whose identity space is $\mathcal{ID} \times \{0, 1\}^{\text{len}_K} \times \{0, 1\}$. Let $\text{GC} = (\text{Garble}, \text{Eval})$ be a garbling scheme. Using SKE, IBE, and GC, we construct the following IBE scheme $\text{KdmIBE} = (\text{Kdm.Setup}, \text{Kdm.KG}, \text{Kdm.Enc}, \text{Kdm.Dec})$ whose message space and identity space are \mathcal{M} and \mathcal{ID} , respectively.

$\text{Kdm.Setup}(1^\lambda)$:

- Return $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$.

$\text{Kdm.KG}(\text{MSK}, \text{id})$:

- Generate $\text{K}_{\text{id}} \leftarrow \text{G}(1^\lambda)$.
- Generate $\text{sk}_{\text{id}, j, \text{K}_{\text{id}}[j]} \leftarrow \text{KG}(\text{MSK}, (\text{id}, j, \text{K}_{\text{id}}[j]))$ for every $j \in [\text{len}_K]$.
- Return $\text{Kdm.sk}_{\text{id}} := \left(\text{K}_{\text{id}}, \{ \text{sk}_{\text{id}, j, \text{K}_{\text{id}}[j]} \}_{j \in [\text{len}_K]} \right)$.

$\text{Kdm.Enc}(\text{PP}, \text{id}, m)$:

- Generate $r_E \xleftarrow{r} \{0, 1\}^{\text{len}_r}$ and compute $\left(\tilde{\text{E}}, \{ \text{lab}_{j, \alpha} \}_{j \in [\text{len}_K], \alpha \in \{0, 1\}} \right) \leftarrow \text{Garble}(1^\lambda, \text{E}(\cdot, m; r_E))$, where $\text{E}(\cdot, m; r_E)$ is the encryption circuit E of SKE into which m and r_E are hardwired.
- Compute $\text{CT}_{j, \alpha} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, \alpha), \text{lab}_{j, \alpha})$ for every $j \in [\text{len}_K]$ and $\alpha \in \{0, 1\}$.
- Return $\text{Kdm.CT} := \left(\tilde{\text{E}}, \{ \text{CT}_{j, \alpha} \}_{j \in [\text{len}_K], \alpha \in \{0, 1\}} \right)$.

$\text{Kdm.Dec}(\text{Kdm.sk}_{\text{id}}, \text{Kdm.CT})$:

- Parse $\left(\text{K}_{\text{id}}, \{ \text{sk}_{\text{id}, j} \}_{j \in [\text{len}_K]} \right) \leftarrow \text{Kdm.sk}_{\text{id}}$.
- Parse $\left(\tilde{\text{E}}, \{ \text{CT}_{j, \alpha} \}_{j \in [\text{len}_K], \alpha \in \{0, 1\}} \right) \leftarrow \text{Kdm.CT}$.
- For every $j \in [\text{len}_K]$, compute $\text{lab}_j \leftarrow \text{Dec}(\text{sk}_{\text{id}, j}, \text{CT}_{j, \text{K}_{\text{id}}[j]})$.
- Compute $\text{CT}_{\text{ske}} \leftarrow \text{Eval}(\tilde{\text{E}}, \{ \text{lab}_j \}_{j \in [\text{len}_K]})$.
- Return $m \leftarrow \text{D}(\text{K}_{\text{id}}, \text{CT}_{\text{ske}})$.

Correctness. When decrypting a ciphertext of KdmIBE that encrypts a message m , we first obtain a ciphertext of SKE that encrypts m from the correctness of IBE and GC. The correctness of KdmIBE then follows from that of SKE.

We prove the following theorem.

Theorem 5. *Let SKE be an SKE scheme that is \mathcal{P} -KDM-CPA secure (resp. \mathcal{B} -KDM-CPA secure). Let IBE be an IND-ID-CPA secure IBE scheme and GC a secure garbling scheme. Then, KdmIBE is an IBE scheme that is \mathcal{P} -KDM-CPA secure (resp. \mathcal{B} -KDM-CPA secure).*

Proof of Theorem 5. Let \mathcal{A} be an adversary that attacks the \mathcal{P} -KDM-CPA security of KdmIBE and makes at most q_{ch} registration queries and q_{kdm} KDM queries. We proceed the proof via a sequence of games. For every $t \in \{0, \dots, 2\}$, let SUC_t be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game t .

Game 0: This is the original \mathcal{P} -KDM-CPA game regarding KdmIBE . Then, we have $\text{Adv}_{\text{KdmIBE}, \mathcal{P}, \mathcal{A}}^{\text{kdmcpa}} = |\Pr[\text{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$, generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$, and sends PP to \mathcal{A} . The challenger also prepares lists $L_{\text{ext}}, L_{\text{ch}}$, and sk_{kdm} all of which are initially empty.
2. \mathcal{A} may adaptively make the following three types of queries.

Extraction queries. \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$ to the challenger.

The challenger responds as follows.

- The challenger generates $\text{K}_{\text{id}} \leftarrow \mathcal{G}(1^\lambda)$.
- The challenger generates $\text{sk}_{\text{id}, j, \text{K}_{\text{id}}[j]} \leftarrow \text{KG}(\text{MSK}, (\text{id}, j, \text{K}_{\text{id}}[j]))$ for every $j \in [\text{len}_{\mathcal{K}}]$.
- The challenger returns $\text{Kdm.sk}_{\text{id}} := \left(\text{K}_{\text{id}}, \{\text{sk}_{\text{id}, j, \text{K}_{\text{id}}[j]}\}_{j \in [\text{len}_{\mathcal{K}}]} \right)$ to \mathcal{A} and adds id to L_{ext} .

Registration queries. \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$ to the challenger.

The challenger generates $\text{Kdm.sk}_{\text{id}}$ in the same way as the answer to an extraction query. The challenger then adds id to L_{ch} and $\text{Kdm.sk}_{\text{id}}$ to sk_{kdm} .

KDM queries. \mathcal{A} sends $(\text{id}, f) \in L_{\text{ch}} \times \mathcal{P}$ to the challenger. The challenger responds as follows.

- (a) The challenger sets $m_1 := f(\text{sk}_{\text{kdm}})$ and $m_0 := 0^{|m_1|}$.
- (b) The challenger computes $\left(\tilde{\text{E}}, \{\text{lab}_{j, \alpha}\}_{j \in [\text{len}_{\mathcal{K}}], \alpha \in \{0, 1\}} \right) \leftarrow \text{Garble}(1^\lambda, \text{E}(\cdot, m_b; r_{\text{E}}))$, where $r_{\text{E}} \xleftarrow{r} \{0, 1\}^{\text{len}_r}$.
- (c) For every $j \in [\text{len}_{\mathcal{K}}]$ and $\alpha \in \{0, 1\}$, the challenger computes $\text{CT}_{j, \alpha} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, \alpha), \text{lab}_{j, \alpha})$.
- (d) The challenger returns $\text{Kdm.CT} := \left(\tilde{\text{E}}, \{\text{CT}_{j, \alpha}\}_{j \in [\text{len}_{\mathcal{K}}], \alpha \in \{0, 1\}} \right)$.

3. \mathcal{A} outputs $b' \in \{0, 1\}$.

Game 1: Same as Game 0 except the following. When \mathcal{A} makes a KDM query $(\text{id}, f) \in L_{\text{ch}} \times \mathcal{P}$, for every $j \in [\text{len}_{\mathcal{K}}]$ the challenger computes $\text{CT}_{j, 1 - \text{K}_{\text{id}}[j]} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, 1 - \text{K}_{\text{id}}[j]), \text{lab}_{j, \text{K}_{\text{id}}[j]})$, where K_{id} is the secret key of SKE generated when id was registered to L_{ch} . Recall that in Game 0, $\text{CT}_{j, 1 - \text{K}_{\text{id}}[j]}$ is generated as $\text{CT}_{j, 1 - \text{K}_{\text{id}}[j]} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, 1 - \text{K}_{\text{id}}[j]), \text{lab}_{j, 1 - \text{K}_{\text{id}}[j]})$. Namely, we eliminate labels of garbled circuits that do not correspond to K_{id} from the view of \mathcal{A} in this game.

In order to simulate both Game 0 and 1, we do not need user secret keys of IBE that do not correspond to $\{\text{K}_{\text{id}}\}_{\text{id} \in L_{\text{ch}}}$, that is $\{\text{sk}_{\text{id}, j, 1 - \text{K}_{\text{id}}[j]}\}_{\text{id} \in L_{\text{ch}}, j \in [\text{len}_{\mathcal{K}}]}$ while we need $\{\text{sk}_{\text{id}, j, \text{K}_{\text{id}}[j]}\}_{\text{id} \in L_{\text{ch}}, j \in [\text{len}_{\mathcal{K}}]}$ to compute the value of $f(\text{sk}_{\text{kdm}})$ when \mathcal{A} makes a KDM query. Therefore, we can use the IND-ID-CPA security of IBE when the challenge identity is $(\text{id}, j, 1 - \text{K}_{\text{id}}[j])$ for every $\text{id} \in L_{\text{ch}}$ and

$j \in [\text{len}_\kappa]$. By using IND-ID-CPA security of IBE $\text{len}_\kappa \cdot q_{\text{kdm}}$ times, we can prove $|\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| = \text{negl}(\lambda)$.

Game 2: Same as Game 1 except that to respond to a KDM query from \mathcal{A} , the challenger generates a garbled circuit using the simulator for GC. More precisely, when \mathcal{A} makes a KDM query $(\text{id}, f) \in L_{\text{ch}} \times \mathcal{P}$, the challenger generates $r_E \xleftarrow{r} \{0, 1\}^{\text{len}_r}$ and $\text{CT}_{\text{ske}} \leftarrow E(\text{K}_{\text{id}}, m_b; r_E)$, and computes $(\tilde{E}, \{\text{lab}_j\}_{j \in [\text{len}_\kappa]}) \leftarrow \text{Sim}(1^\lambda, |E|, \text{CT}_{\text{ske}})$, where Sim is the simulator for GC and $|E|$ denotes the size of the encryption circuit E of SKE. Moreover, the challenger computes $\text{CT}_{j,\alpha} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, \alpha), \text{lab}_j)$ for every $j \in [\text{len}_\kappa]$ and $\alpha \in \{0, 1\}$.

In the last step, we eliminate labels of garbled circuits that do not correspond to $\{\text{K}_{\text{id}}\}_{\text{id} \in L_{\text{ch}}}$. Therefore, by using the security of GC q_{kdm} times, we can show that $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| = \text{negl}(\lambda)$.

Below, we show that $|\Pr[\text{SUC}_2] - \frac{1}{2}| = \text{negl}(\lambda)$ holds by the \mathcal{P} -KDM-CPA security of SKE. Using the adversary \mathcal{A} , we construct an adversary \mathcal{A}_{ske} that attacks the \mathcal{P} -KDM-CPA security of SKE when the number of keys is q_{ch} .

Before describing \mathcal{A}_{ske} , we note on the conversion of projection functions. We let $\text{K}^{(k)}$ be the secret key of SKE generated to respond to the k -th registration query $\text{id}^{(k)}$ made by \mathcal{A} . We let $\alpha_{k,j}$ denote the j -th bit of $\text{K}^{(k)}$, that is, $\text{K}^{(k)}[j]$ for every $j \in [\text{len}_\kappa]$ and $k \in [q_{\text{ch}}]$. Let f be a projection function that \mathcal{A} queries as a KDM query. f is a projection function of $\{\text{K}^{(k)}\}_{k \in [q_{\text{ch}}]}$ and $\{\text{sk}_{\text{id}^{(k)},j,\alpha_{k,j}}\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa]}$. To attack the \mathcal{P} -KDM-CPA security of SKE, \mathcal{A}_{ske} needs to compute a projection function g such that

$$g\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}\right) = f\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sk}_{\text{id}^{(k)},j,\alpha_{k,j}}\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa]}\right). \quad (1)$$

We can compute such a function g from f and $\{\text{sk}_{\text{id}^{(k)},j,\alpha}\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa], \alpha \in \{0,1\}}$ as follows.

We first observe that for every $k \in [q_{\text{ch}}]$ and $j \in [\text{len}_\kappa]$, we can write

$$\begin{aligned} \text{sk}_{\text{id}^{(k)},j,\alpha_{k,j}} &= (1 - \alpha_{k,j}) \cdot \text{sk}_{\text{id}^{(k)},j,0} \oplus \alpha_{k,j} \cdot \text{sk}_{\text{id}^{(k)},j,1} \\ &= \alpha_{k,j} \cdot (\text{sk}_{\text{id}^{(k)},j,1} \oplus \text{sk}_{\text{id}^{(k)},j,0}) \oplus \text{sk}_{\text{id}^{(k)},j,0}. \end{aligned}$$

We suppose that $\text{sk}_{\text{id}^{(k)},j,1}$ and $\text{sk}_{\text{id}^{(k)},j,0}$ are represented as binary strings and \oplus is done in the bit-wise manner. We define a function $\text{sel}_{k,j}$ as $\text{sel}_{k,j}(\gamma \in \{0,1\}) = \gamma \cdot (\text{sk}_{\text{id}^{(k)},j,1} \oplus \text{sk}_{\text{id}^{(k)},j,0}) \oplus \text{sk}_{\text{id}^{(k)},j,0}$. Then, we have

$$\begin{aligned} f\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sk}_{\text{id}^{(k)},j,\alpha_{k,j}}\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa]}\right) \\ = f\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sel}_{k,j}(\alpha_{k,j})\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa]}\right). \end{aligned}$$

We define $g\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}\right) = f\left(\left\{\text{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sel}_{k,j}(\text{K}^{(k)}[j])\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_\kappa]}\right)$. Then, g satisfies Eq. 1.

We show that if f is a projection function, then so is g . Let γ be an output bit of $g\left(\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}\right) = f\left(\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sel}_{k,j}\left(\mathbf{K}^{(k)}[j]\right)\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}]}\right)$. We say that γ is a projective bit for f (resp. g) if it depends on a single bit of an input for f (resp. g). We also say that γ is a constant bit for f (resp. g) if it does not depend on any bit of an input for f (resp. g).

Since f is a projection function, γ is a constant bit or projective bit for f that depends on either part of $\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}$ or $\left\{\text{sel}_{k,j}\left(\mathbf{K}^{(k)}[j]\right)\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}]}$. Thus, we consider the following three cases. (i) If γ is a constant bit for f , γ is clearly a constant bit for g . (ii) If γ is a projective bit for f and depends on a single bit of $\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}$, γ is a projective bit for g since $\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}$ is also an input for g . (iii) If γ is a projective bit for f and depends on some bit of $\left\{\text{sel}_{k,j}\left(\mathbf{K}^{(k)}[j]\right)\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}]}$, γ is a projective bit for g since each bit of $\left\{\text{sel}_{k,j}\left(\mathbf{K}^{(k)}[j]\right)\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}]}$ depends on a bit $\mathbf{K}^{(k)}[j]$ for some $k \in [q_{\text{ch}}]$ and $j \in [\text{len}_{\mathbf{K}}]$, and $\mathbf{K}^{(k)}[j]$ is a part of an input to g . Therefore, γ is a projective bit or constant bit for g in any case, and thus g is a projection function.

We now describe the adversary \mathcal{A}_{ske} that uses the above conversion of projection functions.

1. On input 1^λ , \mathcal{A}_{ske} first generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . Then, \mathcal{A}_{ske} prepares L_{ext} and L_{ch} .
2. \mathcal{A}_{ske} responds to queries made by \mathcal{A} as follows.

Extraction queries. When \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$ as an extraction query, \mathcal{A}_{ske} responds exactly in the same way as the challenger in Game 2.

We note that, in this case, \mathcal{A}_{ske} computes the answer $\text{Kdm.sk}_{\text{id}}$ using a freshly generated key \mathbf{K}_{id} of SKE.

Registration queries. When \mathcal{A} makes the k -th ($k \leq q_{\text{ch}}$) registration query $\text{id}^{(k)} \in \mathcal{ID} \setminus (L_{\text{ext}} \cup L_{\text{ch}})$, \mathcal{A}_{ske} relates $\text{id}^{(k)}$ to $\mathbf{K}^{(k)}$, where $\mathbf{K}^{(k)}$ is the k -th secret key of SKE generated by the challenger. \mathcal{A}_{ske} generates $\text{sk}_{\text{id}^{(k)}, j, \alpha} \leftarrow \text{KG}\left(\text{MSK}, \left(\text{id}^{(k)}, j, \alpha\right)\right)$ for every $j \in [\text{len}_{\mathbf{K}}]$ and $\alpha \in \{0, 1\}$. They are used for the conversion of functions. \mathcal{A}_{ske} then adds $\text{id}^{(k)}$ to L_{ch} .

KDM queries. When \mathcal{A} makes a KDM query $(\text{id}, f) \in L_{\text{ch}} \times \mathcal{P}$, \mathcal{A}_{ske} responds as follows.

- (a) \mathcal{A}_{ske} first computes a projection function g satisfying

$$g\left(\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}\right) = f\left(\left\{\mathbf{K}^{(k)}\right\}_{k \in [q_{\text{ch}}]}, \left\{\text{sk}_{\text{id}^{(k)}, j, \mathbf{K}^{(k)}[j]}\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}]}\right)$$

as we noted above from $\left\{\text{sk}_{\text{id}^{(k)}, j, \alpha}\right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\mathbf{K}}], \alpha \in \{0, 1\}}$.

- (b) Let $k \in [q_{\text{ch}}]$ be the number that related to id . Since id was added to L_{ch} , such $k \in [q_{\text{ch}}]$ exists. \mathcal{A}_{ske} queries (k, g) to the challenger as a KDM query and gets the answer CT_{ske} .
- (c) \mathcal{A}_{ske} computes $(\tilde{\text{E}}, \{\text{lab}_j\}_{j \in [\text{len}_{\mathbf{K}}]}) \leftarrow \text{Sim}(1^\lambda, |\text{E}|, \text{CT}_{\text{ske}})$ and for every $j \in [\text{len}_{\mathbf{K}}]$ and $\alpha \in \{0, 1\}$, computes $\text{CT}_{j, \alpha} \leftarrow \text{Enc}(\text{PP}, (\text{id}, j, \alpha), \text{lab}_j)$.

(d) \mathcal{A}_{ske} returns $\text{Kdm.CT} := \left(\tilde{\mathbf{E}}, \{\text{CT}_{j,\alpha}\}_{j \in [\text{len}_{\kappa}], \alpha \in \{0,1\}} \right)$ to \mathcal{A} .

3. When \mathcal{A} terminates with output $b' \in \{0, 1\}$, \mathcal{A}_{ske} outputs $\beta' = b'$.

\mathcal{A}_{ske} perfectly simulates Game 2 for \mathcal{A} in which the challenge bit is the same as that of \mathcal{P} -KDM-CPA game of SKE between the challenger and \mathcal{A}_{ske} . Moreover, \mathcal{A}_{ske} just outputs \mathcal{A} 's output. Thus, $\text{Adv}_{\text{SKE}, \mathcal{P}, \mathcal{A}_{\text{ske}}, q_{\text{ch}}}^{\text{kdmcpa}}(\lambda) = \left| \Pr[\text{SUC}_2] - \frac{1}{2} \right|$ holds. Since SKE is \mathcal{P} -KDM-CPA secure, $\left| \Pr[\text{SUC}_2] - \frac{1}{2} \right| = \text{negl}(\lambda)$ holds.

From the above arguments, we see that

$$\begin{aligned} \text{Adv}_{\text{KdmIBE}, \mathcal{P}, \mathcal{A}}^{\text{kdmcpa}}(\lambda) &= \left| \Pr[\text{SUC}_0] - \frac{1}{2} \right| \\ &\leq \sum_{t=0}^1 \left| \Pr[\text{SUC}_t] - \Pr[\text{SUC}_{t+1}] \right| + \left| \Pr[\text{SUC}_2] - \frac{1}{2} \right| = \text{negl}(\lambda). \end{aligned}$$

Since the choice of \mathcal{A} is arbitrary, KdmIBE satisfies \mathcal{P} -KDM-CPA security.

On the transformation of \mathcal{B} -KDM-CPA secure schemes. We can also construct \mathcal{B} -KDM-CPA secure IBE based on \mathcal{B} -KDM-CPA secure SKE via the construction. The security proof of \mathcal{B} -KDM-CPA secure IBE is in fact almost the same as that of \mathcal{P} -KDM-CPA secure IBE. The only issue we need to care is whether the conversion of functions performed by \mathcal{A}_{ske} is successful or not also when we construct \mathcal{B} -KDM-CPA secure IBE.

Let f be a function queried by an adversary \mathcal{A} for KdmIBE. As above, consider a function g such that

$$g \left(\left\{ \mathbf{K}^{(k)} \right\}_{k \in [q_{\text{ch}}]} \right) = f \left(\left\{ \mathbf{K}^{(k)} \right\}_{k \in [q_{\text{ch}}]}, \left\{ \text{sel}_{k,j} \left(\mathbf{K}^{(k)}[j] \right) \right\}_{k \in [q_{\text{ch}}], j \in [\text{len}_{\kappa}]} \right),$$

where the function $\text{sel}_{k,j}$ is the function we defined earlier. Since $\text{sel}_{k,j}$ is computable by a circuit of a-priori bounded size, we see that if f is computable by a circuit of a-priori bounded size, then so is g . Therefore, \mathcal{A}_{ske} can successfully perform the conversion of functions also when constructing \mathcal{B} -KDM-CPA secure IBE. \square (**Theorem 5**)

5 SIM-RSO Secure IBE Based on IND-ID-CPA Secure IBE

We construct SIM-RSO secure IBE based on any IND-ID-CPA secure IBE.

Let $\text{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ be an IBE scheme whose message space and identity space are $\{0, 1\}$ and $\mathcal{ID} \times \{0, 1\}$, respectively. Using IBE, we construct the following IBE scheme $\text{RsoIBE} = (\text{Rso.Setup}, \text{Rso.KG}, \text{Rso.Enc}, \text{Rso.Dec})$ whose message space and identity space are $\{0, 1\}$ and \mathcal{ID} .

Rso.Setup(1^λ):

– Return $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$.

Rso.KG(MSK, id):

- Generate $r \xleftarrow{r} \{0, 1\}$.
- Generate $\text{sk}_{\text{id},r} \leftarrow \text{KG}(\text{MSK}, (\text{id}, r))$.
- Return $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id},r})$.

Rso.Enc(PP, id, $m \in \{0, 1\}$):

- For every $\alpha \in \{0, 1\}$, compute $\text{CT}_\alpha \leftarrow \text{Enc}(\text{PP}, (\text{id}, \alpha), m)$.
- Return $\text{Rso.CT} := (\text{CT}_0, \text{CT}_1)$.

Rso.Dec(Rso.sk_{id}, Rso.CT):

- Parse $(r, \text{sk}_{\text{id},r}) \leftarrow \text{Rso.dk}$.
- Parse $(\text{CT}_0, \text{CT}_1) \leftarrow \text{Rso.CT}$.
- Return $m \leftarrow \text{Dec}(\text{sk}_{\text{id},r}, \text{CT}_r)$.

Correctness. The correctness of RsoIBE directly follows from that of IBE.

We prove the following theorem.

Theorem 6. *Let IBE be an IND-ID-CPA secure IBE scheme. Then, RsoIBE is a SIM-RSO secure IBE scheme.*

Proof of Theorem 6. Let \mathcal{A} be an adversary that attacks the SIM-RSO security of RsoIBE. We show the proof via the following sequence of games.

Let \mathcal{D} be an PPT distinguisher with binary output. For every $t \in \{0, 1, 2\}$, let T_t be the event that \mathcal{D} outputs 1 given the output of the challenger in Game t .

Game 0: This is the real game of SIM-RSO security regarding RsoIBE. The detailed description is as follows.

1. First, the challenger generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . The challenger prepares a list L_{ext} .

At any step of the game, \mathcal{A} can make key extraction queries.

Extraction queries. \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus L_{\text{ext}}$ to the challenger. The challenger responds as follows.

- (a) The challenger generates $r \xleftarrow{r} \{0, 1\}$.
- (b) The challenger generates $\text{sk}_{\text{id},r} \leftarrow \text{KG}(\text{MSK}, (\text{id}, r))$.
- (c) The challenger returns $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id},r})$.

2. \mathcal{A} sends q_{ch} identities $\{\text{id}^{(k)} \in \mathcal{ID} \setminus L_{\text{ext}}\}_{k \in [q_{\text{ch}}]}$ and a message distribution Dist on $\{0, 1\}^{q_{\text{ch}}}$ to the challenger, where q_{ch} is an a-priori unbounded polynomial of λ . The challenger generates $\{m^{(k)}\}_{k \in [q_{\text{ch}}]} \leftarrow \text{Dist}$ and computes $\text{Rso.CT}^{(k)}$ for every $k \in [q_{\text{ch}}]$ as follows.

- (a) The challenger computes $\text{CT}_\alpha^{(k)} \leftarrow \text{Enc}(\text{PP}, (\text{id}^{(k)}, \alpha), m^{(k)})$ for every $\alpha \in \{0, 1\}$.
- (b) The challenger sets $\text{Rso.CT}^{(k)} := (\text{CT}_0^{(k)}, \text{CT}_1^{(k)})$.

The challenger sends $\{\text{Rso.CT}^{(k)}\}_{k \in [q_{\text{ch}}]}$ to \mathcal{A} .

Below, \mathcal{A} is not allowed to make extraction queries for $\{\text{id}^{(k)}\}_{k \in [q_{\text{ch}}]}$.

3. \mathcal{A} sends a subset \mathcal{I} of $[q_{\text{ch}}]$ to the challenger. The challenger generates $\text{Rso.sk}_{\text{id}^{(k)}}$ for every $k \in \mathcal{I}$ as follows.
 - (a) The challenger generates $r^{(k)} \xleftarrow{r} \{0, 1\}$.
 - (b) The challenger generates $\text{sk}_{\text{id}^{(k)}, r^{(k)}} \leftarrow \text{KG} \left(\text{MSK}, \left(\text{id}^{(k)}, r^{(k)} \right) \right)$.
 - (c) The challenger sets $\text{Rso.sk}_{\text{id}^{(k)}} := \left(r^{(k)}, \text{sk}_{\text{id}^{(k)}, r^{(k)}} \right)$.
 The challenger sends $\left\{ \left(\text{Rso.sk}_{\text{id}^{(k)}}, m^{(k)} \right) \right\}_{k \in \mathcal{I}}$ to \mathcal{A} .
4. \mathcal{A} sends a string out to the challenger.
5. The challenger outputs

$$\text{out}_{\text{real}} := \left(\left\{ \text{id}^{(k)} \right\}_{k \in [q_{\text{ch}}]}, \left\{ m^{(k)} \right\}_{k \in [q_{\text{ch}}]}, \text{Dist}, \mathcal{I}, \text{out} \right).$$

Game 1: Same as Game 0 except that for every $k \in [q_{\text{ch}}]$, the challenger generates

$$\text{CT}_{1-r^{(k)}}^{(k)} \leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, 1 - r^{(k)} \right), 1 - m^{(k)} \right).$$

We note that the challenger generates $\text{CT}_{r^{(k)}}^{(k)} \leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, r^{(k)} \right), m^{(k)} \right)$ for every $k \in [q_{\text{ch}}]$ in both Games 0 and 1.

Secret keys for identities $\left\{ \left(\text{id}^{(k)}, 1 - r^{(k)} \right) \right\}_{k \in [q_{\text{ch}}]}$ of IBE are not given to \mathcal{A} regardless of which users \mathcal{A} corrupts in both Games 0 and 1. Therefore, by using the security of IBE q_{ch} times, we can prove $|\text{Pr}[\text{T}_0] - \text{Pr}[\text{T}_1]| = \text{negl}(\lambda)$.

Game 2: Same as Game 1 except that for every $k \in [q_{\text{ch}}]$, the challenger uses $r^{(k)} \oplus m^{(k)}$ instead of $r^{(k)}$ as the random bit contained in the k -th RsoIBE 's secret key $\text{Rso.sk}_{\text{id}^{(k)}}$ for $\text{id}^{(k)}$. We note that the challenger does not need $\left\{ r^{(k)} \right\}_{k \in [q_{\text{ch}}]}$ before generating $\left\{ m^{(k)} \right\}_{k \in [q_{\text{ch}}]}$. Thus, the transition from Games 1 to 2 makes sense, and $|\text{Pr}[\text{T}_2] - \text{Pr}[\text{T}_3]| = 0$ holds since $r^{(k)} \oplus m^{(k)}$ is distributed uniformly at random for every $k \in [q_{\text{ch}}]$.

In Game 2, uncorrupted messages $\left\{ m^{(k)} \right\}_{k \in [q_{\text{ch}}] \setminus \mathcal{I}}$ are completely hidden from the view of \mathcal{A} . To verify the fact, we confirm that ciphertexts $\left\{ \text{Rso.CT}^{(k)} \right\}_{k \in [q_{\text{ch}}]}$ are independent of $\left\{ m^{(k)} \right\}_{k \in [q_{\text{ch}}]}$.

For every $k \in [q_{\text{ch}}]$, the challenger generates $\text{Rso.CT}^{(k)} = \left(\text{CT}_0^{(k)}, \text{CT}_1^{(k)} \right)$ by computing

$$\begin{aligned} \text{CT}_{r^{(k)} \oplus m^{(k)}}^{(k)} &\leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, r^{(k)} \oplus m^{(k)} \right), m^{(k)} \right), \\ \text{CT}_{1-r^{(k)} \oplus m^{(k)}}^{(k)} &\leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, 1 - r^{(k)} \oplus m^{(k)} \right), 1 - m^{(k)} \right). \end{aligned}$$

We see that, regardless of the value of $m^{(k)} \in \{0, 1\}$, the challenger computes

$$\begin{aligned} \text{CT}_{r^{(k)}}^{(k)} &\leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, r^{(k)} \right), 0 \right), \\ \text{CT}_{1-r^{(k)}}^{(k)} &\leftarrow \text{Enc} \left(\text{PP}, \left(\text{id}^{(k)}, 1 - r^{(k)} \right), 1 \right). \end{aligned}$$

Therefore, we see that ciphertexts $\{\text{Rso.CT}^{(k)}\}_{k \in [q_{\text{ch}}]}$ are independent of $\{m^{(k)}\}_{k \in [q_{\text{ch}}]}$ in Game 2.

Then, we construct a simulator \mathcal{S} that perfectly simulates Game 2 for \mathcal{A} . The description of \mathcal{S} is as follows.

1. On input 1^λ , \mathcal{S} generates $(\text{PP}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ and sends PP to \mathcal{A} . \mathcal{S} then prepares a list L_{ext} .

Extraction queries. When \mathcal{A} sends $\text{id} \in \mathcal{ID} \setminus L_{\text{ext}}$, \mathcal{S} responds as follows.

- (a) \mathcal{S} generates $r \xleftarrow{r} \{0, 1\}$.
- (b) \mathcal{S} generates $\text{sk}_{\text{id}, r} \leftarrow \text{KG}(\text{MSK}, (\text{id}, r))$.
- (c) \mathcal{S} returns $\text{Rso.sk}_{\text{id}} := (r, \text{sk}_{\text{id}, r})$ to \mathcal{A} and adds id to L_{ext} .

2. When \mathcal{A} outputs a message distribution Dist with identities $\{\text{id}^{(k)}\}_{k \in [q_{\text{ch}}]}$, \mathcal{S} sends them to the challenger. Then, \mathcal{S} computes $\text{Rso.CT}^{(k)}$ for every $k \in [q_{\text{ch}}]$ as follows.

- (a) \mathcal{S} computes $r^{(k)} \xleftarrow{r} \{0, 1\}$.
- (b) \mathcal{S} computes

$$\begin{aligned} \text{CT}_{r^{(k)}}^{(k)} &\leftarrow \text{Enc}\left(\text{PP}, \left(\text{id}^{(k)}, r^{(k)}\right), 0\right) \text{ and} \\ \text{CT}_{1-r^{(k)}}^{(k)} &\leftarrow \text{Enc}\left(\text{PP}, \left(\text{id}^{(k)}, 1 - r^{(k)}\right), 1\right). \end{aligned}$$

- (c) \mathcal{S} sets $\text{Rso.CT}^{(k)} := \left(\text{CT}_0^{(k)}, \text{CT}_1^{(k)}\right)$.

\mathcal{S} sends $\{\text{Rso.CT}^{(k)}\}_{k \in [q_{\text{ch}}]}$ to \mathcal{A} .

3. When \mathcal{A} outputs a subset \mathcal{I} of $[q_{\text{ch}}]$, \mathcal{S} sends it to the challenger, and gets $\{m^{(k)}\}_{k \in \mathcal{I}}$. \mathcal{S} computes $\text{sk}_{\text{id}^{(k)}, r^{(k)} \oplus m^{(k)}} \leftarrow \text{KG}\left(\text{MSK}, \left(\text{id}^{(k)}, r^{(k)} \oplus m^{(k)}\right)\right)$, sets $\text{Rso.sk}_{\text{id}^{(k)}} := (r^{(k)} \oplus m^{(k)}, \text{sk}_{\text{id}^{(k)}, r^{(k)} \oplus m^{(k)}})$ for every $k \in \mathcal{I}$, and sends $\{(\text{Rso.sk}_{\text{id}^{(k)}}, m^{(k)})\}_{k \in \mathcal{I}}$ to \mathcal{A} .
4. When \mathcal{A} outputs a string out , \mathcal{S} outputs it.

\mathcal{S} perfectly simulates Game 2 for \mathcal{A} . Therefore, we have

$$\text{Adv}_{\text{RsolBE}, \mathcal{A}, \mathcal{S}, \mathcal{D}}^{\text{simrso}}(\lambda) = |\Pr[\text{T}_0] - \Pr[\text{T}_2]| \leq \sum_{t=0}^2 |\Pr[\text{T}_t] - \Pr[\text{T}_{t+1}]|. \quad (2)$$

From the above arguments, we see that each term of the right hand side of Inequality 2 is negligible in λ . Since the choice of \mathcal{A} and \mathcal{D} is arbitrary and the description of \mathcal{S} does not depend on that of \mathcal{D} , we see that for any \mathcal{A} , there exists \mathcal{S} such that for any \mathcal{D} we have $\text{Adv}_{\text{RsolBE}, \mathcal{A}, \mathcal{S}, \mathcal{D}}^{\text{simrso}}(\lambda) = \text{negl}(\lambda)$. This means that RsolBE is SIM-RSO secure. \square (**Theorem 6**)

6 KDM Secure PKE from KDM Secure SKE and IND-CPA Secure PKE

We show how to construct KDM secure PKE based on KDM secure SKE and IND-CPA secure PKE. The construction is similar to that of KDM secure IBE we show in Sect. 4 except that IND-CPA secure PKE is used instead of IND-ID-CPA secure IBE as a building block.

Let $\text{SKE} = (\text{G}, \text{E}, \text{D})$ be an SKE scheme whose message space is \mathcal{M} . Let len_K and len_r denote the length of a secret key and encryption randomness of SKE, respectively. Let $\text{PKE} = (\text{KG}, \text{Enc}, \text{Dec})$ be a PKE scheme and $\text{GC} = (\text{Garble}, \text{Eval})$ a garbling scheme. Using SKE, PKE, and GC, we construct the following PKE scheme $\text{KdmPKE} = (\text{Kdm.KG}, \text{Kdm.Enc}, \text{Kdm.Dec})$ whose message space is \mathcal{M} .

Kdm.KG(1^λ):

- Generate $K \leftarrow \text{G}(1^\lambda)$.
- Generate $(\text{ek}_{j,\alpha}, \text{dk}_{j,\alpha}) \leftarrow \text{KG}(1^\lambda)$ for every $j \in [\text{len}_K]$ and $\alpha \in \{0, 1\}$.
- Return $\text{Kdm.ek} := \{\text{ek}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}$ and $\text{Kdm.dk} := \left(K, \{\text{dk}_{j,K[j]}\}_{j \in [\text{len}_K]} \right)$.

Kdm.Enc($\text{Kdm.ek}, m$):

- Parse $\{\text{ek}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}} \leftarrow \text{Kdm.ek}$.
- Generate $r_E \xleftarrow{r} \{0, 1\}^{\text{len}_r}$ and compute $(\tilde{\text{E}}, \{\text{lab}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, \text{E}(\cdot, m; r_E))$, where $\text{E}(\cdot, m; r_E)$ is the encryption circuit E of SKE into which m and r_E are hardwired.
- For every $j \in [\text{len}_K]$ and $\alpha \in \{0, 1\}$, compute $\text{CT}_{j,\alpha} \leftarrow \text{Enc}(\text{ek}_{j,\alpha}, \text{lab}_{j,\alpha})$.
- Return $\text{Kdm.CT} := (\tilde{\text{E}}, \{\text{CT}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}})$.

Kdm.Dec($\text{Kdm.dk}, \text{Kdm.CT}$):

- Parse $(K, \{\text{dk}_j\}_{j \in [\text{len}_K]}) \leftarrow \text{Kdm.dk}$.
- Parse $(\tilde{\text{E}}, \{\text{CT}_{j,\alpha}\}_{j \in [\text{len}_K], \alpha \in \{0,1\}}) \leftarrow \text{Kdm.CT}$.
- For every $j \in [\text{len}_K]$, compute $\text{lab}_j \leftarrow \text{Dec}(\text{dk}_j, \text{CT}_{j,K[j]})$.
- Compute $\text{CT}_{\text{ske}} \leftarrow \text{Eval}(\tilde{\text{E}}, \{\text{lab}_j\}_{j \in [\text{len}_K]})$.
- Return $m \leftarrow \text{D}(K, \text{CT}_{\text{ske}})$.

Correctness. When decrypting a ciphertext of KdmPKE that encrypts a message m , we first obtain a ciphertext of SKE that encrypts m from the correctness of PKE and GC. The correctness of KdmPKE then follows from that of SKE.

We have the following theorem.

Theorem 7. *Let SKE be an SKE scheme that is \mathcal{P} -KDM-CPA secure (resp. \mathcal{B} -KDM-CPA secure). Let PKE be an IND-CPA secure PKE scheme and GC a secure garbling scheme. Then, KdmPKE is a PKE scheme that is \mathcal{P} -KDM-CPA secure (resp. \mathcal{B} -KDM-CPA secure).*

The proof for Theorem 7 is almost the same as that for Theorem 5. Thus, we omit it and provide in the full version of this paper [27].

7 SIM-RSO Secure PKE Based on IND-CPA Secure PKE

We can construct SIM-RSO secure PKE based on any IND-CPA secure PKE if we take the revelation of only secret keys into account. The construction is similar to that of SIM-RSO secure IBE we show in Sect. 5 except that IND-CPA secure PKE is used instead of IND-ID-CPA secure IBE.

Using a PKE scheme $\text{PKE} = (\text{KG}, \text{Enc}, \text{Dec})$, we construct the following PKE scheme $\text{RsoPKE} = (\text{Rso.KG}, \text{Rso.Enc}, \text{Rso.Dec})$ whose message space is $\{0, 1\}$.

$\text{Rso.KG}(1^\lambda)$:

- Generate $(\text{ek}_\alpha, \text{dk}_\alpha) \leftarrow \text{KG}(1^\lambda)$ for every $\alpha \in \{0, 1\}$.
- Generate $r \xleftarrow{r} \{0, 1\}$.
- Return $\text{Rso.ek} := (\text{ek}_0, \text{ek}_1)$ and $\text{Rso.dk} := (r, \text{dk}_r)$.

$\text{Rso.Enc}(\text{Rso.ek}, m \in \{0, 1\})$:

- Parse $(\text{ek}_0, \text{ek}_1) \leftarrow \text{Rso.ek}$.
- For every $\alpha \in \{0, 1\}$, compute $\text{CT}_\alpha \leftarrow \text{Enc}(\text{ek}_\alpha, m)$.
- Return $\text{Rso.CT} := (\text{CT}_0, \text{CT}_1)$.

$\text{Rso.Dec}(\text{Rso.dk}, \text{Rso.CT})$:

- Parse $(r, \text{dk}_r) \leftarrow \text{Rso.dk}$
- Parse $(\text{CT}_0, \text{CT}_1) \leftarrow \text{Rso.CT}$.
- Return $m \leftarrow \text{Dec}(\text{dk}_r, \text{CT}_r)$.

Correctness. The correctness of RsoPKE directly follows from that of PKE .

We have the following theorem.

Theorem 8. *Let PKE be an IND-CPA secure PKE scheme. Then, RsoPKE is a SIM-RSO secure PKE scheme.*

The proof for Theorem 8 is almost the same as that for Theorem 6. Thus, we omit it and provide in the full version of this paper [27].

Acknowledgement. We would like to thank the anonymous reviewers of PKC 2018 for their insightful comments. A part of this work was supported by Input Output Hong Kong, Nomura Research Institute, NTT Secure Platform Laboratories, Mitsubishi Electric, JST CREST JPMJCR14D6, JST OPERA, JSPS KAKENHI JP16H01705, JP16J10322, JP17H01695.

References

1. Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic agility and its relation to circular encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_21
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28

3. Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_20
4. Applebaum, B.: Key-dependent message security: generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_29
5. 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
6. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_22
7. Bellare, M., Dowsley, R., Waters, B., Yilek, S.: Standard security does not imply security against selective-opening. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 645–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_38
8. Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_1
9. Bellare, M., Waters, B., Yilek, S.: Identity-based encryption secure against selective opening attack. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 235–252. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_15
10. Bishop, A., Hohenberger, S., Waters, B.: New circular security counterexamples from decision linear and learning with errors. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 776–800. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_32
11. 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
12. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
13. 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
14. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_1
15. 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
16. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_16

17. Canetti, R., Halevi, S., Katz, J.: Adaptively-secure, non-interactive public-key encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 150–168. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_9
18. Cash, D., Green, M., Hohenberger, S.: New definitions and separations for circular security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 540–557. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_32
19. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_27
20. Döttling, N., Garg, S.: From selective IBE to full IBE and selective HIBE. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 372–408. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_13
21. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_18
22. Galindo, D., Herranz, J., Villar, J.L.: Identity-based encryption with master key-dependent message security and leakage-resilience. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 627–642. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_36
23. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
24. Hazay, C., Patra, A., Warinschi, B.: Selective opening security for receivers. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 443–469. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_19
25. Hofheinz, D., Rao, V., Wichs, D.: Standard security does not imply indistinguishability under selective opening. In: Hirt, M., Smith, A. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 121–145. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_5
26. Hofheinz, D., Rupp, A.: Standard versus selective opening security: separation and equivalence results. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 591–615. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_25
27. Kitagawa, F., Tanaka, K.: Key dependent message security and receiver selective opening security for identity-based encryption. IACR Cryptology ePrint Archive (2017)
28. Koppula, V., Waters, B.: Circular security separations for arbitrary length cycles from LWE. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 681–700. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_24
29. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, pp. 427–437 (1990)
30. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
31. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167 (1986)



Tightly SIM-SO-CCA Secure Public Key Encryption from Standard Assumptions

Lin Lyu^{1,2} , Shengli Liu^{1,2,3} , Shuai Han^{1,2,4} , and Dawu Gu^{1,5}

¹ Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{lvlin, slliu, dalen17, dwgu}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ Westone Cryptologic Research Center, Beijing 100070, China

⁴ Karlsruhe Institute of Technology, Karlsruhe, Germany

⁵ Shanghai Institute for Advanced Communication and Data Science,
Shanghai, China

Abstract. Selective opening security (SO security) is desirable for public key encryption (PKE) in a multi-user setting. In a selective opening attack, an adversary receives a number of ciphertexts for possibly correlated messages, then it opens a subset of them and gets the corresponding messages together with the randomnesses used in the encryptions. SO security aims at providing security for the unopened ciphertexts. Among the existing simulation-based, selective opening, chosen ciphertext secure (SIM-SO-CCA secure) PKEs, only one (Libert *et al.* Crypto'17) enjoys *tight* security, which is reduced to the Non-Uniform LWE assumption. However, their public key and ciphertext are not compact.

In this work, we focus on constructing PKE with *tight* SIM-SO-CCA security based on standard assumptions. We formalize security notions needed for key encapsulation mechanism (KEM) and show how to transform these securities into SIM-SO-CCA security of PKE through a tight security reduction, while the construction of PKE from KEM follows the general framework proposed by Liu and Paterson (PKC'15). We present two KEM constructions with tight securities based on the Matrix Decision Diffie-Hellman assumption. These KEMs in turn lead to two tightly SIM-SO-CCA secure PKE schemes. One of them enjoys not only tight security but also compact public key.

1 Introduction

Selective Opening Security. In the context of public key encryption (PKE), IND-CPA (CCA) security is widely believed to be the right security notion. However, multi-user settings enable more complicated attacks and the traditional IND-CPA (CCA) security may not be strong enough. Consider a scenario of N senders and one receiver. The senders encrypt N (possibly correlated) messages $\mathbf{m}_1, \dots, \mathbf{m}_N$ under the receiver's public key \mathbf{pk} using fresh randomnesses $\mathbf{r}_1, \dots, \mathbf{r}_N$ to get ciphertexts $\mathbf{c}_1, \dots, \mathbf{c}_N$, respectively, i.e., each sender i

computes $\mathbf{c}_i = \text{Enc}(\text{pk}, \mathbf{m}_i; \mathbf{r}_i)$. Upon receiving the ciphertexts $\mathbf{c}_1, \dots, \mathbf{c}_N$, the adversary might be able to open a subset of them via implementing corruptions. Namely, by corrupting a subset of users, say $I \subset [N]$, the adversary obtains the messages $\{\mathbf{m}_i\}_{i \in I}$ together with the randomnesses $\{\mathbf{r}_i\}_{i \in I}$. Such an attack is called selective opening attack (SOA). It is desirable that the unopened ciphertexts $\{\mathbf{c}_i\}_{i \in [N] \setminus I}$ still protect the privacy of $\{\mathbf{m}_i\}_{i \in [N] \setminus I}$, which is exactly what the SO security concerns.

The potential correlation between $\{\mathbf{m}_i\}_{i \in I}$ and $\{\mathbf{m}_i\}_{i \in [N] \setminus I}$ hinders the use of hybrid argument proof technique. Hence, traditional IND-CPA security may not imply SO security. To date, there exist two types of SO security formalizations: indistinguishability-based SO security (IND-SO, [1, 2]) and simulation-based SO security (SIM-SO, [1, 5]). According to whether the adversary has access to a decryption oracle, these securities are further classified into IND-SO-CPA, IND-SO-CCA, SIM-SO-CPA and SIM-SO-CCA.

Intuitively, IND-SO security requires that, given public key pk , ciphertexts $\{\mathbf{c}_i\}_{i \in [N]}$, the opened messages $\{\mathbf{m}_i\}_{i \in I}$ and randomnesses $\{\mathbf{r}_i\}_{i \in I}$ (together with a decryption oracle in the CCA case), the unopened messages $\{\mathbf{m}_i\}_{i \in [N] \setminus I}$ remain computationally indistinguishable from independently sampled messages conditioned on the already opened messages $\{\mathbf{m}_i\}_{i \in I}$. Accordingly, the IND-SO security usually requires the message distributions be *efficiently conditionally re-samplable* [1, 10, 11] (and such security is referred to as *weak* IND-SO security in [2]), which limits its application scenarios.

On the other hand, SIM-SO security is conceptually similar to semantic security [9]. It requires that the output of the SO adversary can be simulated by a simulator which only takes the opened messages $\{\mathbf{m}_i\}_{i \in I}$ as its input after it assigns the corruption set I . Since there is no restriction on message distribution, SIM-SO security has an advantage over IND-SO security from an application point of view. SIM-SO security was also shown to be stronger than (weak) IND-SO security in [2]. However, as shown in [13], SIM-SO security turns out to be significantly harder to achieve.

Generally speaking, there are two approaches to achieve SIM-SO-CCA security. The first approach uses lossy trapdoor functions [22], All-But- N lossy trapdoor functions [10] or All-But-Many lossy trapdoor functions [11] to construct lossy encryption schemes. If this lossy encryption has an efficient opener, then the resulting PKE scheme can be proven to be SIM-SO-CCA secure as shown in [1]. A DCR-based scheme in [11] and a LWE-based scheme in [18] are the only two schemes known to have such an opener. The second approach uses extended hash proof system and cross-authentication codes (XACs) [6]. As pointed out in [14, 15], a stronger property of XAC is required to make this proof rigorous. Following this line of research, Liu and Paterson proposed a general framework for constructing SIM-SO-CCA PKE from a special kind of key encapsulation mechanism (KEM) in combination with a strengthened XAC [19].

Tight Security Reductions. Usually, the security of a cryptographic primitive is established on the hardness of some underlying mathematical problems through a *security reduction*. It shows that any successful probabilistic

polynomial-time (PPT) adversary \mathcal{A} breaking the cryptographic primitive with advantage $\epsilon_{\mathcal{A}}$ can be transformed into a successful PPT problem solver \mathcal{B} for the underlying hard problem with advantage $\epsilon_{\mathcal{B}}$. The ideal case is $\epsilon_{\mathcal{A}} = \epsilon_{\mathcal{B}}$. However, most reductions suffer from a loss in the advantage, for example, $\epsilon_{\mathcal{A}} = L \cdot \epsilon_{\mathcal{B}}$ where L is called *security loss factor* of the reduction. Smaller L always indicates a better security level for a fixed security parameter. For a PKE scheme, L usually depends on λ (the security parameter) as well as Q_e (the number of challenge ciphertexts) and Q_d (the number of decryption queries). A security reduction for a PKE scheme is *tight* and the PKE scheme is called a *tightly secure* one [7, 12] if L depends only on the security parameter λ ¹ (and is independent of both Q_e and Q_d). Note that for concrete settings, λ is much smaller than Q_e and Q_d (for example, $\lambda = 80$ and Q_e, Q_d can be as large as 2^{20} or even 2^{30} in some settings). Most reductions are not tight and it appears to be a non-trivial problem to construct tightly IND-CCA secure PKE schemes.

Among the existing SIM-SO-CCA secure PKEs, only one of them has a tight security reduction [18]. Very recently, Libert *et al.* [18] provide an all-but-many lossy trapdoor function with an efficient opener, leading to a tightly SIM-SO-CCA secure PKE based on the Non-Uniform LWE assumption. Note that, their construction relies on a specific tightly secure PRF which is computable in NC^1 . So far, no construction of such a PRF based on standard LWE assumption is known, which is why their PKE has to rely on a non-standard assumption. Meanwhile, there is no PKE scheme enjoying both tight SIM-SO-CCA security and compact public key & ciphertext up to now.

1.1 Our Contribution

We explore how to construct tightly SIM-SO-CCA secure PKE based on standard assumptions. Following the KEM+XAC framework proposed in [19],

- we characterize stronger security notions needed for KEM and present a tightness preserving security reduction, which shows the PKE is tightly SIM-SO-CCA secure as long as the underlying KEM is tightly secure;
- we present two KEM instantiations and prove that their security can be tightly reduced to the Matrix Decision Diffie-Hellman (MDDH) assumption, thus leading to two tightly SIM-SO-CCA secure PKE schemes. One of them enjoys not only tight security but also compact public key.

1.2 Technique Overview

Roughly speaking, to prove the SIM-SO-CCA security of a PKE (see for Definition 1), for any PPT adversary, we need to construct a simulator and show that the adversary’s outputs are indistinguishable with those of the simulator. Naturally, such a simulator can be realized simply by simulating the entire real

¹ According to [3, 8], such a security reduction is called an *almost* tight one and a security reduction is tight only if L is a constant.

SO-CCA environment, invoking the adversary and returning the adversary's outputs. However, due to lack of essential information like messages and randomnesses, the simulator is not able to provide a perfect environment directly. Therefore, both the PKE scheme and the simulator has to be carefully designed, so that the simulator is able to provide the adversary a *computational indistinguishable* environment. To this end, we have to solve two problems.

- The first problem is how the simulator prepares ciphertexts for the adversary without knowing the messages.
- The second problem is how the simulator prepares randomnesses for the adversary according to the opened messages $\{\mathbf{m}_i\}_{i \in I}$ that it receives later.

To solve the first problem, the simulator has to provide ciphertexts that are computational indistinguishable with real ciphertexts in the setting of selective opening (together with chosen-ciphertext attacks). As to the second problem, note that the adversary can always check the consistence between $\{\mathbf{m}_i\}_{i \in I}, \{\mathbf{c}_i\}_{i \in I}$ and the randomnesses by re-encryption. Therefore, the simulator should not only provide indistinguishable ciphertexts but also be able to explain these ciphertexts as encryptions of any designated messages.

Liu and Paterson [19] solved these two problems and proposed a general framework for constructing SIM-SO-CCA secure PKE with the help of KEM in combination with XAC. Their PKE construction encrypts message in a bitwise manner. Suppose the message \mathbf{m} has bit length ℓ . If the i -th bit of \mathbf{m} is 1 ($\mathbf{m}_i = 1$), a pair of encapsulation ψ_i and key γ_i is generated from KEM, i.e., $(\psi_i, \gamma_i) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$. If $\mathbf{m}_i = 0$, a random pair is generated, i.e., $(\psi_i, \gamma_i) \leftarrow_{\S} \Psi \times \Gamma$. Then a tag T is generated to bind up $(\gamma_1, \dots, \gamma_\ell)$ and $(\psi_1, \dots, \psi_\ell)$ via XAC. And the final ciphertext is $C = (\psi_1, \dots, \psi_\ell, T)$.

They construct a simulator in the following way.

- Without knowledge of the message, the simulator uses an encryption of 1^ℓ as the ciphertext. Thus the encryption involves ℓ encapsulated pairs $(\psi_i, \gamma_i) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$. The simulator then saves all the randomnesses used in these encapsulations.
- When providing the randomnesses for the opened messages, the simulator checks the opened messages bit by bit. If a specific bit is 1, then the simulator outputs the original randomnesses and the simulation is perfect. Otherwise, the simulator views the encapsulated pair as a random pair. Then the simulator resamples randomnesses as if this pair is randomly chosen using these resampled randomnesses.

Thanks to the bit-wise encryption mode and the resampling property of spaces Ψ and Γ , an encapsulation pair (encrypting bit 1) can be easily explained as a random pair (encrypting bit 0). Therefore the second problem is solved.

To solve the first problem, one has to show that the encapsulated pairs and the random pairs are computationally indistinguishable. In [19], a special security named IND-tCCCA is formalized for KEM. This security guarantees that *one* encapsulated pair is computationally indistinguishable with *one* random pair

even when a constrained decryption oracle is provided. With the help of IND-tCCCA security of KEM, the indistinguishability between the encryption of 1^ℓ and the encryption of real messages are proved with ℓ hybrid arguments, each hybrid replacing only one encapsulated pair with one random pair.

To pursue tight security reduction, the ℓ hybrid arguments have to be avoided. To this end, we enhance the IND-tCCCA security and consider the pseudorandomness for *multiple* pairs even when a constrained decryption oracle is provided. This new security for KEM is formalized as mPR-CCCA security in Definition 5. Armed with this enhanced security, it is possible to replace the ℓ encapsulated pairs once for all in the security reduction from the SIM-SO-CCA security of PKE to the mPR-CCCA security of KEM. However, this gives rise to another problem. The SIM-SO-CCA adversary \mathcal{A} may submit a fresh ciphertext which shares the same encapsulation ψ with some challenge encapsulation. In the security reduction, the adversary \mathcal{B} , who invokes \mathcal{A} to attack the mPR-CCCA security of KEM, cannot ask its own decapsulation oracle to decapsulate ψ since ψ is already embedded in some challenge ciphertext for \mathcal{A} . To solve this problem, we define another security notion for KEM, namely, the Random Encapsulation Rejection (RER) security of KEM (cf. Definition 6). Equipped with the RER security of KEM and a security of XAC, \mathcal{B} could simply set 0 as the decryption bit for ψ .

Although the enhancement from IND-tCCCA to mPR-CCCA is conceptually simple, finding an mPR-CCCA secure KEM instantiation with tight reduction to standard assumptions is highly non-trivial. Inspired by the recent work on constructing tightly IND-CCA secure PKE [7, 8], we are able to give two tightly mPR-CCCA & RER secure KEM instantiations, one of which also enjoys compact public key.

1.3 Instantiation Overview

We provide two KEM instantiations.

The first KEM instantiation is inspired by a recent work in Eurocrypt'16. In the work [7], Gay *et al.* proposed the first tightly multi-challenge IND-CCA secure PKE scheme based on the MDDH assumption. From their PKE construction, we extract a KEM and tightly prove its mPR-CCCA security & RER security based on the MDDH assumption.²

The second KEM instantiation is contained in a very recent work by Gay *et al.* [8] in Crypto'17. In [8], a qualified proof system (QPS) is proposed to construct multi-challenge IND-CCCA secure KEM, which can be used to obtain a tightly multi-challenge IND-CCA secure PKE scheme with help of an authenticated encryption scheme. Note that our mPR-CCCA security is stronger than multi-challenge IND-CCCA security. To achieve mPR-CCCA security, we formalize a so-called Pseudorandom Simulated Proof property for QPS. We prove that if

² In [20], a PKE with tight SIM-SO-CCA security is constructed directly on the MDDH assumption. Our work unified their work by characterizing the mPR-CCCA security and RER security for KEM.

QPS has this property, the KEM from QPS is mPR-CCCA secure. Finally, we show that the QPS in [8] possesses the pseudorandom simulated proof property.

Compared with the first instantiation, the public key of our second KEM instantiation has a constant number of group elements. The compactness of public key is in turn transferred to the PKE, resulting in the first tightly SIM-SO-CCA secure PKE based on standard assumptions together with a compact public key.

2 Preliminaries

We use λ to denote the security parameter in this work. Let ε be the empty string. For $n \in \mathbb{N}$, denote by $[n]$ the set $\{1, \dots, n\}$. Denote by $s_1, \dots, s_n \leftarrow_{\S} S$ the process of picking n elements uniformly from set S . For a PPT algorithm \mathcal{A} , we use $y \leftarrow \mathcal{A}(x; r)$ to denote the process of running \mathcal{A} on input x with randomness r and assigning the deterministic result to y . Let $\mathcal{R}_{\mathcal{A}}$ be the randomness space of \mathcal{A} , we use $y \leftarrow_{\S} \mathcal{A}(x)$ to denote $y \leftarrow \mathcal{A}(x; r)$ where $r \leftarrow_{\S} \mathcal{R}_{\mathcal{A}}$. We use $\mathbf{T}(\mathcal{A})$ to denote the running time of \mathcal{A} , which is a polynomial in λ if \mathcal{A} is PPT.

We use boldface letters to denote vectors or matrices. For a vector \mathbf{m} of finite dimension, $|\mathbf{m}|$ denotes the dimension of the vector and \mathbf{m}_i denotes the i -th component of \mathbf{m} . For a set $I = \{i_1, i_2, \dots, i_{|I|}\} \subseteq [|\mathbf{m}|]$, define $\mathbf{m}_I := (\mathbf{m}_{i_1}, \mathbf{m}_{i_2}, \dots, \mathbf{m}_{i_{|I|}})$. For all matrix $\mathbf{A} \in \mathbb{Z}_q^{\ell \times k}$ with $\ell > k$, $\overline{\mathbf{A}} \in \mathbb{Z}_q^{k \times k}$ denotes the upper square matrix of \mathbf{A} and $\underline{\mathbf{A}} \in \mathbb{Z}_q^{(\ell-k) \times k}$ denotes the lower $\ell - k$ rows of \mathbf{A} . By $\text{span}(\mathbf{A}) := \{\mathbf{A}\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_q^k\}$, we denote the span of \mathbf{A} . By $\text{Ker}(\mathbf{A}^\top)$, we denote the orthogonal space of $\text{span}(\mathbf{A})$. For $\ell = k$, we define the *trace* of \mathbf{A} as the sum of all diagonal elements of \mathbf{A} , i.e., $\text{trace}(\mathbf{A}) := \sum_{i=1}^k \mathbf{A}_{i,i}$.

A function $f(\lambda)$ is *negligible*, if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

We use game-based security proof. The games are illustrated using pseudo-codes in figures. By a box in a figure, we denote that the codes in the box appears

in a specific game. For example, $\boxed{\boxed{G_4}} \boxed{G_5}$ means that G_4 contains the codes in

$\boxed{\text{dash box}}$, G_5 contains the codes in $\boxed{\text{oval box}}$, and both of them contain codes

in $\boxed{\text{square box}}$. Moreover, we assume that the unboxed codes are contained in

all games. We use the notation $\Pr_i[\mathbf{E}]$ to denote the probability that event \mathbf{E} occurs in game G_i , and use the notation $G \Rightarrow 1$ to denote the event that game G returns 1. All variables in games are initialized to \perp . We use “ \square ” to denote the end of proof of lemmas and use “ \blacksquare ” to denote the end of proof of theorems.

Due to space limitations, we refer to the full version of this paper [21] for the definitions of collision resistant hash function, universal hash function, public key encryption, the MDDH assumption and its random self-reducibility property, together with leftover hash lemma.

2.1 Prime-Order Groups

Let $\mathbb{G}\text{Gen}$ be a PPT algorithm that on input 1^λ returns $\mathcal{G} = (\mathbb{G}, q, P)$, a description of an additive cyclic group \mathbb{G} with a generator P of order q which is a λ -bit prime. For $a \in \mathbb{Z}_q$, define $[a] := aP \in \mathbb{G}$ as the *implicit representation* of a in \mathbb{G} . More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n \times m}$, we define $[\mathbf{A}]$ as the implicit representation of \mathbf{A} in \mathbb{G} , i.e., $[\mathbf{A}] := (a_{ij}P) \in \mathbb{G}^{n \times m}$. Note that from $[a] \in \mathbb{G}$ it is generally hard to compute the value a (discrete logarithm problem is hard in \mathbb{G}). Obviously, given $[a], [b] \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}$, one can efficiently compute $[ax] \in \mathbb{G}$ and $[a + b] \in \mathbb{G}$. Similarly, for $\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{B} \in \mathbb{Z}_q^{n \times t}$, given \mathbf{A}, \mathbf{B} or $[\mathbf{A}], \mathbf{B}$ or $\mathbf{A}, [\mathbf{B}]$, one can efficiently compute $[\mathbf{AB}] \in \mathbb{G}^{m \times t}$.

2.2 Simulation-Based, Selective-Opening CCA Security of PKE

Let \mathbf{m} and \mathbf{r} be two vectors of dimension $n := n(\lambda)$. Define $\text{Enc}(\text{pk}, \mathbf{m}; \mathbf{r}) := (\text{Enc}(\text{pk}, \mathbf{m}_1; \mathbf{r}_1), \dots, \text{Enc}(\text{pk}, \mathbf{m}_n; \mathbf{r}_n))$ where \mathbf{r}_i is a fresh randomness used for the encryption of \mathbf{m}_i for $i \in [n]$. Then we review the SIM-SO-CCA security definition in [6]. Let \mathcal{M} denote an n -message sampler, which on input a string $\alpha \in \{0, 1\}^*$ outputs a message vector \mathbf{m} of dimension n , i.e., $\mathbf{m} = (\mathbf{m}_1, \dots, \mathbf{m}_n)$. Let R be any PPT relation.

$\text{Exp}_{\text{PKE}, \mathcal{A}, n, \mathcal{M}, R}^{\text{so-cca-real}}(\lambda):$	$\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda):$
$(\text{pk}, \text{sk}) \leftarrow_{\S} \text{Gen}(1^\lambda)$	$(\alpha, s_1) \leftarrow_{\S} \mathcal{S}_1(1^\lambda)$
$(\alpha, a_1) \leftarrow_{\S} \mathcal{A}_1^{\text{Dec}(\cdot)}(\text{pk})$	$\mathbf{m} \leftarrow_{\S} \mathcal{M}(\alpha)$
$\mathbf{m} \leftarrow_{\S} \mathcal{M}(\alpha), \mathbf{r} \leftarrow_{\S} (\mathcal{R}_{\text{Enc}})^n$	$(I, s_2) \leftarrow_{\S} \mathcal{S}_2(s_1, (1^{ \mathbf{m}_i })_{i \in [n]})$
$\mathbf{C} \leftarrow \text{Enc}(\text{pk}, \mathbf{m}; \mathbf{r})$	$out_{\mathcal{S}} \leftarrow_{\S} \mathcal{S}_3(s_2, \mathbf{m}_I)$
$(I, a_2) \leftarrow_{\S} \mathcal{A}_2^{\text{Dec}_{\neq \mathbf{C}(\cdot)}}(a_1, \mathbf{C})$	$\hat{\mathbf{r}}_I \leftarrow \mathbf{r}_I$
$\hat{\mathbf{r}}_I \leftarrow \mathbf{r}_I$	$out_{\mathcal{A}} \leftarrow_{\S} \mathcal{A}_3^{\text{Dec}_{\neq \mathbf{C}(\cdot)}}(a_2, \mathbf{m}_I, \hat{\mathbf{r}}_I)$
$out_{\mathcal{A}} \leftarrow_{\S} \mathcal{A}_3^{\text{Dec}_{\neq \mathbf{C}(\cdot)}}(a_2, \mathbf{m}_I, \hat{\mathbf{r}}_I)$	$\text{Return } R(\mathbf{m}, I, out_{\mathcal{S}})$
$\text{Return } R(\mathbf{m}, I, out_{\mathcal{A}})$	

Fig. 1. Experiments used in the definition of SIM-SO-CCA security of PKE

Definition 1 (SIM-SO-CCA Security). A PKE scheme $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ is *simulation-based, selective-opening, chosen-ciphertext secure (SIM-SO-CCA secure)* if for every PPT n -message sampler \mathcal{M} , every PPT relation R , every stateful PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a stateful PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ such that $\text{Adv}_{\text{PKE}, \mathcal{A}, \mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca}}(\lambda)$ is negligible, where

$$\text{Adv}_{\text{PKE}, \mathcal{A}, \mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca}}(\lambda) := \left| \Pr \left[\text{Exp}_{\text{PKE}, \mathcal{A}, n, \mathcal{M}, R}^{\text{so-cca-real}}(\lambda) = 1 \right] - \Pr \left[\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda) = 1 \right] \right|.$$

Experiments $\text{Exp}_{\text{PKE}, \mathcal{A}, n, \mathcal{M}, R}^{\text{so-cca-real}}(\lambda)$ and $\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda)$ are defined in Fig. 1. Here the restriction on \mathcal{A} is that $\mathcal{A}_2, \mathcal{A}_3$ are not allowed to query the decryption oracle $\text{Dec}(\cdot)$ with any challenge ciphertext $\mathbf{C}_i \in \mathbf{C}$.

2.3 Efficiently Samplable and Explainable (ESE) Domain

A domain \mathcal{D} is said to be *efficiently samplable and explainable* (ESE) [6] if there exist two PPT algorithms ($\text{Sample}_{\mathcal{D}}, \text{Sample}_{\mathcal{D}}^{-1}$) where $\text{Sample}_{\mathcal{D}}(1^\lambda)$ outputs a uniform element over \mathcal{D} and $\text{Sample}_{\mathcal{D}}^{-1}(x)$, on input $x \in \mathcal{D}$, outputs r that is uniformly distributed over the set $\{r \in \mathcal{R}_{\text{Sample}_{\mathcal{D}}} \mid \text{Sample}_{\mathcal{D}}(1^\lambda; r) = x\}$.

It was shown by Damgård and Nielsen in [4] that any dense subset of an efficiently samplable domain is ESE as long as the dense subset admits an efficient membership test.

2.4 Cross-Authentication Codes

The concept of XAC was first proposed by Fehr *et al.* in [6] and later adapted to strong XAC in [15] and strengthened XAC in [17].

Definition 2 (ℓ -Cross-Authentication Code, XAC).

An ℓ -cross-authentication code XAC (for $\ell \in \mathbb{N}$) consists of three PPT algorithms ($\text{XGen}, \text{XAuth}, \text{XVer}$) and two associated spaces, the key space \mathcal{XK} and the tag space \mathcal{XT} . The key generation algorithm $\text{XGen}(1^\lambda)$ outputs a uniformly random key $K \in \mathcal{XK}$, the authentication algorithm $\text{XAuth}(K_1, \dots, K_\ell)$ takes ℓ keys $(K_1, \dots, K_\ell) \in \mathcal{XK}^\ell$ as input and outputs a tag $T \in \mathcal{XT}$, and the verification algorithm $\text{XVer}(K, T)$ outputs a decision bit.

Correctness. $\text{fail}_{\text{XAC}}(\lambda) := \Pr[\text{XVer}(K_i, \text{XAuth}(K_1, \dots, K_\ell)) \neq 1]$ is negligible for all $i \in [\ell]$, where the probability is taken over $K_1, \dots, K_\ell \leftarrow_{\$} \mathcal{XK}$.

Security against impersonation and substitution attacks. Define $\epsilon_{\text{XAC}}^{\text{imp}}(\lambda) := \max_{T'} \Pr[\text{XVer}(K, T') = 1 \mid K \leftarrow_{\$} \mathcal{XK}]$ where max is over all $T' \in$

$$\mathcal{XT}, \text{ and } \epsilon_{\text{XAC}}^{\text{sub}}(\lambda) := \max_{i, K_{\neq i}, F} \Pr \left[\begin{array}{c} T' \neq T \\ \text{XVer}(K_i, T') = 1 \end{array} \middle| \begin{array}{c} K_i \leftarrow_{\$} \mathcal{XK}, \\ T \leftarrow \text{XAuth}(K_1, \dots, K_\ell), \\ T' \leftarrow F(T) \end{array} \right]$$

where max is over all $i \in [\ell]$, all $K_{\neq i} := (K_j)_{j \in [\ell], j \neq i} \in \mathcal{XK}^{\ell-1}$ and all (possibly randomized) functions $F: \mathcal{XT} \rightarrow \mathcal{XT}$. Then we say XAC is secure against impersonation and substitution attacks if both $\epsilon_{\text{XAC}}^{\text{imp}}(\lambda)$ and $\epsilon_{\text{XAC}}^{\text{sub}}(\lambda)$ are negligible.

Definition 3 (Strong and semi-unique XACs). An ℓ -cross-authentication code XAC is strong and semi-unique if it has the following two properties.

Strongness [15]. There exists a PPT algorithm ReSamp , which takes as input $T \in \mathcal{XT}$ and $i \in [\ell]$, with $K_1, \dots, K_\ell \leftarrow_{\$} \text{XGen}(1^\lambda), T \leftarrow \text{XAuth}(K_1, \dots, K_\ell)$, and outputs $\hat{K}_i \in \mathcal{XK}$, denoted by $\hat{K}_i \leftarrow_{\$} \text{ReSamp}(T, i)$. Suppose for each fixed $(k_1, \dots, k_{\ell-1}, t) \in (\mathcal{XK})^{\ell-1} \times \mathcal{XT}$, the statistical distance between \hat{K}_i and K_i , conditioned on $(K_{\neq i}, T) = (k_1, \dots, k_{\ell-1}, t)$, is bounded by $\delta(\lambda)$, i.e.,

$$\frac{1}{2} \sum_{k \in \mathcal{XK}} \left| \frac{\Pr[\hat{K}_i = k \mid (K_{\neq i}, T) = (k_1, \dots, k_{\ell-1}, t)]}{\Pr[K_i = k \mid (K_{\neq i}, T) = (k_1, \dots, k_{\ell-1}, t)]} \right| \leq \delta(\lambda).$$

Then the code XAC is said to be $\delta(\lambda)$ -strong or strong if $\delta(\lambda)$ is negligible.

Semi-uniqueness [17]. *The code XAC is said to be semi-unique if $\mathcal{XK} = \mathcal{K}_x \times \mathcal{K}_y$, and given $T \in \mathcal{XT}$ and $K^x \in \mathcal{K}_x$, there exists at most one $K^y \in \mathcal{K}_y$ such that $\text{XVer}((K^x, K^y), T) = 1$.*

See the full version [21] for a concrete XAC instantiation by Fehr *et al.* in [6].

3 Key Encapsulation Mechanism

In this section, we recall the definition of key encapsulation mechanism and formalize two new security notions for it.

Definition 4 (Key Encapsulation Mechanism). *A KEM KEM is a tuple of PPT algorithms $(\text{KGen}, \text{KEnc}, \text{KDec})$ such that, $\text{KGen}(1^\lambda)$ generates a (public, secret) key pair $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}})$; $\text{KEnc}(\text{pk}_{\text{kem}})$ returns an encapsulation $\psi \in \Psi$ and a key $\gamma \in \Gamma$, where Ψ is the encapsulation space and Γ is the key space; $\text{KDec}(\text{sk}_{\text{kem}}, \psi)$ deterministically decapsulates ψ with sk_{kem} to get $\gamma \in \Gamma$ or \perp .*

We say KEM is perfectly correct if for all λ , $\Pr[\text{KDec}(\text{sk}_{\text{kem}}, \psi) = \gamma] = 1$, where $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda)$ and $(\psi, \gamma) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$.

3.1 mPR-CCCA Security for KEM

We formalize a new security notion for KEM, namely mPR-CCCA. Roughly speaking, mPR-CCCA security guarantees pseudorandomness of multiple (ψ, γ) pairs outputted by KEnc even if a constrained decapsulation oracle is provided.

Definition 5 (mPR-CCCA Security for KEM). *Let \mathcal{A} be an adversary and $b \in \{0, 1\}$ be a bit. Let $\text{KEM} = (\text{KGen}, \text{KEnc}, \text{KDec})$ be a KEM with encapsulation space Ψ and key space Γ . Define the experiment $\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{mpr-ccca-}b}(\lambda)$ in Fig. 2.*

$\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{mpr-ccca-}b}(\lambda) : // b \in \{0, 1\}$ $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda)$ $b' \leftarrow_{\S} \mathcal{A}^{\mathcal{O}_{\text{enc}}(\cdot), \mathcal{O}_{\text{dec}}(\cdot, \cdot)}(\text{pk}_{\text{kem}})$ Return b'	$\mathcal{O}_{\text{enc}}():$ $(\psi_0, \gamma_0) \leftarrow_{\S} \Psi \times \Gamma$ $(\psi_1, \gamma_1) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$ $\psi_{\text{enc}} \leftarrow \psi_{\text{enc}} \cup \{\psi_b\}$ Return (ψ_b, γ_b)	$\mathcal{O}_{\text{dec}}(\text{pred}, \psi):$ $\gamma \leftarrow \text{KDec}(\text{sk}_{\text{kem}}, \psi)$ $\text{Return } \begin{cases} \gamma & \text{If } \left(\psi \notin \psi_{\text{enc}} \wedge \right. \\ & \left. \text{pred}(\gamma) = 1 \right) \\ \perp & \text{Otherwise} \end{cases}$
--	---	---

Fig. 2. Experiment used in the definition of mPR-CCCA security of KEM

In $\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{mpr-ccca-}b}(\lambda)$, $\text{pred} : \Gamma \cup \{\perp\} \rightarrow \{0, 1\}$ denotes a PPT predicate and $\text{pred}(\perp) := 0$. Let Q_{dec} be the total number of decapsulation queries made by \mathcal{A} , which is independent of the environment without loss of generality. The uncertainty of \mathcal{A} is defined as $\text{uncert}_{\mathcal{A}}(\lambda) := \frac{1}{Q_{\text{dec}}} \sum_{i=1}^{Q_{\text{dec}}} \Pr_{\gamma \leftarrow_{\S} \Gamma}[\text{pred}_i(\gamma) = 1]$, where pred_i is the predicate in the i -th \mathcal{O}_{dec} query.

We say KEM has multi-encapsulation pseudorandom security against constrained CCA adversaries (mPR-CCCA security) if for each PPT adversary \mathcal{A} with negligible uncertainty $\text{uncert}_{\mathcal{A}}(\lambda)$, the advantage $\text{Adv}_{\text{KEM},\mathcal{A}}^{\text{mpr-ccca}}(\lambda)$ is negligible, where $\text{Adv}_{\text{KEM},\mathcal{A}}^{\text{mpr-ccca}}(\lambda) := \left| \Pr \left[\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{mpr-ccca-0}}(\lambda) = 1 \right] - \Pr \left[\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{mpr-ccca-1}}(\lambda) = 1 \right] \right|$.

Note that the afore-defined mPR-CCCA security implies multi-challenge IND-CCCA security defined in [8].

3.2 RER Security of KEM

We define Random Encapsulation Rejection security for KEM which requires the decapsulation of a random encapsulation is rejected overwhelmingly.

Definition 6 (Random Encapsulation Rejection Security for KEM).

Let $\text{KEM} = (\text{KGen}, \text{KEnc}, \text{KDec})$ be a KEM with encapsulation space Ψ and key space Γ . Let \mathcal{A} be a stateful adversary and $b \in \{0, 1\}$ be a bit. Define the following experiment $\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{rer-}b}(\lambda)$ in Fig. 3.

$\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{rer-}b}(\lambda): \quad // b \in \{0, 1\}$ $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda)$ $\psi_{\text{ran}} \leftarrow \emptyset$ $(st, 1^n) \leftarrow_{\S} \mathcal{A}^{\mathcal{O}_{\text{cha}}(\cdot, \cdot)}(\text{pk}_{\text{kem}})$ $\psi_{\text{ran}} = \{\psi_1, \dots, \psi_n\} \leftarrow_{\S} \Psi^n$ $b' \leftarrow_{\S} \mathcal{A}^{\mathcal{O}_{\text{cha}}(\cdot, \cdot)}(st, \psi_{\text{ran}})$ $\text{Return } b'$	$\mathcal{O}_{\text{cha}}(\text{pred}, \psi):$ $\text{If } \psi \notin \psi_{\text{ran}}:$ $\quad \text{Return } \text{pred}(\text{KDec}(\text{sk}_{\text{kem}}, \psi))$ $\text{If } b = 1:$ $\quad \text{Return } \text{pred}(\text{KDec}(\text{sk}_{\text{kem}}, \psi))$ Else: $\quad \text{Return } 0$
--	--

Fig. 3. Experiment used in the definition of RER property of KEM

In $\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{rer-}b}(\lambda)$, $\text{pred} : \Gamma \cup \{\perp\} \rightarrow \{0, 1\}$ denotes a PPT predicate and $\text{pred}(\perp) := 0$. Let Q_{cha} be the total number of \mathcal{O}_{cha} queries made by \mathcal{A} , which is independent of the environment without loss of generality. The uncertainty of \mathcal{A} is defined as $\text{uncert}_{\mathcal{A}}(\lambda) := \frac{1}{Q_{\text{cha}}} \sum_{i=1}^{Q_{\text{cha}}} \Pr_{\gamma \leftarrow_{\S} \Gamma} [\text{pred}_i(\gamma) = 1]$, where pred_i is the predicate in the i -th \mathcal{O}_{cha} query.

We say KEM has Random Encapsulation Rejection security (RER security) if for each PPT adversary \mathcal{A} with negligible uncertainty $\text{uncert}_{\mathcal{A}}(\lambda)$, the advantage $\text{Adv}_{\text{KEM},\mathcal{A}}^{\text{rer}}(\lambda) := \left| \Pr \left[\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{rer-0}}(\lambda) = 1 \right] - \Pr \left[\text{Exp}_{\text{KEM},\mathcal{A}}^{\text{rer-1}}(\lambda) = 1 \right] \right|$ is negligible.

4 SIM-SO-CCA Secure PKE from KEM

4.1 PKE Construction

In Fig. 4, we recall the general framework for constructing SIM-SO-CCA secure PKE proposed in [19]. A small difference from [19] is that we make use of hash function H_1 to convert the key space of KEM to the key space of XAC.

$\text{Gen}(1^\lambda)$: $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda)$ $\text{H}_1 \leftarrow_{\S} \mathcal{H}_1(1^\lambda)$ $\text{H}_2 \leftarrow_{\S} \mathcal{H}_2(1^\lambda)$ $K^x \leftarrow_{\S} \mathcal{K}_x$ $\text{pk} \leftarrow (\text{pk}_{\text{kem}}, \text{H}_1, \text{H}_2, K^x)$ $\text{sk} \leftarrow (\text{pk}, \text{sk}_{\text{kem}})$ Return (pk, sk)	$\text{Enc}(\text{pk}, \mathbf{m} \in \{0, 1\}^\ell)$: For $j \leftarrow 1$ to ℓ : If $\mathbf{m}_j = 1$: $(\psi_j, \gamma_j) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$ $K_j \leftarrow \text{H}_1(\gamma_j)$ Else: $\psi_j \leftarrow_{\S} \Psi$ $K_j \leftarrow_{\S} \mathcal{XK}$ $K^y \leftarrow \text{H}_2(\psi_1, \dots, \psi_\ell)$ $K_{\ell+1} \leftarrow (K^x, K^y)$ $T \leftarrow \text{XAuth}(K_1, \dots, K_{\ell+1})$ Return $C \leftarrow (\psi_1, \dots, \psi_\ell, T)$	$\text{Dec}(\text{sk}, C = (\psi_1, \dots, \psi_\ell, T))$: $\mathbf{m}' \leftarrow 0^\ell$ $K^{y'} \leftarrow \text{H}_2(\psi_1, \dots, \psi_\ell)$ $K'_{\ell+1} \leftarrow (K^x, K^{y'})$ If $\text{XVer}(K'_{\ell+1}, T) = 1$: For $j \leftarrow 1$ to ℓ : $\gamma'_j \leftarrow \text{KDec}(\text{sk}_{\text{kem}}, \psi_j)$ $K'_j \leftarrow \text{H}_1(\gamma'_j)$ $\mathbf{m}'_j \leftarrow \text{XVer}(K'_j, T)$ Return \mathbf{m}'
--	--	---

Fig. 4. Construction of PKE = (Gen, Enc, Dec).

Ingredients. This construction uses the following ingredients.

- KEM = (KGen, KEnc, KDec) with key space Γ & ESE encapsulation space Ψ .
- $(\ell + 1)$ -XAC XAC with ESE key space $\mathcal{XK} = \mathcal{K}_x \times \mathcal{K}_y$.
- Hash function $\text{H}_1 : \Gamma \rightarrow \mathcal{XK}$ generated by hash function generator $\mathcal{H}_1(1^\lambda)$.
- Hash function $\text{H}_2 : \Psi^\ell \rightarrow \mathcal{K}_y$ generated by hash function generator $\mathcal{H}_2(1^\lambda)$.

4.2 Tight Security Proof of PKE

In this subsection, we prove the SIM-SO-CCA security of PKE with tight reduction to the security of KEM. We state our main result in the following theorem.

Theorem 1. *Suppose the KEM is $m\text{PR-CCCA}$ and RER secure, the $(\ell + 1)$ -cross-authentication code XAC is $\delta(\lambda)$ -strong, semi-unique, and secure against impersonation and substitution attacks; \mathcal{H}_1 is universal; \mathcal{H}_2 outputs collision resistant function. Then the PKE scheme PKE constructed in Fig. 4 is SIM-SO-CCA secure. More precisely, for each PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ against PKE in the SIM-SO-CCA real experiment, for each PPT n -message sampler \mathcal{M} , and each PPT relation R , we can construct a stateful PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ for the SIM-SO-CCA ideal experiment and PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ with $\mathbf{T}(\mathcal{B}_1) \approx \mathbf{T}(\mathcal{B}_2) \approx \mathbf{T}(\mathcal{B}_3) \leq \mathbf{T}(\mathcal{A}) + Q_{\text{dec}} \cdot \text{poly}(\lambda)$, such that*

$$\begin{aligned} \text{Adv}_{\text{PKE}, \mathcal{A}, \mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca}}(\lambda) &\leq \text{Adv}_{\text{KEM}, \mathcal{B}_2}^{\text{mpr-ccca}}(\lambda) + \text{Adv}_{\text{KEM}, \mathcal{B}_3}^{\text{rer}}(\lambda) + \ell \cdot Q_{\text{dec}} \cdot \epsilon_{\text{XAC}}^{\text{sub}}(\lambda) \\ &\quad + 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + (n\ell) \cdot (\delta(\lambda) + \Delta), \end{aligned} \quad (1)$$

where Q_{dec} denotes the total number of \mathcal{A} 's decryption oracle queries, $\text{poly}(\lambda)$ is a polynomial independent of $\mathbf{T}(\mathcal{A})$ and $\Delta = \frac{1}{2} \cdot \sqrt{|\mathcal{XK}|/|\Gamma|}$.

Remark. If we instantiate the construction with the information-theoretically secure XAC in [6] and choose proper set \mathcal{XK} and Γ , then $\Delta, \delta(\lambda), \epsilon_{\text{XAC}}^{\text{imp}}(\lambda)$ and $\epsilon_{\text{XAC}}^{\text{sub}}(\lambda)$ are all exponentially small in λ . Then (1) turns out to be

$\mathcal{S}_1(1^\lambda):$ $(\text{pk}, \text{sk}) \leftarrow_{\S} \text{SimKeyGen}(1^\lambda)$ $(\alpha, a_1) \leftarrow_{\S} \mathcal{A}_1^{\text{Dec}(\cdot)}(\text{pk})$ $\text{Return } (\alpha, s_1 = (\text{pk}, \text{sk}, a_1))$ $\mathcal{S}_2(s_1, (1^{ \mathbf{m}_I })_{i \in [n]}):$ $(\mathbf{C}, \mathbf{R}, \mathbf{K}) \leftarrow_{\S} \text{SimCtGen}(\text{pk})$ $(I, a_2) \leftarrow_{\S} \mathcal{A}_2^{\text{Dec}_{\mathcal{C}(\cdot)}}(a_1, \mathbf{C})$ $\text{Return } (I, s_2 = (s_1, a_2, I, \mathbf{C}, \mathbf{R}, \mathbf{K}))$ $\mathcal{S}_3(s_2, \mathbf{m}_I):$ $\hat{\mathbf{R}}_I \leftarrow_{\S} \text{SimOpen}(I, \mathbf{m}_I, \mathbf{C}, \mathbf{R}, \mathbf{K})$ $\text{out}_{\mathcal{A}} \leftarrow_{\S} \mathcal{A}_3^{\text{Dec}_{\mathcal{C}(\cdot)}}(a_2, \mathbf{m}_I, \hat{\mathbf{R}}_I)$ $\text{Return } \text{out}_{\mathcal{A}}$	$\text{SimKeyGen}(1^\lambda):$ $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda), \text{H}_1 \leftarrow_{\S} \mathcal{H}_1(1^\lambda), \text{H}_2 \leftarrow_{\S} \mathcal{H}_2(1^\lambda), K^x \leftarrow_{\S} \mathcal{K}_x$ $\text{pk} \leftarrow (\text{pk}_{\text{kem}}, \text{H}_1, \text{H}_2, K^x); \text{sk} \leftarrow (\text{pk}, \text{sk}_{\text{kem}})$ $\text{Return } (\text{pk}, \text{sk})$	$\text{SimCtGen}(\text{pk}):$ $\text{For } i \leftarrow 1 \text{ to } n:$ $\text{For } j \leftarrow 1 \text{ to } \ell:$ $r_{i,j} \leftarrow_{\S} \mathcal{R}_{\text{KEnc}}$ $(\psi_{i,j}, \gamma_{i,j}) \leftarrow \text{KEnc}(\text{pk}_{\text{kem}}; r_{i,j})$ $K_{i,j} \leftarrow \text{H}_1(\gamma_{i,j})$ $K_i^y \leftarrow \text{H}_2(\psi_{i,1}, \dots, \psi_{i,\ell})$ $K_{i,\ell+1} \leftarrow (K^x, K_i^y)$ $T_i \leftarrow \text{XAuth}(K_{i,1}, \dots, K_{i,\ell+1})$ $\mathbf{C}_i \leftarrow (\psi_{i,1}, \dots, \psi_{i,\ell}, T_i)$ $\mathbf{R}_i \leftarrow (r_{i,1}, \dots, r_{i,\ell})$ $\mathbf{K}_i \leftarrow (K_{i,1}, \dots, K_{i,\ell+1})$ $\text{Return } \begin{pmatrix} \mathbf{C} \\ \mathbf{R} \\ \mathbf{K} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1, \dots, \mathbf{C}_n \\ \mathbf{R}_1, \dots, \mathbf{R}_n \\ \mathbf{K}_1, \dots, \mathbf{K}_n \end{pmatrix}$	$\text{SimOpen}(I, \mathbf{m}_I, \mathbf{C}, \mathbf{R}, \mathbf{K}):$ $\text{For } i \in I:$ $\text{For } j \leftarrow 1 \text{ to } \ell:$ $\text{If } \mathbf{m}_{i,j} = 1:$ $\hat{r}_{i,j} \leftarrow r_{i,j}$ Else: $\hat{K}_{i,j} \leftarrow_{\S} \text{ReSamp}(T_i, j)$ $\hat{r}_{i,j}^{\text{K}} \leftarrow_{\S} \text{Sample}_{\chi_{\text{K}}}^1(\hat{K}_{i,j})$ $\hat{r}_{i,j}^{\psi} \leftarrow_{\S} \text{Sample}_{\psi}^1(\psi_{i,j})$ $\hat{r}_{i,j} \leftarrow (\hat{r}_{i,j}^{\text{K}}, \hat{r}_{i,j}^{\psi})$ $\hat{\mathbf{R}}_i \leftarrow (\hat{r}_{i,1}, \dots, \hat{r}_{i,\ell})$ $\text{Return } \hat{\mathbf{R}}_I = (\hat{\mathbf{R}}_i)_{i \in I}$
	$\text{SimKeyGen}(1^\lambda):$ $(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda), \text{H}_1 \leftarrow_{\S} \mathcal{H}_1(1^\lambda), \text{H}_2 \leftarrow_{\S} \mathcal{H}_2(1^\lambda), K^x \leftarrow_{\S} \mathcal{K}_x$ $\text{pk} \leftarrow (\text{pk}_{\text{kem}}, \text{H}_1, \text{H}_2, K^x); \text{sk} \leftarrow (\text{pk}, \text{sk}_{\text{kem}})$ $\text{Return } (\text{pk}, \text{sk})$	$\text{SimCtGen}(\text{pk}):$ $\text{For } i \leftarrow 1 \text{ to } n:$ $\text{For } j \leftarrow 1 \text{ to } \ell:$ $r_{i,j} \leftarrow_{\S} \mathcal{R}_{\text{KEnc}}$ $(\psi_{i,j}, \gamma_{i,j}) \leftarrow \text{KEnc}(\text{pk}_{\text{kem}}; r_{i,j})$ $K_{i,j} \leftarrow \text{H}_1(\gamma_{i,j})$ $K_i^y \leftarrow \text{H}_2(\psi_{i,1}, \dots, \psi_{i,\ell})$ $K_{i,\ell+1} \leftarrow (K^x, K_i^y)$ $T_i \leftarrow \text{XAuth}(K_{i,1}, \dots, K_{i,\ell+1})$ $\mathbf{C}_i \leftarrow (\psi_{i,1}, \dots, \psi_{i,\ell}, T_i)$ $\mathbf{R}_i \leftarrow (r_{i,1}, \dots, r_{i,\ell})$ $\mathbf{K}_i \leftarrow (K_{i,1}, \dots, K_{i,\ell+1})$ $\text{Return } \begin{pmatrix} \mathbf{C} \\ \mathbf{R} \\ \mathbf{K} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1, \dots, \mathbf{C}_n \\ \mathbf{R}_1, \dots, \mathbf{R}_n \\ \mathbf{K}_1, \dots, \mathbf{K}_n \end{pmatrix}$	$\text{SimOpen}(I, \mathbf{m}_I, \mathbf{C}, \mathbf{R}, \mathbf{K}):$ $\text{For } i \in I:$ $\text{For } j \leftarrow 1 \text{ to } \ell:$ $\text{If } \mathbf{m}_{i,j} = 1:$ $\hat{r}_{i,j} \leftarrow r_{i,j}$ Else: $\hat{K}_{i,j} \leftarrow_{\S} \text{ReSamp}(T_i, j)$ $\hat{r}_{i,j}^{\text{K}} \leftarrow_{\S} \text{Sample}_{\chi_{\text{K}}}^1(\hat{K}_{i,j})$ $\hat{r}_{i,j}^{\psi} \leftarrow_{\S} \text{Sample}_{\psi}^1(\psi_{i,j})$ $\hat{r}_{i,j} \leftarrow (\hat{r}_{i,j}^{\text{K}}, \hat{r}_{i,j}^{\psi})$ $\hat{\mathbf{R}}_i \leftarrow (\hat{r}_{i,1}, \dots, \hat{r}_{i,\ell})$ $\text{Return } \hat{\mathbf{R}}_I = (\hat{\mathbf{R}}_i)_{i \in I}$

Fig. 5. Construction of simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ for $\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda)$.

$$\text{Adv}_{\text{PKE}, \mathcal{A}, \mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca}}(\lambda) \leq \text{Adv}_{\text{KEM}, \mathcal{B}_2}^{\text{mpr-ccca}}(\lambda) + \text{Adv}_{\text{KEM}, \mathcal{B}_3}^{\text{rer}}(\lambda) + 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + 2^{-\Omega(\lambda)}.$$

If the underlying KEM has tight mPR-CCCA security and RER security, then our PKE turns out to be tightly SIM-SO-CCA secure.

Proof of Theorem 1. For each PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, we can construct a stateful PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ as shown in Fig. 5.

The differences between the real and the ideal experiments lie in two aspects. The first is how the challenge ciphertext vector is generated and the second is how the corrupted ciphertexts are opened. In other words, the algorithms SimCtGen and SimOpen used by the simulator differ from the real experiment. In the proof, we focus on these two algorithms and gradually change them through a series of games starting with game G_0 and ending with game G_9 , with adjacent games being proved to be computationally indistinguishable. The full set of games are illustrated in Fig. 6.

Game G_0 . Game G_0 is exactly the ideal experiment $\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda)$. Hence

$$\Pr \left[\text{Exp}_{\mathcal{S}, n, \mathcal{M}, R}^{\text{so-cca-ideal}}(\lambda) = 1 \right] = \Pr_0[G \Rightarrow 1]. \quad (2)$$

Game $G_0 - G_1$. The only difference between G_1 and G_0 is that a collision check for H_2 is added in G_1 and G_1 aborts if a collision is found. More precisely, we use a set \mathcal{Q} to log all the (input, output) pairs for H_2 in algorithm SimCtGen . Then in the Dec oracle, if there exists a usage of H_2 such that its output collides with some output in \mathcal{Q} but with different inputs, then a collision for H_2 is found and the

<p>Exp_{S,n,M,R}^{so-cca-ideal}(λ): (pk, sk) \leftarrow_s SimKeyGen(1^λ) (α, a_1) \leftarrow_s $\mathcal{A}_1^{\text{Dec}(\cdot)}$(pk) $\mathbf{m} \leftarrow_s \mathcal{M}(\alpha)$ (C, R, K) \leftarrow_s SimCtGen(pk) (I, a_2) \leftarrow_s $\mathcal{A}_2^{\text{Dec}_{\mathcal{C}}(\cdot)}$($a_1, \mathbf{C}$) $\hat{\mathbf{R}}_I \leftarrow_s$ SimOpen(I, \mathbf{m}_I, C, R, K) $out_{\mathcal{A}} \leftarrow_s$ $\mathcal{A}_3^{\text{Dec}_{\mathcal{C}}(\cdot)}$($a_2, \mathbf{m}_I, \hat{\mathbf{R}}_I$) Return $R(\mathbf{m}, I, out_{\mathcal{A}})$</p> <hr/> <p>SimCtGen(pk): G_0 $[G_1, G_2]$ $[G_3, [G_4 - G_7]]$ $[G_8 : G_9]$ For $i \leftarrow 1$ to n: For $j \leftarrow 1$ to ℓ: If $\mathbf{m}_{i,j} = 0$: $r_{i,j}^\psi \leftarrow_s \mathcal{R}_{\text{Sample}_\psi}$ $\psi_{i,j} \leftarrow \text{Sample}_\psi(1^\lambda; r_{i,j}^\psi)$ $\gamma_{i,j} \leftarrow_s \Gamma$ $K_{i,j} \leftarrow H_1(\gamma_{i,j})$ $[r_{i,j}^K \leftarrow_s \mathcal{R}_{\text{Sample}_{\mathcal{X}\mathcal{K}}}]$ $[K_{i,j} \leftarrow \text{Sample}_{\mathcal{X}\mathcal{K}}(1^\lambda; r_{i,j}^K)]$ $[r_{i,j} \leftarrow (r_{i,j}^K, r_{i,j}^\psi)]$ Else: $r_{i,j} \leftarrow_s \mathcal{R}_{\text{KEnc}}$ $(\psi_{i,j}, \gamma_{i,j}) \leftarrow \text{KEnc}(\text{pk}_{\text{kem}}; r_{i,j})$ $K_{i,j} \leftarrow H_1(\gamma_{i,j})$ $K_i^y \leftarrow H_2(\psi_{i,1}, \dots, \psi_{i,\ell})$ $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(K_i^y, (\psi_{i,1}, \dots, \psi_{i,\ell}))\}$ $K_{i,\ell+1} \leftarrow (K^x, K_i^y)$ $T_i \leftarrow \text{XAuth}(K_{i,1}, \dots, K_{i,\ell+1})$ $\mathbf{C}_i \leftarrow (\psi_{i,1}, \dots, \psi_{i,\ell}, T_i)$ $\mathbf{R}_i \leftarrow (r_{i,1}, \dots, r_{i,\ell})$ $\mathbf{K}_i \leftarrow (K_{i,1}, \dots, K_{i,\ell+1})$ Return $\begin{pmatrix} \mathbf{C} \\ \mathbf{R} \\ \mathbf{K} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1, \dots, \mathbf{C}_n \\ \mathbf{R}_1, \dots, \mathbf{R}_n \\ \mathbf{K}_1, \dots, \mathbf{K}_n \end{pmatrix}$</p>	<p>SimOpen(I, \mathbf{m}_I, C, R, K): $[G_0 - G_6]$ $[G_7, G_8]$ G_9 For $i \in I$: For $j \leftarrow 1$ to ℓ: If $\mathbf{m}_{i,j} = 1$: $\hat{r}_{i,j} \leftarrow r_{i,j}$ Else: $[K_{i,j} \leftarrow_s \text{ReSamp}(T_i, j)]$ $[r_{i,j}^K \leftarrow_s \text{Sample}_{\mathcal{X}\mathcal{K}}^{-1}(K_{i,j})]$ $[r_{i,j}^K \leftarrow_s \text{Sample}_{\mathcal{X}\mathcal{K}}^{-1}(K_{i,j})]$ $\hat{r}_{i,j}^\psi \leftarrow_s \text{Sample}_\psi^{-1}(\psi_{i,j})$ $\hat{r}_{i,j} \leftarrow (\hat{r}_{i,j}^K, \hat{r}_{i,j}^\psi)$ $\hat{\mathbf{R}}_i \leftarrow (\hat{r}_{i,1}, \dots, \hat{r}_{i,\ell})$ $\hat{\mathbf{R}}_I \leftarrow \mathbf{R}_I$ Return $\hat{\mathbf{R}}_I$</p> <hr/> <p>Dec_C(C = ($\psi_1, \dots, \psi_\ell, T$)): G_0 $[G_1 : G_2, G_3, G_4 : G_5]$ $[G_6, G_7]$ G_8, G_9 If $C \in \mathbf{C}$: Return \perp $\mathbf{m} \leftarrow 0^\ell$ $K^{y'} \leftarrow H_2(\psi_1, \dots, \psi_\ell)$ If $[\exists (\hat{K}^y, (\hat{\psi}_1, \dots, \hat{\psi}_\ell)) \in \mathcal{Q} \text{ s.t. } K^{y'} = \hat{K}^y \wedge (\psi_1, \dots, \psi_\ell) \neq (\hat{\psi}_1, \dots, \hat{\psi}_\ell)]$: Abort game //Find a collision for H_2 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(K^{y'}, (\psi_1, \dots, \psi_\ell))\}$ $K_{\ell+1}^x \leftarrow (K^x, K^{y'})$ If $\text{XVer}(K_{\ell+1}^x, T) = 1$: For $\eta \leftarrow 1$ to ℓ: $\gamma'_\eta \leftarrow \text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)$ If $[\exists (i, j) \in [n] \times [\ell] \text{ s.t. } \mathbf{m}_{i,j} = 0 \wedge \psi_\eta = \psi_{i,j}]$: $\mathbf{m}'_\eta \leftarrow \text{XVer}(H_1(\gamma'_\eta), T)$ $[\mathbf{m}'_\eta \leftarrow \text{XVer}(K_{i,j}, T)]$ $[\mathbf{m}'_\eta \leftarrow 0]$ Else: $\mathbf{m}'_\eta \leftarrow \text{XVer}(H_1(\gamma'_\eta), T)$ Return \mathbf{m}'</p>
<p>SimKeyGen(1^λ): G_0 $[G_1 - G_7]$ G_8, G_9 (pk_{kem}, sk_{kem}) \leftarrow_s KGen(1^λ), $H_1 \leftarrow_s \mathcal{H}_1(1^\lambda)$, $H_2 \leftarrow_s \mathcal{H}_2(1^\lambda)$, $K^x \leftarrow_s \mathcal{K}_x$ pk \leftarrow (pk_{kem}, H_1, H_2, K^x), sk \leftarrow (pk, sk_{kem}) $[T \leftarrow \emptyset]$ Return (pk, sk)</p>	

Fig. 6. Games $G_0 - G_9$ in the proof of Theorem 1.

game G_1 aborts immediately. It is straightforward to build a PPT adversary \mathcal{B}_1 with $\mathbf{T}(\mathcal{B}_1) \approx \mathbf{T}(\mathcal{A}) + Q_{\text{dec}} \cdot \text{poly}(\lambda)$, where $\text{poly}(\lambda)$ is a polynomial independent of $\mathbf{T}(\mathcal{A})$, such that,

$$|\Pr_0[G \Rightarrow 1] - \Pr_1[G \Rightarrow 1]| \leq \text{Adv}_{\mathcal{H}_1, \mathcal{B}_1}^{\text{cr}}(\lambda). \quad (3)$$

Game $G_1 - G_2$. G_2 is essentially the same as G_1 except for one conceptual change in the Dec oracle. More precisely, for a $\text{Dec}(C = (\psi_1, \dots, \psi_\ell, T))$ query such that $\exists(i, j) \in [n] \times [\ell], \eta \in [\ell]$ s.t. $\mathbf{m}_{i,j} = 0 \wedge \psi_\eta = \psi_{i,j}$,

- in G_1 , we proceed exactly the same as the decryption algorithm, i.e.,

$$\text{set } \mathbf{m}'_\eta \leftarrow \text{XVer}(\text{H}_1(\gamma'_\eta), T) \text{ where } \gamma'_\eta = \text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta);$$

- in G_2 , we set $\mathbf{m}'_\eta \leftarrow \text{XVer}(K_{i,j}, T)$.

Since $\psi_\eta = \psi_{i,j}$, $\gamma'_\eta = \text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)$ and $(\psi_{i,j}, \gamma_{i,j})$ is the output of $\text{KEnc}(\text{pk}_{\text{kem}})$, we have that $\gamma'_\eta = \gamma_{i,j}$ due to the perfect correctness of KEM. Then $K_{i,j} = \text{H}_1(\gamma_{i,j}) = \text{H}_1(\gamma'_\eta)$. Thus the difference between G_1 and G_2 is only conceptual, and it follows

$$\Pr_1[G \Rightarrow 1] = \Pr_2[G \Rightarrow 1]. \quad (4)$$

Game $G_2 - G_3$. G_3 is almost the same as G_2 except for one change in the SimCtGen algorithm.

- In G_2 , all $(\psi_{i,j}, \gamma_{i,j})$ pairs are the output of $\text{KEnc}(\text{pk}_{\text{kem}})$.
- In G_3 , for $\mathbf{m}_{i,j} = 1$, $(\psi_{i,j}, \gamma_{i,j})$ pairs are the output of $\text{KEnc}(\text{pk}_{\text{kem}})$;
for $\mathbf{m}_{i,j} = 0$, $(\psi_{i,j}, \gamma_{i,j})$ pairs are uniformly selected from $\Psi \times \Gamma$.

We will reduce the indistinguishability between game G_2 and G_3 to the mPR-CCCA security of KEM. Given $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, we can build a PPT adversary \mathcal{B}_2 with $\mathbf{T}(\mathcal{B}_2) \approx \mathbf{T}(\mathcal{A})$ and uncertainty $\text{uncert}_{\mathcal{B}_2}(\lambda) \leq \epsilon_{\text{XAC}}^{\text{imp}}(\lambda) + \Delta$ such that

$$|\Pr_2[G \Rightarrow 1] - \Pr_3[G \Rightarrow 1]| \leq \text{Adv}_{\text{KEM}, \mathcal{B}_2}^{\text{mpr-ccca}}(\lambda). \quad (5)$$

On input pk_{kem} , \mathcal{B}_2 selects H_1, H_2 and K^x itself and embeds pk_{kem} in $\text{pk} = (\text{pk}_{\text{kem}}, \text{H}_1, \text{H}_2, K^x)$. In the first phase, \mathcal{B}_2 calls $\mathcal{A}_1^{\text{Dec}(\cdot)}(\text{pk})$. To respond the decryption query $\text{Dec}(C = (\psi_1, \dots, \psi_\ell, T))$ submitted by \mathcal{A} , \mathcal{B}_2 simulates Dec until it needs to call $\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)$ to decapsulate ψ_η . Since \mathcal{B}_2 does not possess sk_{kem} relative to pk_{kem} , \mathcal{B}_2 is not able to invoke KDec itself. Then \mathcal{B}_2 submits a $\mathcal{O}_{\text{dec}}(\text{pred}, \psi_\eta)$ query to its own oracle \mathcal{O}_{dec} where $\text{pred}(\cdot) := \text{XVer}(\text{H}_1(\cdot), T)$. Clearly, this predicate is a PPT one. If the response of \mathcal{O}_{dec} is \perp , \mathcal{B}_2 sets \mathbf{m}'_η to 0. Otherwise \mathcal{B}_2 sets \mathbf{m}'_η to 1.

Case 1: $\mathcal{O}_{\text{dec}}(\text{XVer}(\text{H}_1(\cdot), T), \psi_\eta) = \perp$. This happens if and only if

$$\psi_\eta \in \psi_{\text{enc}} \vee \text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T) = 0.$$

In the first phase, \mathcal{B}_2 has not submitted any \mathcal{O}_{enc} query yet and ψ_{enc} is empty. So $\psi_\eta \notin \psi_{\text{enc}}$. In this case, $\mathcal{O}_{\text{dec}}(\text{XVer}(\text{H}_1(\cdot), T), \psi_\eta) = \perp$ if and only if

$$\text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T) = 0.$$

Therefore \mathcal{B}_2 perfectly simulates the Dec oracle in $G_2(G_3)$ by setting $\mathbf{m}'_\eta \leftarrow 0$.

Case 2: $\mathcal{O}_{\text{dec}}(\text{XVer}(\text{H}_1(\cdot), T), \psi_\eta) \neq \perp$. This happens if and only if

$$\psi_\eta \notin \psi_{\text{enc}} \wedge \text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T) = 1.$$

For the same reason as case 1, the condition $\psi_\eta \notin \psi_{\text{enc}}$ always holds. In this case, $\mathcal{O}_{\text{dec}}(\text{XVer}(\text{H}_1(\cdot), T), \psi_\eta) \neq \perp$ if and only if $\text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T) = 1$. Therefore \mathcal{B}_2 perfectly simulates the Dec oracle in $G_2(G_3)$ by setting $\mathbf{m}'_\eta \leftarrow 1$.

In either case, \mathcal{B}_2 can perfectly simulate the Dec oracle for \mathcal{A}_1 . At the end of this phase, \mathcal{B}_2 gets \mathcal{A}_1 's output (α, a_1) . Then \mathcal{B}_2 calls $\mathbf{m} \leftarrow_{\mathfrak{s}} \mathcal{M}(\alpha)$ and simulates algorithm $\text{SimCtGen}(\text{pk})$.

- If $\mathbf{m}_{i,j} = 1$, \mathcal{B}_2 proceeds just like game $G_2(G_3)$, i.e., $(\psi_{i,j}, \gamma_{i,j}) \leftarrow_{\mathfrak{s}} \text{KEnc}(\text{pk}_{\text{kem}})$ and set $K_{i,j} \leftarrow \text{H}_1(\gamma_{i,j})$.
- If $\mathbf{m}_{i,j} = 0$, \mathcal{B}_2 submits an $\mathcal{O}_{\text{enc}}()$ query to its own oracle and gets the response (ψ, γ) (ψ is added into set ψ_{enc}). Then \mathcal{B}_2 sets $(\psi_{i,j}, \gamma_{i,j}) \leftarrow (\psi, \gamma)$.
If $b = 1$, (ψ, γ) is the output of $\text{KEnc}(\text{pk}_{\text{kem}})$, \mathcal{B}_2 perfectly simulates $\text{SimCtGen}(\text{pk})$ to generate challenge ciphertexts \mathbf{C} in G_2 .
If $b = 0$, (ψ, γ) is uniformly over $\Psi \times \Gamma$, \mathcal{B}_2 perfectly simulates $\text{SimCtGen}(\text{pk})$ to generate challenge ciphertexts \mathbf{C} in G_3 .

In the second phase, \mathcal{B}_2 calls $\mathcal{A}_2^{\text{Dec}_{\neq \mathbf{C}}(\cdot)}(a_1, \mathbf{C})$ to get (I, a_2) . Upon an decryption query $\text{Dec}_{\neq \mathbf{C}}(C = (\psi_1, \dots, \psi_\ell, T))$ submitted by \mathcal{A}_2 , \mathcal{B}_2 responds almost in the same way as in the first phase, except that \mathcal{B}_2 has to deal with the case of $\exists \psi_\eta \in \psi_{\text{enc}}$. This case does happen: even if $C = (\psi_1, \dots, \psi_\ell, T) \notin \mathbf{C}$, it is still possible that $\exists \psi_\eta \in \{\psi_i\}_{i \in [\ell]}$ with $\psi_\eta \in \psi_{\text{enc}}$. In this case, there is no chance for \mathcal{B}_2 to submit an $\mathcal{O}_{\text{dec}}(\text{pred}, \psi_\eta)$ query for a useful response because the response will always be \perp . However, it does not matter. By the specification of $G_2(G_3)$, \mathbf{m}'_η should be set to the output of $\text{XVer}(K_{i,j}, T)$ which \mathcal{B}_2 can perfectly do.

Note that the execution of algorithm SimOpen in game $G_2(G_3)$ does not need all information about \mathbf{R} . Only those randomnesses with respect to $\mathbf{m}_{i,j} = 1$ are needed. Now that \mathcal{B}_2 does have $I, \mathbf{m}_I, \mathbf{C}, \mathbf{K}$ and part of \mathbf{R} (for $\mathbf{m}_{i,j} = 1$), it can call $\text{SimOpen}(I, \mathbf{m}_I, \mathbf{C}, \mathbf{R}, \mathbf{K})$ to get $\hat{\mathbf{R}}_I$.

In the third phase, \mathcal{B}_2 calls $\mathcal{A}_3^{\text{Dec}_{\neq \mathbf{C}}(\cdot)}(a_2, \mathbf{m}_I, \hat{\mathbf{R}}_I)$ to get $\text{out}_{\mathcal{A}}$. The $\text{Dec}_{\neq \mathbf{C}}$ query submitted by \mathcal{A} in this phase is responded by \mathcal{B}_2 in the same way as in the second phase. Finally, \mathcal{B}_2 outputs $R(\mathbf{m}, I, \text{out}_{\mathcal{A}})$.

According to the above analysis, \mathcal{B}_2 perfectly simulates G_2 for \mathcal{A} if $b = 1$ and perfectly simulates G_3 for \mathcal{A} if $b = 0$. Moreover, for $\gamma \leftarrow_{\mathfrak{s}} \Gamma$, $\text{H}_1(\gamma)$ is Δ -close to uniform by leftover hash lemma since H_1 is universal. Then

$$\Pr_{\gamma \leftarrow_{\mathfrak{s}} \Gamma} [\text{pred}(\gamma) = 1] = \Pr_{\gamma \leftarrow_{\mathfrak{s}} \Gamma} [\text{XVer}(\text{H}_1(\gamma), T) = 1] \leq \epsilon_{\text{XAC}}^{\text{imp}}(\lambda) + \Delta.$$

By the definition of uncertainty, we have.

$$\text{uncert}_{\mathcal{B}_2}(\lambda) \leq \epsilon_{\text{XAC}}^{\text{imp}}(\lambda) + \Delta. \quad (6)$$

Thus (5) follows.

Game $G_3 - G_4$. G_4 is almost the same as G_3 except for one change in the SimCtGen algorithm. In the SimCtGen algorithm, if $\mathbf{m}_{i,j} = 0$,

- in G_3 , $K_{i,j} \leftarrow \mathbf{H}_1(\gamma_{i,j})$ for $\gamma_{i,j} \leftarrow_{\S} \Gamma$;
- in G_4 , $K_{i,j}$ is uniformly selected from \mathcal{XK} .

Since \mathbf{H}_1 is universal, by leftover hash lemma and a union bound, we have that

$$|\Pr_3[G \Rightarrow 1] - \Pr_4[G \Rightarrow 1]| \leq (n\ell) \cdot \Delta. \quad (7)$$

Game $G_4 - G_5$. G_5 is almost the same as G_4 except for one change in the Dec oracle. More precisely, to reply a $\text{Dec}_{\notin \mathbf{C}}(C = (\psi_1, \dots, \psi_\ell, T))$ query such that $\exists(i, j) \in [n] \times [\ell], \eta \in [\ell]$ s.t. $\mathbf{m}_{i,j} = 0 \wedge \psi_\eta = \psi_{i,j}$,

- in G_4 , we set $\mathbf{m}'_\eta \leftarrow \text{XVer}(K_{i,j}, T)$;
- in G_5 , we set $\mathbf{m}'_\eta \leftarrow 0$ directly.

Suppose $\psi_\eta = \psi_{i,j} \in \mathbf{C}_i = (\psi_{i,1}, \dots, \psi_{i,\ell}, T_i)$ where $T_i = \text{XAuth}(K_{i,1}, \dots, K_{i,\ell+1})$. There are two cases according to whether $T = T_i$.

Case 1: $T = T_i$. In this case, since $C \notin \mathbf{C}$, we have that $(\psi_1, \dots, \psi_\ell) \neq (\psi_{i,1}, \dots, \psi_{i,\ell})$. Note that $K_i^y = \mathbf{H}_2(\psi_{i,1}, \dots, \psi_{i,\ell})$ and $K^{y'} = \mathbf{H}_2(\psi_1, \dots, \psi_\ell)$. If $K_i^y = K^{y'}$, a collision for \mathbf{H}_2 occurs, both G_4 and G_5 abort. Otherwise, we must have $K^{y'} \neq K_i^y$, hence $K'_{\ell+1} = (K^x, K^{y'}) \neq (K^x, K_i^y) = K_{i,\ell+1}$. Since XAC is semi-unique and $\text{XVer}(K_{i,\ell+1}, T) = 1$, it holds that $\text{XVer}(K'_{\ell+1}, T) \neq 1$ which implies that $\mathbf{m}'_\eta = 0$. In this case, the responses of $\text{Dec}_{\notin \mathbf{C}}$ make no difference in G_4 and G_5 .

Case 2: $T \neq T_i$. Note that all the information about $K_{i,j}$ is leaked to \mathcal{A} only through T_i in game G_4 . Thus, the probability that $\text{XVer}(K_{i,j}, T) = 1$ for $T \neq T_i$ will be no more than $\epsilon_{\text{XAC}}^{\text{sub}}(\lambda)$.

By a union bound, we have that

$$|\Pr_4[G \Rightarrow 1] - \Pr_5[G \Rightarrow 1]| \leq \ell \cdot Q_{\text{dec}} \cdot \epsilon_{\text{XAC}}^{\text{sub}}(\lambda). \quad (8)$$

Game $G_5 - G_6$. G_6 is almost the same as G_5 except for one change in the Dec oracle. More precisely, for a $\text{Dec}(C = (\psi_1, \dots, \psi_\ell, T))$ query such that $\exists(i, j) \in [n] \times [\ell]$ s.t. $\mathbf{m}_{i,j} = 0 \wedge \psi_\eta = \psi_{i,j}$ for any $\eta \in [\ell]$,

- in G_5 , we set $\mathbf{m}'_\eta \leftarrow 0$ directly;
- in G_6 , we proceed exactly the same as the decryption algorithm, i.e., setting $\mathbf{m}'_\eta \leftarrow \text{XVer}(\mathbf{H}_1(\gamma'_\eta), T)$, where $\gamma'_\eta = \text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)$.

We will reduce the indistinguishability between game G_5 and G_6 to the RER security of KEM. More precisely, we can build a PPT adversary \mathcal{B}_3 with $\mathbf{T}(\mathcal{B}_3) \approx \mathbf{T}(\mathcal{A})$ and with uncertainty $\text{uncert}_{\mathcal{B}_3}(\lambda) \leq \epsilon_{\text{XAC}}^{\text{imp}}(\lambda) + \Delta$ such that

$$|\Pr_5[G \Rightarrow 1] - \Pr_6[G \Rightarrow 1]| \leq \text{Adv}_{\text{KEM}, \mathcal{B}_3}^{\text{rer}}(\lambda). \quad (9)$$

On input pk_{kem} , \mathcal{B}_3 selects $\mathbf{H}_1, \mathbf{H}_2$ and K^x itself and embeds pk_{kem} in $\text{pk} = (\text{pk}_{\text{kem}}, \mathbf{H}_1, \mathbf{H}_2, K^x)$. In the first phase, \mathcal{B}_3 calls $\mathcal{A}_1^{\text{Dec}(\cdot)}(\text{pk})$. To respond the

decryption query $\text{Dec}(C = (\psi_1, \dots, \psi_\ell, T))$ submitted by \mathcal{A} , \mathcal{B}_3 simulates Dec until it needs to call $\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)$ to decapsulate ψ_η . Since \mathcal{B}_3 does not hold sk_{kem} relative to pk_{kem} , \mathcal{B}_3 is not able to invoke KDec itself. Then \mathcal{B}_3 submits a $\mathcal{O}_{\text{cha}}(\text{pred}, \psi)$ query to its own oracle \mathcal{O}_{cha} where $\text{pred}(\cdot) := \text{XVer}(\text{H}_1(\cdot), T)$ and $\psi = \psi_\eta$. Clearly, this predicate is a PPT one. Since ψ_{ran} is empty set in this phase, the condition $\psi \notin \psi_{\text{ran}}$ will always hold and \mathcal{B}_3 will get a bit $\beta = \text{pred}(\text{KDec}(\text{sk}_{\text{kem}}, \psi)) = \text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T)$ in return. Then \mathcal{B}_3 sets $\mathbf{m}'_\eta \leftarrow \beta$ and perfectly simulates Dec for \mathcal{A} in this phase.

At the end of this phase, \mathcal{B}_3 gets \mathcal{A} 's output (α, a_1) . Then \mathcal{B}_3 calls $\mathbf{m} \leftarrow_{\S} \mathcal{M}(\alpha)$ and then simulates algorithm $\text{SimCtGen}(\text{pk})$ as follows. \mathcal{B}_3 first outputs $1^{n\ell}$ and get $\psi_{\text{ran}} = \{\psi_1^{\text{ran}}, \dots, \psi_{n\ell}^{\text{ran}}\}$ which are $n\ell$ random encapsulations. During the generation of the challenge ciphertexts, \mathcal{B}_3 sets $(\psi_{i,j}, K_{i,j})$ according to \mathbf{m} .

- If $\mathbf{m}_{i,j} = 1$, \mathcal{B}_3 sets $(\psi_{i,j}, \gamma_{i,j}) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}})$ and sets $K_{i,j} \leftarrow \text{H}_1(\gamma_{i,j})$.
- If $\mathbf{m}_{i,j} = 0$, \mathcal{B}_3 sets $\psi_{i,j} \leftarrow \psi_{(i-1)\ell+j}^{\text{ran}}$ and $K_{i,j} \leftarrow_{\S} \mathcal{K}\mathcal{K}$. Since $(i, j) \in [n] \times [\ell]$, the subscript $(i-1)\ell+j \in \{1, \dots, n\ell\}$ is well defined.

Then \mathcal{B}_3 proceeds just like algorithm $\text{SimCtGen}(\text{pk})$ in game $G_5(G_6)$.

In the second phase, \mathcal{B}_3 calls $\mathcal{A}_2^{\text{Dec}_{\neq \mathbf{C}}(\cdot)}(a_1, \mathbf{C})$ to get (I, a_2) . To respond the decryption query $\text{Dec}_{\neq \mathbf{C}}(C = (\psi_1, \dots, \psi_\ell, T))$ submitted by \mathcal{A} , \mathcal{B}_3 proceeds just like game $G_5(G_6)$. When a decapsulation of ψ_η is needed, \mathcal{B}_3 submits a $\mathcal{O}_{\text{cha}}(\text{pred}, \psi_\eta)$ query to its own oracle \mathcal{O}_{cha} where $\text{pred}(\cdot) := \text{XVer}(\text{H}_1(\cdot), T)$. After that, \mathcal{B}_3 will get a bit β in return and \mathcal{B}_3 sets $\mathbf{m}'_\eta \leftarrow \beta$. Note that

- In case of $\psi_\eta \notin \psi_{\text{ran}}$, $\mathbf{m}'_\eta = \text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T)$, which is exactly how \mathbf{m}'_η is computed in both game G_5 and G_6 .
- In case of $\psi_\eta \in \psi_{\text{ran}}$, there must exist $(i, j) \in [n] \times [\ell]$ s.t. $\mathbf{m}_{i,j} = 0 \wedge \psi_\eta = \psi_{i,j}$. Thus $\mathbf{m}'_\eta = \text{XVer}(\text{H}_1(\text{KDec}(\text{sk}_{\text{kem}}, \psi_\eta)), T)$ if $b = 1$ and $\mathbf{m}'_\eta = 0$ if $b = 0$. The former case is exactly how \mathbf{m}'_η is computed in game G_6 and the latter case is exactly how \mathbf{m}'_η is computed in game G_5 .

As a result, \mathcal{B}_3 perfectly simulates $\text{Dec}_{\neq \mathbf{C}}$ in the second phase of game G_5 for \mathcal{A} if $b = 0$ and perfectly simulates $\text{Dec}_{\neq \mathbf{C}}$ in the second phase of game G_6 for \mathcal{A} if $b = 1$. After \mathcal{B}_3 gets (I, a_2) , \mathcal{B}_3 is able to call $\text{SimOpen}(I, \mathbf{m}_I, \mathbf{C}, \mathbf{R}, \mathbf{K})$ to get $\hat{\mathbf{R}}_I$ for the similar reason as in the proof of $G_2 - G_3$.

In the third phase, \mathcal{B}_3 calls $\mathcal{A}_3^{\text{Dec}_{\neq \mathbf{C}}(\cdot)}(a_2, \mathbf{m}_I, \hat{\mathbf{R}}_I)$ to get $\text{out}_{\mathcal{A}}$. The $\text{Dec}_{\neq \mathbf{C}}$ query submitted by \mathcal{A} in this phase is responded using the same way as in the second phase. Finally, \mathcal{B}_3 outputs $R(\mathbf{m}, I, \text{out}_{\mathcal{A}})$.

Thus \mathcal{B}_3 perfectly simulates G_6 for \mathcal{A} if $b = 1$ and perfectly simulates G_5 for \mathcal{A} if $b = 0$. Similar to (6), $\text{uncert}_{\mathcal{B}_3}(\lambda) \leq \epsilon_{\text{XAC}}^{\text{imp}}(\lambda) + \Delta$. Thus (9) follows.

Game $G_6 - G_7$. G_7 is almost the same as G_6 except for one change in the SimOpen algorithm. More precisely,

- in G_6 , $\hat{r}_{i,j}^K$ is the output of $\text{Sample}_{\mathcal{X}\mathcal{K}}^{-1}(\hat{K}_{i,j})$ where $\hat{K}_{i,j} \leftarrow_{\S} \text{ReSamp}(T_i, j)$;
- in G_7 , $\hat{r}_{i,j}^K$ is the output of $\text{Sample}_{\mathcal{X}\mathcal{K}}^{-1}(K_{i,j})$ for the original $K_{i,j}$ generated in algorithm SimCtGen .

In game G_6 and G_7 , before the invocation of algorithm `SimOpen`, only T_i leaks information about $K_{i,j}$ to \mathcal{A} when $\mathbf{m}_{i,j} = 0$. Since `XAC` is $\delta(\lambda)$ -strong, the statistical distance between the resampled $\hat{K}_{i,j} \leftarrow_{\S} \text{ReSamp}(T_i, j)$ and the original $K_{i,j}$ is at most $\delta(\lambda)$. By a union bound, we have that

$$|\Pr_6[G \Rightarrow 1] - \Pr_7[G \Rightarrow 1]| \leq (n\ell) \cdot \delta(\lambda). \quad (10)$$

Game $G_7 - G_8$. G_8 is almost the same as G_7 except for the dropping of the collision check added in G_1 . Similar to the proof of $G_0 - G_1$, we can show that

$$|\Pr_7[G \Rightarrow 1] - \Pr_8[G \Rightarrow 1]| \leq \text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda). \quad (11)$$

Game $G_8 - G_9$. G_9 is almost the same as G_8 except for one change in `SimOpen`. More precisely,

- in G_8 , the opened randomness is a “reverse sampled” randomness, i.e., $\hat{r}_{i,j}^K \leftarrow_{\S} \text{Sample}_{\mathcal{XK}}^{-1}(K_{i,j})$ and $\hat{r}_{i,j}^\psi \leftarrow_{\S} \text{Sample}_{\Psi}^{-1}(\psi_{i,j})$;
- in G_9 , the opened randomness $(\hat{r}_{i,j}^K, \hat{r}_{i,j}^\psi)$ is changed to be the original randomness used to sample $K_{i,j}$ and $\psi_{i,j}$, i.e., $(\hat{r}_{i,j}^K, \hat{r}_{i,j}^\psi) \leftarrow (r_{i,j}^K, r_{i,j}^\psi)$.

This change is conceptual since Ψ and \mathcal{XK} are ESE domains. Thus

$$\Pr_8[G \Rightarrow 1] = \Pr_9[G \Rightarrow 1]. \quad (12)$$

Game G_9 . Game G_9 is exactly the real experiment $\text{Exp}_{\text{PKE}, \mathcal{A}, n, \mathcal{M}, R}^{\text{so-cca-real}}(\lambda)$. Thus

$$\Pr_9[G \Rightarrow 1] = \Pr \left[\text{Exp}_{\text{PKE}, \mathcal{A}, n, \mathcal{M}, R}^{\text{so-cca-real}}(\lambda) = 1 \right]. \quad (13)$$

Finally, Theorem 1 follows from (2, 3, 4, 5, 7, 8, 9, 10, 11, 12) and (13). ■

5 Instantiations

We give two instantiations of KEM with mPR-CCCA security and RER security.

5.1 KEM from MDDH

We present a KEM which is extracted from the multi-challenge IND-CCA secure PKE proposed by Gay *et al.* in [7]. The KEM $\text{KEM}_{\text{mddh}} = (\text{KGen}, \text{KEnc}, \text{KDec})$ is shown in Fig. 7.

Suppose $\mathcal{G} = (\mathbb{G}, q, P) \leftarrow_{\S} \text{GGen}(1^\lambda)$ and \mathcal{H} is a hash generator outputting functions $\text{H} : \mathbb{G}^k \rightarrow \{0, 1\}^\lambda$. For a vector $\mathbf{y} \in \mathbb{Z}_q^{3k}$, we use $\bar{\mathbf{y}} \in \mathbb{Z}_q^k$ to denote the upper k components and $\underline{\mathbf{y}} \in \mathbb{Z}_q^{2k}$ to denote the lower $2k$ components.

Perfectly correctness of KEM_{mddh} is straightforward. See the full version [21] for the proofs of its tight mPR-CCCA security and tight RER security.

$\text{KGen}(1^\lambda) :$ $\mathbf{M} \leftarrow_{\S} \mathcal{U}_{3k,k}, \mathbf{H} \leftarrow_{\S} \mathcal{H}(1^\lambda).$ $\mathbf{k}_{1,0}, \dots, \mathbf{k}_{\lambda,1} \leftarrow_{\S} \mathbb{Z}_q^{3k}$ $\text{pk}_{\text{kem}} \leftarrow \left(\begin{array}{c} \mathcal{G}, \mathbf{H}, [\mathbf{M}] \\ ([\mathbf{M}^\top \mathbf{k}_{j,\beta}]_{\substack{0 \leq \beta \leq 1 \\ 1 \leq j \leq \lambda}}) \end{array} \right)$ $\text{sk}_{\text{kem}} \leftarrow (\mathbf{k}_{j,\beta})_{1 \leq j \leq \lambda, 0 \leq \beta \leq 1}$ $\text{Return } (\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}})$	$\text{KEnc}(\text{pk}_{\text{kem}}) :$ $\mathbf{r} \leftarrow_{\S} \mathbb{Z}_q^k, [\mathbf{y}] \leftarrow [\mathbf{M}]\mathbf{r}$ $\tau \leftarrow \mathbf{H}([\bar{\mathbf{y}}])$ $\gamma \leftarrow \mathbf{r}^\top \cdot \sum_{j=1}^{\lambda} [\mathbf{M}^\top \mathbf{k}_{j,\tau_j}]$ $\text{Return } (\psi \leftarrow [\mathbf{y}], \gamma)$ $// \Psi = \mathbb{G}^{3k}, \Gamma = \mathbb{G}$	$\text{KDec}(\text{sk}_{\text{kem}}, \psi) :$ $\psi = [\mathbf{y}]$ $\tau \leftarrow \mathbf{H}([\bar{\mathbf{y}}])$ $\mathbf{k}_\tau \leftarrow \sum_{j=1}^{\lambda} \mathbf{k}_{j,\tau_j}$ $\gamma \leftarrow [\mathbf{y}^\top] \cdot \mathbf{k}_\tau$ $\text{Return } \gamma$
--	--	--

Fig. 7. The KEM $\text{KEM}_{\text{mddh}} = (\text{KGen}, \text{KEnc}, \text{KDec})$ extracted from [7].

5.2 KEM from Qualified Proof System with Compact Public Key

First we recall the definition of a *proof system* described in [8].

Definition 7 (Proof System). Let $\mathcal{L} = \{\mathcal{L}_{\text{pars}}\}$ be a family of languages indexed by public parameters pars , with $\mathcal{L}_{\text{pars}} \subseteq \mathcal{X}_{\text{pars}}$ and an efficiently computable witness relation \mathcal{R} . A proof system $\text{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$ for \mathcal{L} consists of a tuple of PPT algorithms.

- $\text{PGen}(\text{pars})$. It outputs a public key ppk and a secret key psk .
- $\text{PPrv}(\text{ppk}, x, w)$. On input a statement $x \in \mathcal{L}$ and a witness w with $\mathcal{R}(x, w) = 1$, it deterministically outputs a proof $\Pi \in \Pi$ and a key $K \in \mathcal{K}$.
- $\text{PVer}(\text{ppk}, \text{psk}, x, \Pi)$. On input $\text{ppk}, \text{psk}, x \in \mathcal{X}$ and Π , it deterministically outputs $b \in \{0, 1\}$ together with a key $K \in \mathcal{K}$ if $b = 1$ or \perp if $b = 0$.
- $\text{PSim}(\text{ppk}, \text{psk}, x)$. Given $\text{ppk}, \text{psk}, x \in \mathcal{X}$, it deterministically outputs a proof Π and a key $K \in \mathcal{K}$.

Next we recall the definition of a qualified proof system.

Definition 8 (Qualified Proof System [8]). Let $\text{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$ be a proof system for a family of languages $\mathcal{L} = \mathcal{L}_{\text{pars}}$. Let $\mathcal{L}^{\text{snd}} = \{\mathcal{L}_{\text{pars}}^{\text{snd}}\}$ be a family of languages, such that $\mathcal{L}_{\text{pars}} \subseteq \mathcal{L}_{\text{pars}}^{\text{snd}}$. We say that PS is \mathcal{L}^{snd} -qualified, if the following properties hold.

- **Completeness:** For all possible public parameters pars , for all statements $x \in \mathcal{L}$ and all witnesses w such that $\mathcal{R}(x, w) = 1$, $\Pr[\text{PVer}(\text{ppk}, \text{psk}, x, \Pi)] = 1$, where $(\text{ppk}, \text{psk}) \leftarrow_{\S} \text{PGen}(\text{pars})$ and $(\Pi, K) \leftarrow_{\S} \text{PPrv}(\text{ppk}, x, w)$.
- **Perfect zero-knowledge:** For all possible public parameters pars , all key pairs (ppk, psk) in the output range of $\text{PGen}(\text{pars})$, all statements $x \in \mathcal{L}$ and all witnesses w with $\mathcal{R}(x, w) = 1$, we have $\text{PPrv}(\text{ppk}, x, w) = \text{PSim}(\text{ppk}, \text{psk}, x)$.
- **Unique of the proofs:** For all possible public parameters pars , all key pairs (ppk, psk) in the output range of $\text{PGen}(\text{pars})$ and all statements $x \in \mathcal{X}$, there exists at most one Π^* such that $\text{PVer}(\text{ppk}, \text{psk}, x, \Pi^*) = 1$.

$\text{Exp}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{A}}^{\text{csnd}}(\lambda):$ $\text{win} = 0$ $(\text{ppk}, \text{psk}) \leftarrow_{\mathfrak{s}} \text{PGen}(\text{pars})$ $\mathcal{A}^{\mathcal{O}_{\text{sim}}(), \mathcal{O}_{\text{ver}}(\cdot, \cdot, \cdot)}(\text{ppk})$ <hr/> $\mathcal{O}_{\text{sim}}():$ $x \leftarrow_{\mathfrak{s}} \mathcal{L}^{\text{snd}} \setminus \mathcal{L}$ $(\Pi, K) \leftarrow \text{PSim}(\text{psk}, x)$ $\text{Return } (x, \Pi, K)$	$\mathcal{O}_{\text{ver}}(x, \Pi, \text{pred}):$ $(v, K) \leftarrow \text{PVer}(\text{psk}, x, \Pi)$ $\text{If } v = 1 \wedge \text{pred}(K) = 1:$ $\text{If } x \in \mathcal{L}:$ $\text{Return } K$ Else: $\text{win} = \begin{cases} 0 & \text{If } x \in \mathcal{L}^{\text{snd}} \\ 1 & \text{Otherwise} \end{cases}$ Abort game $\text{Return } \perp$
---	--

Fig. 8. Experiment used in the definition of constrained \mathcal{L}^{snd} -soundness of PS.

- **Constrained \mathcal{L}^{snd} -Soundness:** For any stateful PPT adversary \mathcal{A} , consider the soundness experiment in Fig. 8 (where PSim and PVer are implicitly assumed to have access to ppk).

Let Q_{ver} be the total number of \mathcal{O}_{ver} queries, which is independent of the environment without loss of generality. Let $\text{pred}_i : \mathcal{K} \cup \{\perp\} \rightarrow \{0, 1\}$ be the predicate submitted by \mathcal{A} in the i -th query, where $\text{pred}_i(\perp) = 0$ for all i . The uncertainty of \mathcal{A} is defined as

$$\text{uncert}_{\mathcal{A}}(\lambda) := \frac{1}{Q_{\text{ver}}} \sum_{i=1}^{Q_{\text{ver}}} \Pr_{K \leftarrow_{\mathfrak{s}} \mathcal{K}}[\text{pred}_i(K) = 1].$$

We say constrained \mathcal{L}^{snd} -soundness holds for PS if for each PPT adversary \mathcal{A} with negligible uncertainty, $\text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{A}}^{\text{csnd}}(\lambda)$ is negligible, where

$$\text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{A}}^{\text{csnd}}(\lambda) := \Pr[\text{win} = 1 \text{ in } \text{Exp}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{A}}^{\text{csnd}}(\lambda)]$$

We omit the definition for \mathcal{L}^{snd} -indistinguishability of two proof systems and the definition for $\widetilde{\mathcal{L}^{\text{snd}}}$ -extensibility of a proof system (See [8] and also our full version [21] for details). Here we define a new property for qualified proof system, which stresses that the simulated proof Π for a random $x \in \mathcal{L}^{\text{snd}} \setminus \mathcal{L}$ is pseudorandom when providing verification oracle for only $x \in \mathcal{L}$.

Definition 9 (Pseudorandom Simulated Proof of Qualified Proof System). Let $\text{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$ be a \mathcal{L}^{snd} -qualified proof system for a family of languages \mathcal{L} . Let \mathcal{A} be a stateful adversary and $b \in \{0, 1\}$ be a bit. Define the following experiment $\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof-}b}(\lambda)$ in Fig. 9. We say PS has pseudorandom simulated proof if for each PPT adversary \mathcal{A} , the advantage

$$\text{Adv}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda) := \left| \Pr \left[\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof-0}}(\lambda) = 1 \right] - \Pr \left[\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof-1}}(\lambda) = 1 \right] \right| \text{ is negl.}$$

$\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof-}b}(\lambda) : // b \in \{0, 1\}$ $(\text{ppk}, \text{psk}) \leftarrow_{\S} \text{PGen}(\text{pars})$ $b' \leftarrow_{\S} \mathcal{A}^{\mathcal{O}_{\text{sim}}(), \mathcal{O}_{\text{ver}}(\cdot, \cdot)}(\text{ppk})$ $\text{Return } b'$	$\mathcal{O}_{\text{sim}}():$ $x \leftarrow_{\S} \mathcal{L}^{\text{snd}} \setminus \mathcal{L}$ $\Pi_0 \leftarrow_{\S} \Pi$ $(\Pi_1, K) \leftarrow \text{PSim}(\text{psk}, x)$ $\text{Return } (x, \Pi_b)$	$\mathcal{O}_{\text{ver}}(x, \Pi):$ $(v, K) \leftarrow \text{PVer}(\text{psk}, x, \Pi)$ $\text{If } x \notin \mathcal{L} \vee v = 0:$ $\text{Return } \perp$ $\text{Return } K$
--	---	---

Fig. 9. Experiment used in the definition of pseudorandom simulated proof of PS.

The Qualified Proof System in [8]. First we explain how the public parameters pars are sampled. Fix some $k \in \mathbb{N}$, invoke $\mathcal{G} \leftarrow_{\S} \text{GGen}(1^\lambda)$ where $\mathcal{G} = (\mathbb{G}, q, P)$. Let $\mathcal{D}_{2k, k}$ be a fixed matrix distribution, we sample $\mathbf{A} \leftarrow_{\S} \mathcal{D}_{2k, k}$ and $\mathbf{A}_0 \leftarrow_{\S} \mathcal{U}_{2k, k}$ where $\overline{\mathbf{A}}$ and $\overline{\mathbf{A}}_0$ are both full rank. Additionally select $\mathbf{A}_1 \in \mathbb{Z}_q^{2k \times k}$ according to $\mathcal{U}_{2k, k}$ with the restriction $\overline{\mathbf{A}}_0 = \overline{\mathbf{A}}_1$. Let \mathcal{H}_0 and \mathcal{H}_1 be universal hash function generators returning functions $\mathbf{h}_0 : \mathbb{G}^{k^2+1} \rightarrow \mathbb{Z}_q^{k \times k}$ and $\mathbf{h}_1 : \mathbb{G}^{k+1} \rightarrow \mathbb{Z}_q^k$ respectively. Let $\mathbf{h}_0 \leftarrow_{\S} \mathcal{H}_0$ and $\mathbf{h}_1 \leftarrow_{\S} \mathcal{H}_1$. Let $\text{pars} \leftarrow (k, \mathcal{G}, [\mathbf{A}], [\mathbf{A}_0], [\mathbf{A}_1], \mathbf{h}_0, \mathbf{h}_1)$ be the public parameters and we assume pars is an implicit input of all algorithms. The languages are defined as $\mathcal{L} := \text{span}([\mathbf{A}])$, $\mathcal{L}^{\text{snd}} := \text{span}([\mathbf{A}]) \cup \text{span}([\mathbf{A}_0])$ and $\widetilde{\mathcal{L}}^{\text{snd}} := \text{span}([\mathbf{A}]) \cup \text{span}([\mathbf{A}_0]) \cup \text{span}([\mathbf{A}_1])$.

The construction³ of \mathcal{L}^{snd} -qualified proof system $\text{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$ in [8] is shown in Fig. 10.

According to Theorem 1 of [8], PS is \mathcal{L}^{snd} -qualified and $\widetilde{\mathcal{L}}^{\text{snd}}$ -extensible, both admitting tight security reductions to the MDDH assumption. More precisely, $\text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{A}}^{\text{csnd}}, \text{PS}, \mathcal{A}(\lambda), \text{Adv}_{\widetilde{\mathcal{L}}^{\text{snd}}, \widetilde{\text{PS}}, \mathcal{A}}^{\text{csnd}}(\lambda) \leq 2k \cdot \text{Adv}_{\mathcal{D}_{2k, k}, \text{GGen}, \mathcal{B}}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}, \text{Adv}_{\mathcal{L}^{\text{snd}}}^{\text{PS-ind}} \leq 2^{-\Omega(\lambda)}$.

We now prove that PS has pseudorandom simulated proof with Theorem 2.

Theorem 2. *The \mathcal{L}^{snd} -qualified proof system PS in Fig. 10 has pseudorandom simulated proof if \mathcal{U}_k -MDDH assumption holds. Specifically, for each PPT adversary \mathcal{A} , we can build a PPT adversary \mathcal{B} with $\mathbf{T}(\mathcal{B}) \leq \mathbf{T}(\mathcal{A}) + (Q_{\text{sim}} + Q_{\text{ver}}) \cdot \text{poly}(\lambda)$ such that the advantage*

$$\text{Adv}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda) \leq 2\text{Adv}_{\mathcal{U}_k, \text{GGen}, \mathcal{B}}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}.$$

where $Q_{\text{sim}}(Q_{\text{ver}})$ is the total number of $\mathcal{O}_{\text{sim}}(\mathcal{O}_{\text{ver}})$ queries made by \mathcal{A} and $\text{poly}(\lambda)$ is a polynomial independent of $\mathbf{T}(\mathcal{A})$.

Proof of Theorem 2.

For a fixed PPT adversary \mathcal{A} , consider an experiment $\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda)$ which first uniformly selects $b \leftarrow_{\S} \{0, 1\}$, then calls $\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof-}b}(\lambda)$ and gets its output b' . It is straightforward that

³ This construction in Fig. 10 is an updated version of [8] from a personal communication.

<p><u>PGen(pars):</u></p> $\mathbf{K}_X \leftarrow_{\S} \mathbb{Z}_q^{(k^2+1) \times 2k}$ $\mathbf{K}_Y \leftarrow_{\S} \mathbb{Z}_q^{(k+1) \times 2k}$ $[\mathbf{P}_X] \leftarrow \mathbf{K}_X[\mathbf{A}] \in \mathbb{G}^{(k^2+1) \times k}$ $[\mathbf{P}_Y] \leftarrow \mathbf{K}_Y[\mathbf{A}] \in \mathbb{G}^{(k+1) \times k}$ $\text{ppk} \leftarrow ([\mathbf{P}_X], [\mathbf{P}_Y])$ $\text{psk} \leftarrow (\mathbf{K}_X, \mathbf{K}_Y)$ <p>Return (ppk, psk)</p>	<p><u>PSim(ppk, psk, [c]):</u></p> $\mathbf{X} \leftarrow h_0(\mathbf{K}_X[\mathbf{c}])$ $\mathbf{y} \leftarrow h_1(\mathbf{K}_Y[\mathbf{c}])$ $[\pi] \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\bar{\mathbf{c}}] \cdot \mathbf{y}^\top$ $[\mathbf{K}] \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\mathbf{c}] \cdot \mathbf{y}^\top$ $[\kappa] \leftarrow \text{trace}([\mathbf{K}])$ <p>Return ([π], [κ])</p>
<p><u>PPrv(ppk, [c], r):</u></p> $\mathbf{X} \leftarrow h_0([\mathbf{P}_X]\mathbf{r}) \in \mathbb{Z}_q^{k \times k}$ $\mathbf{y} \leftarrow h_1([\mathbf{P}_Y]\mathbf{r}) \in \mathbb{Z}_q^k$ $[\pi] \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\bar{\mathbf{c}}] \cdot \mathbf{y}^\top \in \mathbb{G}^{k \times k}$ $[\mathbf{K}] \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\mathbf{c}] \cdot \mathbf{y}^\top \in \mathbb{G}^{k \times k}$ $[\kappa] \leftarrow \text{trace}([\mathbf{K}]) \in \mathbb{G}$ <p>Return ([π], [κ])</p>	<p><u>PVer(ppk, psk, [c], [π^*]):</u></p> $([\pi], [\kappa]) \leftarrow \text{PSim}(\text{ppk}, \text{psk}, [\mathbf{c}])$ <p>Return $\begin{cases} (1, [\kappa]) & \text{If } [\pi] = [\pi^*] \\ (0, \perp) & \text{Otherwise} \end{cases}$</p>

Fig. 10. Construction of the \mathcal{L}^{snd} -qualified proof system $\text{PS} = (\text{PGen}, \text{PPrv}, \text{PVer}, \text{PSim})$ in [8].

$$\text{Adv}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda) = 2 \left| \Pr[b' = b \text{ in } \text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda)] - \frac{1}{2} \right|.$$

Now we rewrite $\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda)$ in Fig. 11 and make changes to it gradually through game G_0 to G_3 . Games $G_0 - G_3$ are defined as follows.

Game G_0 . This game is the same as $\text{Exp}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda)$. Then

$$\text{Adv}_{\text{PS}, \mathcal{A}}^{\text{pr-proof}}(\lambda) = 2 \left| \Pr_0[b' = b] - \frac{1}{2} \right|. \quad (14)$$

Game $G_0 - G_1$. G_1 is almost the same as G_0 except for the \mathcal{O}_{sim} oracle.

- In G_0 , $\mathbf{X} = h_0(\mathbf{K}_X[\mathbf{c}])$, where $[\mathbf{c}] = [\mathbf{A}_0]\mathbf{r}$ and $\mathbf{r} \leftarrow_{\S} \mathbb{Z}_q^k$ for each \mathcal{O}_{sim} query.
- In G_1 , $\mathbf{X} = h_0([\mathbf{V}\mathbf{r}])$, where (i) a fresh \mathbf{r} is uniformly chosen from \mathbb{Z}_q^k for each \mathcal{O}_{sim} query; (ii) \mathbf{V} is uniformly chosen from $\mathbb{Z}_q^{(k^2+1) \times k}$ beforehand but will be fixed for each \mathcal{O}_{sim} query.

Define $\mathbf{U} := \mathbf{K}_X \mathbf{A}_0$, so $(\mathbf{P}_X | \mathbf{U}) = \mathbf{K}_X(\mathbf{A} | \mathbf{A}_0)$. Note that, the square matrix $(\mathbf{A} | \mathbf{A}_0)$ is of full rank with probability $1 - 2^{-\Omega(\lambda)}$, then the entropy of \mathbf{K}_X is transferred to $(\mathbf{P}_X | \mathbf{U})$ intactly. Recall that \mathbf{K}_X is uniform over $\mathbb{Z}_q^{(k^2+1) \times 2k}$. Therefore, $(\mathbf{P}_X | \mathbf{U})$ is uniform over $\mathbb{Z}_q^{(k^2+1) \times 2k}$ as well. Consequently, \mathbf{U} is uniformly distributed over $\mathbb{Z}_q^{(k^2+1) \times k}$ even conditioned on \mathbf{P}_X .

$\text{Exp}_{\mathcal{P}_{\mathcal{S}}, \mathcal{A}}^{\text{pr-proof}}(\lambda): G_0 \text{ } \overline{G_1 - G_3}$ $b \leftarrow_{\mathcal{S}} \{0, 1\}$ $\mathbf{V} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{(k^2+1) \times k}$ $\mathbf{K}_X \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{(k^2+1) \times 2k}$ $\mathbf{K}_Y \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{(k+1) \times 2k}$ $[\mathbf{P}_X] \leftarrow \mathbf{K}_X[\mathbf{A}]$ $[\mathbf{P}_Y] \leftarrow \mathbf{K}_Y[\mathbf{A}]$ $\text{ppk} \leftarrow ([\mathbf{P}_X], [\mathbf{P}_Y])$ $b' \leftarrow_{\mathcal{S}} \mathcal{A}^{\mathcal{O}_{\text{sim}}(\cdot), \mathcal{O}_{\text{ver}}(\cdot, \cdot)}(\text{ppk})$ $\text{Return } b'$	$\mathcal{O}_{\text{sim}}(): G_0 \text{ } \overline{G_1} \text{ } \overline{G_2 \ G_3}$ $\mathbf{r} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k, [\mathbf{c}] \leftarrow [\mathbf{A}_0]\mathbf{r}$ $\Pi_0 \leftarrow_{\mathcal{S}} \mathbb{G}^{k \times k}$ $\mathbf{X} \leftarrow \mathbf{h}_0(\mathbf{K}_X[\mathbf{c}])$ $\overline{\mathbf{X}} \leftarrow \mathbf{h}_0([\mathbf{V}\mathbf{r}])$ $\overline{\mathbf{X}} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{k \times k}$ $\mathbf{y} \leftarrow \mathbf{h}_1(\mathbf{K}_Y[\mathbf{c}])$ $\Pi_1 \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\overline{\mathbf{c}}] \cdot \mathbf{y}^\top$ $\overline{\Pi}_1 \leftarrow_{\mathcal{S}} \mathbb{G}^{k \times k}$ $\text{Return } ([\mathbf{c}], \Pi_b)$	$\mathcal{O}_{\text{ver}}([\mathbf{c}], \Pi^*): G_0 - G_3$ $\mathbf{X} \leftarrow \mathbf{h}_0(\mathbf{K}_X[\mathbf{c}])$ $\mathbf{y} \leftarrow \mathbf{h}_1(\mathbf{K}_Y[\mathbf{c}])$ $\Pi \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\overline{\mathbf{c}}] \cdot \mathbf{y}^\top$ $[\mathbf{K}] \leftarrow [\mathbf{A}_0] \cdot \mathbf{X} + [\mathbf{c}] \cdot \mathbf{y}^\top$ $[\kappa] \leftarrow \text{trace}([\mathbf{K}])$ $\text{If } \left[\begin{array}{l} [\mathbf{c}] \notin \text{span}([\mathbf{A}]) \\ \vee \Pi \neq \Pi^* \end{array} \right]:$ $\text{Return } \perp$ $\text{Return } [\kappa]$
--	--	---

Fig. 11. Games $G_0 - G_3$ in the proof of Theorem 2.

In G_0 , the \mathcal{O}_{ver} oracle rejects all $[\mathbf{c}] \notin [\text{span}(\mathbf{A})]$. Therefore, the information of \mathbf{K}_X leaked through \mathcal{O}_{ver} is characterized by the public key \mathbf{P}_X . Together with the fact that $[\mathbf{c}] = [\mathbf{A}_0]\mathbf{r}$ in \mathcal{O}_{sim} of G_0 and G_1 , the computation of $\mathbf{K}_X[\mathbf{c}] = [\mathbf{K}_X\mathbf{A}_0]\mathbf{r}$ in \mathcal{O}_{sim} of G_0 can be replaced with $[\mathbf{V}]\mathbf{r}$ for $\mathbf{V} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{(k^2+1) \times k}$ in G_1 . Thus we have

$$|\Pr_0[b' = b] - \Pr_1[b' = b]| \leq 2^{-\Omega(\lambda)}. \quad (15)$$

Game $G_1 - G_2$. G_2 is the same as G_1 except for the \mathcal{O}_{sim} oracle.

- In G_1 , $\mathbf{X} = \mathbf{h}_0([\mathbf{V}\mathbf{r}])$ is computed with the same \mathbf{V} but a fresh $\mathbf{r} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$.
- In G_2 , \mathbf{X} is uniformly selected from $\mathbb{Z}_q^{k \times k}$ for each \mathcal{O}_{sim} oracle.

We will show that

$$|\Pr_1[b' = b] - \Pr_2[b' = b]| \leq \text{Adv}_{\mathcal{U}_k, \mathcal{G}^{\text{mddh}}, \mathcal{B}}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (16)$$

To prove (16), we define two intermediate games G'_1 and G''_1 .

G'_1 is the same as G_1 except for the generation of \mathbf{r} in \mathcal{O}_{sim} . For each \mathcal{O}_{sim} query,

- in G_1 , $\mathbf{r} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$;
- in G'_1 , $\mathbf{r} \leftarrow \mathbf{W}\mathbf{s}$ with a fresh $\mathbf{s} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$ but the same \mathbf{W} , which is uniformly selected from $\mathbb{Z}_q^{k \times k}$ beforehand.

Since \mathbf{W} is invertible with probability $1 - 2^{-\Omega(\lambda)}$, we have that

$$|\Pr_1[b' = b] - \Pr_{1'}[b' = b]| \leq 2^{-\Omega(\lambda)}. \quad (17)$$

G''_1 is the same with G'_1 except for the \mathcal{O}_{sim} oracle. For each \mathcal{O}_{sim} query,

- G'_1 sets $[\mathbf{c}] \leftarrow \mathbf{A}_0[\mathbf{W}]\mathbf{s}$ and $\mathbf{X} \leftarrow \mathbf{h}_0([\mathbf{V}\mathbf{W}]\mathbf{s})$, where $\mathbf{s} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$;
- G''_1 sets $[\mathbf{c}] \leftarrow \mathbf{A}_0[\mathbf{r}]$ and $\mathbf{X} \leftarrow \mathbf{h}_0([\mathbf{u}])$, where $\mathbf{r} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^k$, $\mathbf{u} \leftarrow_{\mathcal{S}} \mathbb{Z}_q^{k^2+1}$.

Note that, with overwhelming probability, $[\mathbf{B}] = [\frac{\mathbf{W}}{\mathbf{VW}}]$ distributes uniformly over $\mathbb{G}^{(k^2+k+1) \times k}$. Then we can build an adversary \mathcal{B} and show that

$$|\Pr_{1'}[b' = b] - \Pr_{1''}[b' = b]| \leq \text{Adv}_{\mathcal{U}_{k^2+k+1,k}, \text{GGen}, \mathcal{B}}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (18)$$

To prove (18), we construct an adversary \mathcal{B}' and show that

$$|\Pr_{1'}[b' = b] - \Pr_{1''}[b' = b]| \leq \text{Adv}_{\mathcal{U}_{k^2+k+1,k}, \text{GGen}, \mathcal{B}'}^{Q_{\text{sim}}\text{-mddh}}(\lambda). \quad (19)$$

Upon receiving a challenge $(\mathcal{G}, [\mathbf{B}] \in \mathbb{G}^{(k^2+k+1) \times k}, [\mathbf{H}] := ([\mathbf{h}_1 | \dots | \mathbf{h}_{Q_{\text{sim}}}] \in \mathbb{G}^{(k^2+k+1) \times Q_{\text{sim}}})$ for the Q_{sim} -fold $\mathcal{U}_{k^2+k+1,k}$ -MDDH problem, \mathcal{B}' simulates game $G'_1(G''_1)$. In the simulation of the i -th \mathcal{O}_{sim} oracle query for $i \in [Q_{\text{sim}}]$, \mathcal{B}' embeds $[\mathbf{h}_i]$ in $[\mathbf{c}]$ with $[\mathbf{c}] \leftarrow \mathbf{A}_0[\mathbf{h}_i]$. Then \mathcal{B}' embeds $[\mathbf{h}_i]$ in \mathbf{X} with $\mathbf{X} \leftarrow \mathbf{h}_0([\mathbf{h}_i])$.

If $[\mathbf{h}_i]$ is uniformly chosen from $\text{span}([\mathbf{B}])$ for all $i \in [Q_{\text{sim}}]$, then $[\mathbf{h}_i] = [\frac{\mathbf{W}}{\mathbf{VW}}]\mathbf{s}_i$, $[\mathbf{h}_i] = [\mathbf{W}]\mathbf{s}_i$ and $[\mathbf{h}_i] = [\mathbf{VW}]\mathbf{s}_i$ with $\mathbf{s}_i \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^k$. In this case, \mathcal{B}' perfectly simulates G'_1 . If $[\mathbf{h}_i]$ is uniformly chosen from \mathbb{G}^{k^2+k+1} for all $i \in [Q_{\text{sim}}]$, then both $[\mathbf{h}_i]$ and $[\mathbf{h}_i]$ are uniform. In this case, \mathcal{B}' perfectly simulates G''_1 .

From above, (19) follows. Then, (18) follows from (19) and the random self-reducibility property of the MDDH problem.

In G''_1 , $\mathbf{X} \leftarrow \mathbf{h}_0([\mathbf{u}])$ for a uniform $\mathbf{u} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{k^2+1}$. Since \mathbf{h}_0 is universal, by leftover hash lemma and a union bound, we have that

$$|\Pr_{1''}[b' = b] - \Pr_2[b' = b]| \leq \frac{Q_{\text{sim}}}{2\sqrt{q}} = 2^{-\Omega(\lambda)}. \quad (20)$$

Then (16) follows from (17, 18) and (20).

Game $G_2 - G_3$. G_3 is the same as G_2 except for the \mathcal{O}_{sim} oracle.

For each \mathcal{O}_{sim} query,

- in G_2 , $\Pi_1 = [\overline{\mathbf{A}_0}] \cdot \mathbf{X} + [\overline{\mathbf{c}}] \cdot \mathbf{y}^\top$ for $[\mathbf{c}] = [\mathbf{A}_0]\mathbf{r}$ and a fresh $\mathbf{X} \leftarrow_{\mathfrak{s}} \mathbb{Z}_q^{k \times k}$;
- in G_3 , Π_1 is uniformly selected from $\mathbb{G}^{k \times k}$.

Note that in G_2 ,

$$\Pi_1 = [\overline{\mathbf{A}_0}] \cdot \mathbf{X} + [\overline{\mathbf{c}}] \cdot \mathbf{y}^\top = [\overline{\mathbf{A}_0}](\mathbf{X} + \mathbf{r} \cdot \mathbf{y}^\top).$$

Due to the uniformness of \mathbf{X} , Π_1 has the same distribution as $[\overline{\mathbf{A}_0}]\mathbf{X}$. Since $\overline{\mathbf{A}_0}$ is an invertible matrix, $[\overline{\mathbf{A}_0}]\mathbf{X}$ is uniformly distributed over $\mathbb{G}^{k \times k}$. Thus we have

$$\Pr_2[b' = b] = \Pr_3[b' = b]. \quad (21)$$

Game G_3 . In G_3 , Π_0 distributes identically to Π_1 and

$$\Pr_3[b' = b] = \frac{1}{2}. \quad (22)$$

Finally, Theorem 2 follows from (14, 15, 16, 21) and (22). \blacksquare

$\frac{(\text{pk}_{\text{kem}}, \text{sk}_{\text{kem}}) \leftarrow_{\S} \text{KGen}(1^\lambda):}{(\text{ppk}, \text{psk}) \leftarrow_{\S} \text{PGen}(\text{pars})}$ $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_{\S} \mathbb{Z}_q^{2k}, \quad [\mathbf{p}_0^\top] \leftarrow \mathbf{k}_0^\top [\mathbf{A}] \in \mathbb{G}^{1 \times k}, \quad [\mathbf{p}_1^\top] \leftarrow \mathbf{k}_1^\top [\mathbf{A}] \in \mathbb{G}^{1 \times k}$ $\text{Return } \text{pk}_{\text{kem}} \leftarrow (\text{ppk}, [\mathbf{p}_0^\top], [\mathbf{p}_1^\top]), \quad \text{sk}_{\text{kem}} \leftarrow (\text{psk}, \mathbf{k}_0, \mathbf{k}_1)$	
$\frac{(\psi, \gamma) \leftarrow_{\S} \text{KEnc}(\text{pk}_{\text{kem}}):}{\mathbf{r} \leftarrow_{\S} \mathbb{Z}_q^k, \quad [\mathbf{c}] \leftarrow [\mathbf{A}]\mathbf{r} \in \mathbb{G}^{2k}}$ $(II, [\kappa]) \leftarrow_{\S} \text{PPrv}(\text{ppk}, [\mathbf{c}], \mathbf{r})$ $\tau \leftarrow \text{H}([\bar{\mathbf{c}}]) \in \{0, 1\}^\lambda \subseteq \mathbb{Z}_q$ $\gamma \leftarrow ([\mathbf{p}_0^\top] + \tau[\mathbf{p}_1^\top]) \cdot \mathbf{r} + [\kappa] \in \mathbb{G}$ $\text{Return } (\psi \leftarrow ([\mathbf{c}], II), \gamma)$ $// \Psi = \mathbb{G}^{2k} \times \mathbb{G}^{k \times k}, \quad \Gamma = \mathbb{G}$	$\frac{\gamma/\perp \leftarrow \text{KDec}(\text{sk}_{\text{kem}}, \psi):}{\text{Parse } \psi = ([\mathbf{c}], II)}$ $(v \in \{0, 1\}, [\kappa]) \leftarrow \text{PVer}(\text{psk}, [\mathbf{c}], II)$ $\tau \leftarrow \text{H}([\bar{\mathbf{c}}]) \in \{0, 1\}^\lambda \subseteq \mathbb{Z}_q$ $\gamma \leftarrow (\mathbf{k}_0^\top + \tau\mathbf{k}_1^\top) \cdot [\mathbf{c}] + [\kappa] \in \mathbb{G}$ $\text{Return } \begin{cases} \gamma & \text{If } v = 1 \\ \perp & \text{Otherwise} \end{cases}$

Fig. 12. Construction of $\text{KEM}_{\text{qps}} = (\text{KGen}, \text{KEnc}, \text{KDec})$ in [8]

KEM from Qualified Proof System. The construction of the qualified PS based KEM $\text{KEM}_{\text{qps}} = (\text{KGen}, \text{KEnc}, \text{KDec})$ from [8] is shown in Fig. 12. Suppose \mathcal{H} is a hash generator outputting functions $\text{H} : \mathbb{G}^k \rightarrow \{0, 1\}^\lambda$. The parameters pars used in this construction are specified in Sect. 5.2.

Theorem 2 in [8] has shown that KEM_{qps} is IND-CCCA secure. Now we prove that KEM_{qps} is mPR-CCCA secure (through Theorem 3) and is RER secure (through Theorem 4), both admitting tight security reductions.

Theorem 3. *The KEM KEM_{qps} in Fig. 12 is mPR-CCCA secure if the $\mathcal{D}_{2k, k}$ -MDDH assumption holds, \mathcal{H} outputs collision-resistant hash function, PS is \mathcal{L}^{snd} -qualified, \mathcal{L}^{snd} -extensible and has pseudorandom simulated proof. Specifically, for each PPT adversary \mathcal{A} with negligible uncertainty $\text{uncert}_{\mathcal{A}}(\lambda)$, we can build PPT adversaries $\mathcal{B}_1, \dots, \mathcal{B}_7$ with $\mathbf{T}(\mathcal{B}_1) \approx \dots \approx \mathbf{T}(\mathcal{B}_7) \leq \mathbf{T}(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \text{poly}(\lambda)$ and $\text{uncert}_{\mathcal{B}_4}(\lambda) = \text{uncert}_{\mathcal{B}_6}(\lambda) = \text{uncert}_{\mathcal{A}}(\lambda)$, such that the advantage*

$$\begin{aligned} \text{Adv}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda) &\leq 2\text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{\text{cr}}(\lambda) + (4\lambda + 3k)\text{Adv}_{\mathcal{D}_{2k, k}, \text{GGen}, \mathcal{B}_2}^{\text{mddh}}(\lambda) \\ &\quad + 7\text{Adv}_{\mathcal{U}_k, \text{GGen}, \mathcal{B}_3}^{\text{mddh}}(\lambda) + \text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{B}_4}^{\text{csnd}}(\lambda) + \text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \widetilde{\text{PS}}, \mathcal{B}_5}^{\text{ps-ind}}(\lambda) \\ &\quad + \lambda\text{Adv}_{\mathcal{L}^{\text{snd}}, \widetilde{\text{PS}}, \mathcal{B}_6}^{\text{csnd}}(\lambda) + 2\text{Adv}_{\text{PS}, \mathcal{B}_7}^{\text{pr-proof}}(\lambda) \\ &\quad + ((\lambda + 2) \cdot Q_{\text{enc}} + 3) \cdot Q_{\text{dec}} \cdot \text{uncert}_{\mathcal{A}}(\lambda) + 2^{-\Omega(\lambda)}. \end{aligned}$$

where $Q_{\text{enc}}(Q_{\text{dec}})$ is the total number of $\mathcal{O}_{\text{enc}}(\mathcal{O}_{\text{dec}})$ queries made by \mathcal{A} and $\text{poly}(\lambda)$ is a polynomial independent of $\mathbf{T}(\mathcal{A})$.

Proof of Theorem 3. For a fixed PPT adversary \mathcal{A} with negligible uncertainty $\text{uncert}_{\mathcal{A}}(\lambda)$, consider an experiment $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda)$ which first randomly selects

$b \leftarrow_{\S} \{0, 1\}$, then calls $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca-}b}(\lambda)$ and gets its output b' . It is straightforward that $\text{Adv}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda) = 2 \left| \Pr[b' = b \text{ in } \text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda)] - \frac{1}{2} \right|$. Then we rewrite experiment $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda)$ in Fig. 13 and make changes to it gradually through game G_0 to G_9 which are defined as follows.

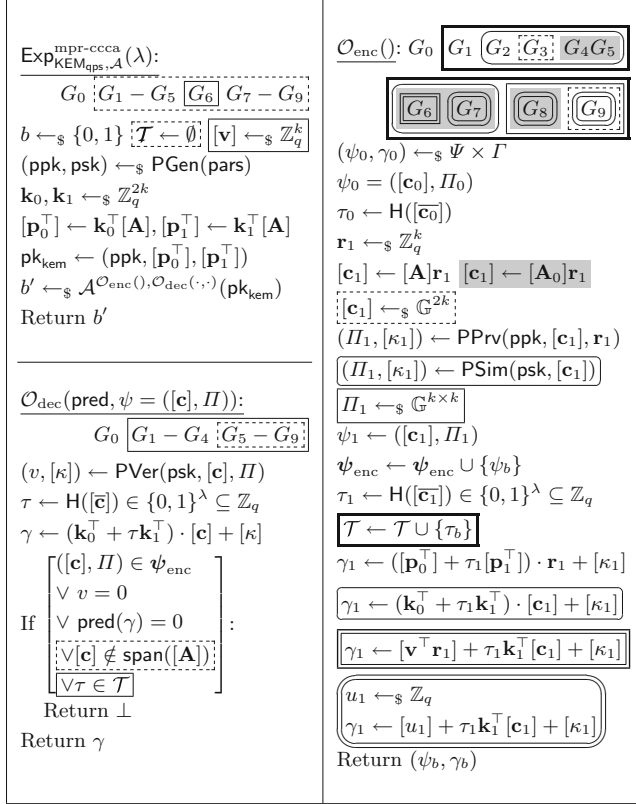


Fig. 13. Game $G_0 - G_9$ in the Proof of Theorem 3.

Game G_0 . This game is identical to $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda)$. Then

$$\text{Adv}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{mpr-ccca}}(\lambda) = 2 \left| \Pr_0[b' = b] - \frac{1}{2} \right|. \quad (23)$$

Game $G_0 - G_1$. G_1 is the same as G_0 except that an additional rejection rule is added in \mathcal{O}_{dec} . More precisely, in G_1 , we use a set \mathcal{T} to log all the tags $\tau_b = \text{H}([\bar{\mathbf{c}}_b])$ used in oracle \mathcal{O}_{enc} , and any $\mathcal{O}_{\text{dec}}(\text{pred}, \psi = ([\mathbf{c}], \Pi))$ query will be rejected if $\tau = \text{H}([\bar{\mathbf{c}}]) \in \mathcal{T}$.

Lemma 1.

$$\begin{aligned} |\Pr_0[b' = b] - \Pr_1[b' = b]| &\leq \text{Adv}_{\mathcal{H}, \mathcal{B}_1}^{cr}(\lambda) + \frac{k}{2} \cdot \text{Adv}_{\mathcal{D}_{2k, k}, \text{Gen}, \mathcal{B}_2}^{\text{mddh}}(\lambda) \\ &\quad + \frac{1}{2} \text{Adv}_{\mathcal{U}_k, \text{Gen}, \mathcal{B}_3}^{\text{mddh}}(\lambda) + \frac{3}{2} Q_{\text{dec}} \cdot \text{uncert}_{\mathcal{A}}(\lambda) + 2^{-\Omega(\lambda)}. \end{aligned}$$

We refer to the full version [21] for the proof of this lemma.

Game $G_1 - G_2$. G_2 is almost the same as G_1 except for two changes in \mathcal{O}_{enc} . The first change is that PPrv is replaced with PSim. The second change is that sk_{KEM} is used to calculate γ_1 . More precisely, for $[\mathbf{c}_1] = [\mathbf{A}]\mathbf{r}_1$ in oracle \mathcal{O}_{enc} ,

- in G_1 , $(\Pi_1, [\kappa_1]) \leftarrow \text{PPrv}(\text{ppk}, [\mathbf{c}_1], \mathbf{r}_1)$, $\gamma_1 \leftarrow ([\mathbf{p}_0^\top] + \tau_1[\mathbf{p}_1^\top]) \cdot \mathbf{r}_1 + [\kappa_1]$;
- in G_2 , $(\Pi_1, [\kappa_1]) \leftarrow \text{PSim}(\text{psk}, [\mathbf{c}_1])$, $\gamma_1 \leftarrow (\mathbf{k}_0^\top + \tau_1\mathbf{k}_1^\top) \cdot [\mathbf{c}_1] + [\kappa_1]$.

Due to the perfect zero-knowledge property of PS, we have $\text{PPrv}(\text{ppk}, [\mathbf{c}_1], \mathbf{r}_1) = \text{PSim}(\text{psk}, [\mathbf{c}_1])$. Meanwhile, $[\mathbf{p}_0^\top] = \mathbf{k}_0^\top[\mathbf{A}]$ and $[\mathbf{p}_1^\top] = \mathbf{k}_1^\top[\mathbf{A}]$, so we have $([\mathbf{p}_0^\top] + \tau_1[\mathbf{p}_1^\top]) \cdot \mathbf{r}_1 + [\kappa_1] = (\mathbf{k}_0^\top + \tau_1\mathbf{k}_1^\top) \cdot [\mathbf{c}_1] + [\kappa_1]$.

These changes are only conceptual, so G_1 is identical to G_2 and

$$\Pr_1[b' = b] = \Pr_2[b' = b]. \quad (24)$$

Game $G_2 - G_3$. G_3 is the same as G_2 except for one difference in \mathcal{O}_{enc} .

- In game G_2 , $[\mathbf{c}_1]$ is uniform over $\text{span}([\mathbf{A}])$ for each \mathcal{O}_{enc} query.
- In game G_3 , $[\mathbf{c}_1]$ is uniform over \mathbb{G}^{2k} for each \mathcal{O}_{enc} query.

We can build an adversary \mathcal{B}_2 and show that

$$|\Pr_2[b' = b] - \Pr_3[b' = b]| \leq k \cdot \text{Adv}_{\mathcal{D}_{2k, k}, \text{Gen}, \mathcal{B}_2}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (25)$$

The reduction is straightforward, since \mathcal{B}_2 can simulate $G_2(G_3)$ by generating the secret key itself and embed its own challenge in $[\mathbf{c}_1]$. We omit the details.

We refer to the full version [21] for the proof of this lemma.

Game $G_3 - G_4$. G_4 is the same as G_3 except for one difference in \mathcal{O}_{enc} .

- In game G_3 , $[\mathbf{c}_1]$ is uniform over \mathbb{G}^{2k} for each \mathcal{O}_{enc} query.
- In game G_4 , $[\mathbf{c}_1]$ is uniform over $\text{span}([\mathbf{A}_0])$ for each \mathcal{O}_{enc} query.

We can build an adversary \mathcal{B}_3 and show that

$$|\Pr_3[b' = b] - \Pr_4[b' = b]| \leq \text{Adv}_{\mathcal{U}_k, \text{Gen}, \mathcal{B}_3}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (26)$$

The reduction is straightforward and the proof of (26) is almost the same as (25).

Game $G_4 - G_5$. G_5 is almost the same as G_4 except that a rejection rule is added in \mathcal{O}_{dec} . More precisely, in G_5 , an $\mathcal{O}_{\text{dec}}(\text{pred}, \psi = ([\mathbf{c}], \Pi))$ query is directly rejected if $[\mathbf{c}] \notin \text{span}([\mathbf{A}])$. We have that

$$\begin{aligned} |\Pr_4[b' = b] - \Pr_5[b' = b]| &\leq \frac{1}{2} \text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \mathcal{B}_4}^{\text{csnd}}(\lambda) + \frac{1}{2} \text{Adv}_{\mathcal{L}^{\text{snd}}, \text{PS}, \widetilde{\text{PS}}, \mathcal{B}_5}^{\text{PS-ind}}(\lambda) + Q_{\text{enc}} \cdot 2^{-\Omega(\lambda)} \\ &\quad + 2\lambda \cdot \text{Adv}_{\mathcal{D}_{2k, k}, \text{Gen}, \mathcal{B}_2}^{\text{mddh}}(\lambda) + \frac{\lambda}{2} \text{Adv}_{\mathcal{L}^{\text{snd}}, \widetilde{\text{PS}}, \mathcal{B}_6}^{\text{csnd}}(\lambda) + \frac{\lambda + 2}{2} \cdot Q_{\text{enc}} \cdot Q_{\text{dec}} \cdot \text{uncert}_{\mathcal{A}}(\lambda) \end{aligned} \quad (27)$$

The proof of (27) is the same as Lemma 9 in [8]. We refer [8] for details.

Game $G_5 - G_6$. G_6 is almost the same as G_5 except for one difference in \mathcal{O}_{enc} .

- In game G_5 , $\gamma_1 = (\mathbf{k}_0^\top + \tau_1 \mathbf{k}_1^\top) \cdot [\mathbf{c}_1] + [\kappa_1]$ for each \mathcal{O}_{enc} query.
- In game G_6 , $\gamma_1 = [\mathbf{v}^\top \mathbf{r}_1] + \tau_1 \mathbf{k}_1^\top [\mathbf{c}_1] + [\kappa_1]$ where \mathbf{v} is uniformly chosen from \mathbb{Z}_q^k beforehand but will be fixed for each \mathcal{O}_{enc} query.

We have that

$$|\Pr_5[b' = b] - \Pr_6[b' = b]| \leq 2^{-\Omega(\lambda)}. \quad (28)$$

The proof of (28) is almost the same as (15), and is put in our full version [21].

Game $G_6 - G_7$. G_7 is almost the same as G_6 except for one difference in \mathcal{O}_{enc} .

- In game G_6 , $\gamma_1 = [\mathbf{v}^\top \mathbf{r}_1] + \tau_1 \mathbf{k}_1^\top [\mathbf{c}_1] + [\kappa_1]$ for each \mathcal{O}_{enc} query.
- In game G_7 , $\gamma_1 \leftarrow [u_1] + \tau_1 \mathbf{k}_1^\top [\mathbf{c}_1] + [\kappa_1]$ where $u_1 \leftarrow_{\S} \mathbb{Z}_q$ for each \mathcal{O}_{enc} query. In other words, γ_1 is uniform for each \mathcal{O}_{enc} query in G_7 . We have that

$$|\Pr_6[b' = b] - \Pr_7[b' = b]| \leq \text{Adv}_{\mathcal{U}_{k,\text{GGen},\mathcal{B}_3}}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (29)$$

The proof of (29) is almost the same as that of (16). We can set $\mathbf{r}_1 = \mathbf{W}\mathbf{s}$ and $[\mathbf{B}] = [\mathbf{v}^\top \mathbf{W}] \in \mathbb{G}^{(k+1) \times k}$ which has the distribution $\mathcal{U}_{k+1,k}$ overwhelmingly. Then we can reduce the indistinguishability between G_6 and G_7 to the Q_{enc} -fold $\mathcal{U}_{k+1,k}$ -MDDH assumption. We omit the detailed proof here.

Note that, in game G_7 , $[\kappa_1]$ is not needed any longer since we can just select a uniform γ_1 for each \mathcal{O}_{enc} query.

Game $G_7 - G_8$. G_8 is almost the same as G_7 except for one difference in \mathcal{O}_{enc} .

- In game G_7 , Π_1 is the output of $\text{PSim}(\text{psk}, [\mathbf{c}_1])$ for each \mathcal{O}_{enc} query.
- In game G_8 , Π_1 is uniform selected for each \mathcal{O}_{enc} query.

We can build an adversary \mathcal{B}_7 and show that

$$|\Pr_7[b' = b] - \Pr_8[b' = b]| \leq \text{Adv}_{\text{PS}, \mathcal{B}_7}^{\text{pr-proof}}(\lambda). \quad (30)$$

On input ppk , \mathcal{B}_7 uniformly selects $b \leftarrow_{\S} \{0, 1\}$ and sets $\mathcal{T} \leftarrow \emptyset$. Then \mathcal{B}_7 uniformly selects $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_{\S} \mathbb{Z}_q^{2k}$ and sets $[\mathbf{p}_0^\top] \leftarrow \mathbf{k}_0^\top [\mathbf{A}]$, $[\mathbf{p}_1^\top] \leftarrow \mathbf{k}_1^\top [\mathbf{A}]$, $\text{pk}_{\text{KEM}} \leftarrow (\text{ppk}, [\mathbf{p}_0^\top], [\mathbf{p}_1^\top])$. Then \mathcal{B}_7 calls $\mathcal{A}^{\mathcal{O}_{\text{enc}}(\cdot), \mathcal{O}_{\text{dec}}(\cdot, \cdot)}(\text{pk}_{\text{KEM}})$ by simulating the two oracles for \mathcal{A} in the following way.

- For \mathcal{A} 's $\mathcal{O}_{\text{enc}}(\cdot)$ query, \mathcal{B}_7 uniformly chooses (ψ_0, γ_0) and calculates τ_0 just like game $G_7(G_8)$. Then \mathcal{B}_7 submits an \mathcal{O}_{sim} query to its own oracle and gets $([\mathbf{c}], \Pi)$ where $[\mathbf{c}]$ is uniform over $\mathcal{L}^{\text{snd}} \setminus \mathcal{L} = \text{span}([\mathbf{A}_0])$ and Π is either an output of $\text{PSim}(\text{psk}, [\mathbf{c}])$ or uniformly chosen from Π . After that \mathcal{B}_7 sets $[\mathbf{c}_1] \leftarrow [\mathbf{c}]$ and $\Pi_1 \leftarrow \Pi$. Then \mathcal{B}_7 sets ψ_{enc} , calculates τ_1 from $[\bar{\mathbf{c}}_1]$ and uniformly selects γ_1 just like game $G_7(G_8)$. Finally \mathcal{B}_7 returns (ψ_b, γ_b) to \mathcal{A} .

- For \mathcal{A} 's $\mathcal{O}_{\text{dec}}(\text{pred}, \psi = ([\mathbf{c}], \Pi))$ query, \mathcal{B}_7 submits $\mathcal{O}_{\text{ver}}([\mathbf{c}], \Pi)$ query to its own oracle and gets the response K . If $K = \perp$, \mathcal{B}_7 returns \perp to \mathcal{A} . Since $K = \perp$ means $[\mathbf{c}] \notin \text{span}([\mathbf{A}])$ or the verification $\text{PVer}(\text{psk}, [\mathbf{c}], \Pi)$ does not pass, \mathcal{B}_7 acts exactly the same as game $G_7(G_8)$ in such cases. If $[\kappa] = K \neq \perp$, \mathcal{B}_7 calculates τ and γ just like game $G_7(G_8)$. Then \mathcal{B}_7 tests if $([\mathbf{c}], \Pi) \in \psi_{\text{enc}}$ or $\text{pred}(\gamma) = 0$ or $\forall \tau \in \mathcal{T}$ happens. If so, \mathcal{B}_7 returns \perp to \mathcal{A} . Otherwise \mathcal{B}_7 returns γ to \mathcal{A} .

Finally, according to \mathcal{A} 's output b' , \mathcal{B}_7 outputs 1 if and only if $b' = b$. It is clear that if Π is an output of $\text{PSim}(\text{psk}, [\mathbf{c}])$ for each \mathcal{O}_{sim} query, \mathcal{B}_7 perfectly simulates game G_7 for \mathcal{A} . And if Π is uniformly chosen from Π for each \mathcal{O}_{sim} query, \mathcal{B}_7 perfectly simulates game G_8 for \mathcal{A} . Thus (30) follows.

Game $G_8 - G_9$. G_9 is the same as G_8 except for one difference in \mathcal{O}_{enc} .

- In game G_8 , $[\mathbf{c}_1]$ is uniform selected from $\text{span}([\mathbf{A}_0])$ for each \mathcal{O}_{enc} query.
- In game G_9 , $[\mathbf{c}_1]$ is uniform selected from \mathbb{G}^{2k} for each \mathcal{O}_{enc} query.

We can build an adversary \mathcal{B}_3 and show that

$$|\Pr_8[b' = b] - \Pr_9[b' = b]| \leq \text{Adv}_{\mathcal{U}_k, \text{GGen}, \mathcal{B}_3}^{\text{mddh}}(\lambda) + 2^{-\Omega(\lambda)}. \quad (31)$$

The reduction is straightforward and the proof of (31) is the same as the proof for (25). We omit the details here.

Game G_9 . In game G_9 , (ψ_1, Π_1) is uniform over $\Psi \times \Gamma$ for each \mathcal{O}_{enc} query, which distributes exactly the same as (ψ_0, Π_0) . Thus we have

$$\Pr_9[b' = b] = \frac{1}{2}. \quad (32)$$

Finally, Theorem 3 follows from (23), Lemma 1, (24)–(32). \blacksquare

Theorem 4. *The KEM KEM_{qps} in Fig. 12 is RER secure. Specifically, for each PPT adversary \mathcal{A} with negligible uncertainty $\text{uncert}_{\mathcal{A}}(\lambda)$, the advantage $\text{Adv}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}}(\lambda) \leq 2^{-\Omega(\lambda)}$.*

Proof of Theorem 4. In $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}-b}(\lambda)$, among all the $\mathcal{O}_{\text{cha}}(\psi, \text{pred})$ queries submitted by \mathcal{A} , if $\psi \notin \psi_{\text{ran}}$, the oracle \mathcal{O}_{cha} will answer \mathcal{A} with $\text{pred}(\text{KDec}(\text{sk}_{\text{KEM}}, \psi))$. Thus no information about b is leaked to \mathcal{A} .

Therefore, we only consider those $\mathcal{O}_{\text{cha}}(\psi, \text{pred})$ queries such that $\psi = ([\mathbf{c}], \Pi) \in \psi_{\text{ran}}$. In this case, both $[\mathbf{c}]$ and Π are uniform.

If $b = 0$, $\mathcal{O}_{\text{cha}}(\psi, \text{pred})$ will always return 0 in $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}-0}(\lambda)$.

If $b = 1$, $\mathcal{O}_{\text{cha}}(\psi, \text{pred})$ will use $\text{KDec}(\text{sk}_{\text{KEM}}, \psi)$ to decapsulate ψ . More precisely, it will invoke $\text{PVer}(\text{psk}, [\mathbf{c}], \Pi)$ to obtain $(v, [\kappa])$ and output \perp if $v = 0$. By the proof uniqueness of PS and the uniformness of Π , the probability that $v = 1$ in this query is at most $\frac{1}{|\Pi|}$. Taking into account all the \mathcal{O}_{cha} queries, a union bound suggests that $\mathcal{O}_{\text{cha}}(\psi, \text{pred})$ always outputs 0 in $\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}-1}(\lambda)$ except with probability at most $\frac{Q_{\text{cha}}}{|\Pi|} = 2^{-\Omega(\lambda)}$. Thus

$$\text{Adv}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}}(\lambda) = \left| \Pr \left[\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}-0}(\lambda) = 1 \right] - \Pr \left[\text{Exp}_{\text{KEM}_{\text{qps}}, \mathcal{A}}^{\text{rer}-1}(\lambda) = 1 \right] \right| \leq 2^{-\Omega(\lambda)}.$$

Acknowledgments. Lin Lyu, Shengli Liu and Shuai Han are supported by the National Natural Science Foundation of China (Grant Nos. 61672346, 61373153). Dawu Gu is supported by the National Natural Science Foundation of China (Grant No. U1636217) together with Program of Shanghai Academic Research Leader (16XD1401300). The authors greatly thank the anonymous reviewers of PKC 2018 for their comments and suggestions.

References

1. Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_1
2. Böhl, F., Hofheinz, D., Kraschewski, D.: On definitions of selective opening security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 522–539. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_31
3. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_25
4. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_27
5. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. In: FOCS 1999, pp. 523–534. IEEE Computer Society (1999). <https://doi.org/10.1109/SFCS.1999.814626>
6. Fehr, S., Hofheinz, D., Kiltz, E., Wee, H.: Encryption schemes secure against chosen-ciphertext selective opening attacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 381–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_20
7. 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
8. Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-Desmedt meets tight security. In: Katz and Shacham [16], pp. 133–160 (2017). https://doi.org/10.1007/978-3-319-63697-9_5
9. Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984). [https://doi.org/10.1016/0022-0000\(84\)90070-9](https://doi.org/10.1016/0022-0000(84)90070-9)
10. Hemenway, B., Libert, B., Ostrovsky, R., Vergnaud, D.: Lossy encryption: constructions from general assumptions and efficient selective opening chosen ciphertext security. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 70–88. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_4
11. Hofheinz, D.: All-but-many lossy trapdoor functions. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 209–227. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_14

12. 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
13. Hofheinz, D., Jager, T., Rupp, A.: Public-key encryption with simulation-based selective-opening security and compact ciphertexts. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 146–168. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_6
14. Huang, Z., Liu, S., Qin, B.: Sender-equivocable encryption schemes secure against chosen-ciphertext attacks revisited. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 369–385. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_23
15. Huang, Z., Liu, S., Qin, B., Chen, K.: Fixing the sender-equivocable encryption scheme in eurocrypt 2010. In: 2013 5th International Conference on Intelligent Networking and Collaborative Systems, pp. 366–372. IEEE (2013). <https://doi.org/10.1109/INCoS.2013.69>
16. Katz, J., Shacham, H. (eds.): CRYPTO 2017. LNCS, vol. 10403. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-319-63697-9>
17. Lai, J., Deng, R.H., Liu, S., Weng, J., Zhao, Y.: Identity-based encryption secure against selective opening chosen-ciphertext attack. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 77–92. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_5
18. Libert, B., Sakzad, A., Stehlé, D., Steinfeld, R.: All-but-many lossy trapdoor functions and selective opening chosen-ciphertext security from LWE. In: Katz and Shacham [16], pp. 332–364 (2017). https://doi.org/10.1007/978-3-319-63697-9_12
19. Liu, S., Paterson, K.G.: Simulation-based selective opening CCA security for PKE from key encapsulation mechanisms. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 3–26. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_1
20. Lyu, L., Liu, S., Han, S.: Public-key encryption with tight simulation-based selective-opening security. *Comput. J.* 1–31 (2017). <https://doi.org/10.1093/comjnl/bxx080>
21. Lyu, L., Liu, S., Han, S., Gu, D.: Tightly SIM-SO-CCA secure public key encryption from standard assumptions. *Cryptology ePrint Archive, Report 2018/030* (2018). <https://eprint.iacr.org/2018/030>
22. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Dwork, C. (ed.) 2008 ACM Symposium on Theory of Computing, pp. 187–196. ACM (2008). <https://doi.org/10.1145/1374376.1374406>

Searchable and Fully Homomorphic Encryption



Multi-Key Searchable Encryption, Revisited

Ariel Hamlin^(✉), Abhi Shelat, Mor Weiss^(✉), and Daniel Wichs

Northeastern University, Boston, USA

hamlin.a@husky.neu.edu, {abhi,m.weiss}@northeastern.edu,
wichs@ccs.neu.edu

Abstract. We consider a setting where users store their encrypted documents on a remote server and can selectively share documents with each other. A user should be able to perform keyword searches over all the documents she has access to, including the ones that others shared with her. The contents of the documents, and the search queries, should remain private from the server.

This setting was considered by Popa et al. (NSDI '14) who developed a new cryptographic primitive called *Multi-Key Searchable Encryption (MKSE)*, together with an instantiation and an implementation within a system called Mylar, to address this goal. Unfortunately, Grubbs et al. (CCS '16) showed that the proposed MKSE definition fails to provide basic security guarantees, and that the Mylar system is susceptible to simple attacks. Most notably, if a malicious Alice colludes with the server and shares a document with an honest Bob then the privacy of all of Bob's search queries is lost.

In this work we revisit the notion of MKSE and propose a new strengthened definition that rules out the above attacks. We then construct MKSE schemes meeting our definition. We first give a simple and efficient construction using only *pseudorandom functions*. This construction achieves our strong security definition at the cost of increasing the server storage overhead relative to Mylar, essentially replicating the document each time it is shared. We also show that high server storage overhead is not inherent, by giving an alternate (albeit impractical) construction that manages to avoid it using obfuscation.

1 Introduction

Searchable (symmetric) encryption (SSE) [6, 9, 14, 33] allows a user to outsource her encrypted documents to a remote server. Later, she (or someone she authorizes) can send the server encrypted keyword search queries and receive the set of (encrypted) documents matching her keyword. Ideally, even a compromised server would not learn anything about the user's data or her queries. This functionality can in theory be achieved using techniques such as Oblivious RAM and Fully Homomorphic Encryption, but the efficiency overhead makes the resultant schemes largely impractical.

To allow for more practical schemes, SSE relaxes the ideal security requirement and allows the server to learn some *leakage*—namely the access pattern of which documents are returned by each query. The initial SSE definitions failed to capture natural attacks, and were revised by several follow-up works culminating in the work of Curtmola et al. [9], who gave meaningful definitions that captured the intuitive security goal. There are now constructions of SSE schemes that meet this definition and are simple, practically efficient, and updatable [5, 21, 34]. We also note that there have been several works [22, 28, 36] showing that the leakage provided by SSE can already be too damaging to give meaningful security guarantees in some contexts. Despite such attacks, it seems that in many cases SSE can provide meaningful security for certain data sets, even if it is imperfect.

One benefit of outsourcing data to the cloud is that it allows users to easily share data with each other. Therefore, it is natural to consider a setting where a large group of users store their individual documents, encrypted under their own keys, on a remote cloud server, where each document can be shared with an arbitrary subset of other users. As with SSE, a user should be able to perform keyword search queries over all of the documents she has access to, including both her own documents and the ones shared with her. A trivial solution is to have each user generate a new SSE key for each set of documents she wishes to share, and provide this key to the authorized group of users. However, this solution has two main drawbacks: the user must maintain many keys (one for each set of documents shared with her), and the query size scales with the number of documents that have been shared with the user. These limitations are undesirable in many realistic scenarios, since a user may have tens of thousands of document sets shared with her.

To avoid these drawbacks, Popa et al. [29, 30] introduced the notion of *Multi-Key Searchable Encryption* (MKSE) to specifically address the query size problem. They provided a formal definition of MKSE along with a construction using bilinear maps, and an implementation of their scheme within a framework called *Mylar* for building secure web-applications over encrypted data. As part of the framework, they provide a number of prototype applications, including a chat room and a medical application.

The MKSE definition of [29, 30] aimed at capturing the following intuitive security guarantee: the scheme hides the content of *both* queries and stored documents, and the only information leaked is whether a given query matched any of the keywords in a given document. This should hold even when a subset of corrupted users colludes with the server. However, Grubbs et al. [17] showed that Mylar does not achieve this goal, and suffers from several serious security deficiencies, far beyond the limited leakage inherent in SSE. While some of these issues only apply to the particular design choices of Mylar and its applications, others are more general problems with the proposed MKSE definition. Recently, Van Rompay et al. [32] designed a different attack that pointed to another problem with the proposed MKSE definition. All these deficiencies remain present in follow up works that build on top of the MKSE definition of [29, 30], such as the work of Kiayias et al. [23]. We outline the three main issues below.

Separating Data and Query Privacy. The first issue with the MKSE definition is that it *separately* defines *data privacy* and *query privacy*, although it is intuitively clear that data and query privacy are *inherently* intertwined. Indeed, the server learns which documents are returned in response to a query, so knowing the contents of these documents leaks information about the contents of the query, and vice versa. Therefore, the two properties cannot be meaningfully defined in isolation. This observation has already been made in the context of single-key SSE by Curtmola et al. [9], who showed that earlier definitions that separated data and query privacy did not meaningfully rule out trivially insecure schemes. The work of [17] gives analogous examples in the context of MKSE, showing that there are trivially insecure schemes which satisfy the proposed definition.

Malicious Users Sharing Data. The second issue with the MKSE definition is more subtle. If an honest user Alice shares her document with a malicious user Mallory, then clearly privacy of her document is inherently lost. This limitation is intuitive, and users know they *should not* share their data with people they do not trust. But if Mallory shares her document with an honest Bob, one does not (and should not) expect Bob’s security to be compromised. Unfortunately, [17] show that the proposed MKSE notion of [29,30] does not guarantee any privacy for Bob’s search queries in this scenario. In particular, the security game designed to capture query privacy in this setting [30, Definition 5.6] *explicitly* prevents the adversary from sharing documents with the honest user (whose queries the adversary is trying to learn). Not only is this issue overlooked in the definition, but it is actually inherent to the proposed MKSE syntax, so every construction which realizes this syntax (e.g., the follow-up works of [23,35]) *necessarily* inherits this flaw. According to the original MKSE definition, when Mallory shares a document with Bob, a *share key* Δ is generated. The share key Δ does not depend on Mallory’s set, and allows *any* query under Bob’s key to be transformed into a query under Mallory’s key. Therefore, a malicious server, colluding with Mallory, can use Δ to transform every query Bob makes into a query under Mallory’s key, and the transformed query can then be executed offline against a set of single-word documents containing the full dictionary. Thus, if Mallory shares a document with Bob, the server can (through this offline dictionary attack) recover *all keywords* Bob searched for.

Searching by Comparing Queries and Encrypted Keywords. The third issue is that the MKSE definition implicitly restricts the algorithmic structure of the scheme to encrypt each keyword in the document separately, and search in a document by comparing the given query to each of the encrypted keywords. Thus, a “hit” reveals not only that the query appears in the document, but also *which (encrypted) keyword it matched*. Van Rompay et al. [32] show that this allows the server to compare queries issued by different users (even if both users are honest), and encrypted keywords from different documents (when they match the same keyword token).

1.1 Our Contribution

In this work, we propose a new MKSE definition that does not suffer from the above issues. In particular, our definition *simultaneously* addresses data and query privacy in a holistic manner, explicitly considers malicious data owners that may share data with honest users, and prevents the adversary from comparing queries issued by different users and keywords from different documents. We then propose a simple construction which provably satisfies our definition using only Pseudo-Random Functions (PRFs). Queries in this scheme consist of a single PRF image, and searching in a document is constant-time, but the server storage overhead is high. In particular, each time a document is shared with a user, it is essentially replicated, causing the per-document server storage overhead to be linear in the number of users the document is shared with.

We initially conjectured that such overhead is inherent to achieving our stronger MKSE definition. However, we show that proving such a conjecture will be quite challenging, since it will require ruling out the existence of certain program obfuscators. Concretely, in Sect. 5, we construct an MKSE scheme that uses obfuscation (specifically, public-coin differing-input obfuscation [20]) and requires per-document server storage that is roughly the document size *plus* the number of users it is shared with. (The construction has constant query size and polynomial time search.) We view our construction as providing evidence that a more efficient construction may possibly achieve the stronger MKSE notion with optimal server storage overhead.

Overview of Our MKSE Definition. We consider users that can take on two types of roles: data owners and queriers. Data owners have a document they wish to share with some subset of the users. Each document has its own associated *data key* K^d , where the data owner “encrypts” the document using this key, and uploads the encrypted document to the server. Each user has a *query key* K^u that it uses to issue search queries. When a data owner shares a document d with a user u they create a *share key* $\Delta_{u,d}$ which depends on the keys K^u, K^d , as well as the encrypted document, and store $\Delta_{u,d}$ on the server. When a querier wants to search for some keyword, he “encrypts” the keyword using his query key K^u , and sends the resulting encrypted query to the server. For each document d that was shared with the user u , the server uses the share key $\Delta_{u,d}$ to execute the encrypted query over the encrypted document, and learns if the keyword is contained in that document. This allows the server to return all relevant documents the querier has access to and which contain the keyword.

The main syntactic difference between our notion, and the MKSE notion used in Mylar, is in how the share key $\Delta_{u,d}$ is generated. As noted above, the share key in Mylar depends only on the keys K^u, K^d , whereas in our notion it also depends on the encrypted document. By tying the share key to the document, we can ensure that each query can only be executed on *the specific documents that were shared with the querier*, rather than on arbitrary documents, *even if the server has the key K^d* .

To define security, we consider a *share graph* between data owners (documents) and queriers, representing who shares data with whom, where some subset of data owners are malicious and collude with the server. The desired security guarantee is that the server learns nothing about the contents of the documents belonging to the honest data owners, or the keywords being queried, beyond the *access pattern* of which documents are returned by each query (i.e., out of the documents shared with the querier, which ones contain the queried keyword). We provide an *indistinguishability-based* definition where the adversary chooses the documents and data keys belonging to the malicious data owners, and two potential values (a “left” and a “right” value) for each query and each document belonging to an honest data owner. The left and right values must lead to the same access pattern, and the queries of each querier must be distinct. The adversary then gets all encrypted documents, share keys, and encrypted queries, and should not be able to distinguish whether these were created using the left or right values.

Since the adversary only learns the access pattern of which documents are returned by each query, the above definition captures the *minimal* leakage for schemes that reveal the access pattern, which seems to be the case in all practical schemes. This is a significant qualitative improvement over the leakage allowed by the previous definition of [30] and the corresponding schemes. Most importantly, when a malicious user Mallory is colluding with the sever and shares some data with Bob, the previous schemes completely leaked the contents of Bob’s query whereas our definition still only reveals the access pattern. We note that similar to single-key SSE, leaking the access pattern does reveal some potentially sensitive information and in some scenarios (e.g., when combined with auxiliary information about the documents) this may allow a sufficiently powerful attacker to completely recover the query, as shown in the single-key SSE setting by the recent works [4, 18, 22, 28, 31, 36]. In the multi-key setting this might be amplified since, whenever malicious data owners share documents with honest querier, the adversary already knows (and even chooses) the contents of these documents and hence can learn more information by seeing which of these documents match the query. For example, if the shared documents correspond to every individual word in a dictionary, then by seeing which document matches a given query the contents of the query are completely revealed. However, this is not a very natural scenario, and in many settings it is reasonable to believe that leaking the access pattern alone may not reveal significant information about the query. Furthermore, users can perform sanity checks on the documents shared with them to test how much leakage the server will get on their queries, and refuse to accept shared documents if they lead to too much leakage. Understanding when access pattern leakage is acceptable and when it is not is a fascinating and important direction for future study.

One implementation concern in the above notion of MKSE comes from how the share key $\Delta_{u,d}$ is generated, since it relies on knowledge of both the data-owner’s key K^d for the document being shared, the user’s querier key K^u , and the encrypted document itself. We envision that the data owner simply sends

the data key K^d to the querier via a secure channel. The querier then downloads the encrypted document from the server, generates $\Delta_{u,d}$, and uploads it to the server. Note that the querier can also check at this point that the document was encrypted correctly, and therefore in the security definition we always assume that documents are encrypted honestly.

Finally, our default definition is *selective*, meaning the adversary specifies the entire share graph, the data, and the queries ahead of time. We can also consider *adaptive* security for SSE (a notion introduced by [9] in the single-user setting) in which the adversary generates queries on the fly during the course of the attack. Furthermore, our definition is *indistinguishability based*, where one could also consider a *simulation-based* version (as introduced in the single-user setting by [9]) in which the simulator, given the share graph, document sizes, and access patterns, produces the encrypted documents and queries. We discuss these alternate variants in Sect. 6, and note that our PRF-based construction described below satisfies the strongest security notion (adaptive, simulation-based) when the PRF is instantiated in the random-oracle model.

Overview of the PRF-Based Construction. We provide a simple and efficient MKSE scheme based only on the existence of one-way functions. As noted above, each share key $\Delta_{u,d}$ contains a copy of the document, which allows Search to use only this value (and not the encrypted document).

If $\Delta_{u,d}$ is “allowed” to encode the entire document, a natural approach is to assign to each querier a PRF key K^u for a PRF F , and store in $\Delta_{u,d}$ the images of $F(K^u, \cdot)$ on all keywords in the document. However, this construction is fundamentally insecure, since the share keys themselves leak information, *even if the querier never makes any queries, and even if all data owners are honest*. More specifically, consider the case of two honest data owners that share their documents with an honest querier. Then the two share keys reveal the number of keywords that appear in both documents. This is because the token associated with each keyword depends only on the keyword and the querier key, and *not* on the document.

To solve this issue we use another layer of PRF images, where the first layer generates PRF keys for a second layer that will be applied to a document-specific random identifier. Concretely, when generating $\Delta_{u,d}$, we assign a random value r to the document. For every keyword w in the document, we generate a (second layer) PRF key $k^w = F(K^u, w)$, and compute a token t^w for w as $t^w = F(k^w, r)$. Using perfect hashing [11], these tokens are inserted into a hash table to accelerate searching. The share key $\Delta_{u,d}$ consists of r and the hash table containing the tokens. (Notice that if r is chosen from a sufficiently large domain, then with overwhelming probability each document is assigned a *unique* identifier, and so the share keys associated with two documents reveal no information about their intersection.)

To search for keyword w in her documents, the querier sends the query $k^w = F(K^u, w)$ to the server. Searching for k^w in a document with $\Delta_{u,d} = (r, D')$ (where D' is a hash table of tokens) is performed by searching the hash table D' on the key $F(k^w, r)$. This query reveals no information about w . Notice that

the scheme uses the encrypted document only to generate $\Delta_{u,d}$, so the document can simply be encrypted with its own unique symmetric encryption key.

1.2 Related Work

Single User Schemes. First introduced by Song et al. [33], the notion of (single user) Searchable Encryption has been extensively studied in the last decade (see [3, 12] for a survey of many of these works). The first works (e.g., [6, 14, 33]) constructed schemes under several (simulation-based or indistinguishability-based) security definitions. These definitions separated the properties of query and data privacy, and were shown by [9] to be insecure (by a fairly simple attack). Curtmola et al. [9] also presented a unified definition that combined both properties.

Multi-User Schemes. In this model multiple users can issue queries to a single dataset which is encrypted under a single key that is known to all users. Consequently, most works in this setting focus on access control, and efficient revocation of querying privileges. Following the work of [9], who provided the first definitions and constructions, there have been three main approaches to enforcing access control across the documents: traditional access control mechanisms [10, 25, 37], broadcast encryption [9, 26], and attribute-based encryption [7, 24].

We emphasize that in the multi-*user* setting, there is a *single* dataset owned by a single data owner, so using such schemes in settings with multiple data owners would require instantiating the scheme separately for each dataset, and thus the query size would be *linear* in the number of datasets shared with the querier. This should be contrasted with the multi-*key* setting which is the focus of this work, in which users can search over multiple datasets by issuing a *single* query whose size is *independent* of the number of datasets being searched.

Multi-key Schemes. In this setting multiple users share data encrypted under their own keys, and search across the data shared with them by issuing a single query whose size is independent of the number of shared datasets. First introduced by Popa et al. [29], follow-up works that build on [29] focused on optimizing server storage, and eliminating the trusted party needed to distribute data and querier keys [23]; mitigating attacks in which a malicious data owner shares a dictionary dataset with the querier, by having the querier explicitly determine which data owners are allowed to share data with her [35]¹; and constructing schemes that are secure in restricted security models when honest data owners only share their documents with honest queriers, or when a single data owner has not shared his dataset with anyone else [35] (in both models, the server might be corrupted). We note that since these works use the syntax of [29] (in particular, share keys are generated independently of the shared set), the aforementioned attacks of [17] apply to these works as well.

¹ This functionality was also discussed in [29], but was not defined as part of the MKSE syntax.

Other Related Models. The notion of Key Aggregate Searchable Encryption (KASE), introduced by [8], considers a data owner who has several documents, each encrypted under a unique key. This allows data owners to share different subsets of their documents with different users. The goal is to grant search access to a subset of documents by providing *one* aggregate key whose length is independent of the number of documents (whereas in a naive solution, the key size would scale with the number of documents), and the querier issues *one* query for *every* subset of documents under the aggregate key (whereas in the MKSE setting, the user issues *a single query, regardless* of the number of documents shared with her). Thus, this model is fundamentally different from MKSE (as pointed out in [8]). We note that the construction of [8] is vulnerable to dictionary attacks (as shown by [23]).

2 Preliminaries

In the following, λ denotes a security parameter, and $\text{negl}(\lambda)$ denotes a function that is negligible in λ . We use \approx to denote computational indistinguishability, and $S \setminus T$ to denote the difference between sets S, T . We use $\Pr[E : E_1, \dots, E_n]$ to denote the probability of event E given events E_1, \dots, E_n . For strings $x = x_1 \dots x_n, y = y_1 \dots y_m$, $x \circ y$ denotes their concatenation, i.e., $x \circ y = x_1 \dots x_n y_1 \dots y_m$. We use standard cryptographic definitions of one-way functions (OWFs), one-way permutations (OWPs), collision resistant hash functions (CRHFs), pseudorandom functions (PRFs), and existentially-unforgeable signature schemes (see, e.g., [15, 16]).

3 Defining Multi-Key Searchable Encryption

In this section we define the notion of *Multi-Key Searchable Encryption* (MKSE) schemes. Intuitively, an MKSE scheme allows data owners to share their documents with queriers who can later query these documents under their own keying material, while preserving both *data* and *query privacy*. In the definition, documents are represented as *sets of keywords*, so searching in a document translates to checking set membership; see the discussion following the definition.

Definition 1 (Multi-Key Searchable Encryption). *We say that a tuple $(\text{DataKeyGen}, \text{QueryKeyGen}, \text{ProcessSet}, \text{Share}, \text{Query}, \text{Search})$ of PPT algorithms is a Multi-Key Searchable Encryption (MKSE) scheme for a universe \mathcal{U} , if the following holds.*

– *Syntax:*

- *DataKeyGen* takes as input the security parameter 1^λ , and outputs a data key K .
- *QueryKeyGen* takes as input the security parameter 1^λ , and outputs a query key K^u .
- *ProcessSet* takes as input a data key K and a set S , and outputs a processed set T .

- *Share* takes as input a data key K , a query key K^u , and a processed set T , and generates a user-specific share key Δ .
 - *Query* takes as input an element $w \in \mathcal{U}$ and a query key K^u , and outputs a query q .
 - *Search* takes as input a user-specific share key Δ , a query q , and a processed set T , and outputs $b \in \{0, 1\}$.
- **Correctness:** For every security parameter $\lambda \in \mathbb{N}$, data set $S \subseteq \mathcal{U}$, and element $w \in \mathcal{U}$:

$$\Pr \left[\begin{array}{l} K \leftarrow \text{DataKeyGen}(1^\lambda) \\ K^u \leftarrow \text{QueryKeyGen}(1^\lambda) \\ T \leftarrow \text{ProcessSet}(K, S) \\ \Delta \leftarrow \text{Share}(K, K^u, T) \\ q \leftarrow \text{Query}(K^u, w) \end{array} \right] \geq 1 - \text{negl}(\lambda)$$

where $b = 0$ if $w \notin S$, otherwise $b = 1$.

- **Security:** Every PPT adversary \mathcal{A} has only a $\text{negl}(\lambda)$ advantage in the following security game with a challenger \mathcal{C} :

1. \mathcal{A} sends to \mathcal{C} :

- A set $\mathcal{Q} = \{1, \dots, m\}$ of queriers, a set $\mathcal{D} = \{1, \dots, n\}$ of data owners, and a subset $\mathcal{D}_c \subseteq \mathcal{D}$ of corrupted data owners.
- For every $i \in \mathcal{D}_c$, a data key K_i .
- For every $i \in \mathcal{D}$, two sets $S_i^0, S_i^1 \subseteq \mathcal{U}$, where $|S_i^0| = |S_i^1|$ for $i \notin \mathcal{D}_c$, and $S_i^0 = S_i^1$ for $i \in \mathcal{D}_c$.
- A bipartite share graph $G = (\mathcal{Q}, \mathcal{D}, E)$.
- For every $j \in \mathcal{Q}$, two sequences of distinct keywords $(w_{j,1}^0, \dots, w_{j,k_j}^0)$ and $(w_{j,1}^1, \dots, w_{j,k_j}^1)$ (for some $k_j \in \mathbb{N}$), such that for every $i \in \mathcal{D}$, if $(j, i) \in E$ then for every $1 \leq l \leq k_j$, $w_{j,l}^0 \in S_i^0$ if and only if $w_{j,l}^1 \in S_i^1$.

2. \mathcal{C} performs the following:

- Chooses a random bit $b \leftarrow \{0, 1\}$.
- For each querier $j \in \mathcal{Q}$, generates $K_j^u \leftarrow \text{QueryKeyGen}(1^\lambda)$.
- For each data owner $i \in \mathcal{D} \setminus \mathcal{D}_c$, generates $K_i \leftarrow \text{DataKeyGen}(1^\lambda)$.
- For each set $S_i^b, i \in \mathcal{D}$, generates $T_i \leftarrow \text{ProcessSet}(K_i, S_i^b)$.
- For each edge $(j, i) \in E$, generates $\Delta_{j,i} \leftarrow \text{Share}(K_i, K_j^u, T_i)$.
- For each querier j and keyword $w_{j,l}^b, 1 \leq l \leq k_j$, generates a query $q_{j,l} \leftarrow \text{Query}(K_j^u, w_{j,l}^b)$.
- Sends $\{T_i\}_{i \in \mathcal{D}}$, $\{\Delta_{j,i}\}_{(j,i) \in E}$, and $(q_{j,1}, \dots, q_{j,k_j})_{j \in \mathcal{Q}}$ to \mathcal{A} .

3. \mathcal{A} outputs a guess b' , and its advantage is $\text{Adv}_{\mathcal{A}}(1^\lambda) = \frac{1}{2} - \Pr[b = b']$.

Discussion. In Definition 1, sets of keywords are shared with queriers, and share keys are generated by an honest party. Such schemes can be easily adapted to the setting in which *documents* are shared between users: each document d is associated with the set S of keywords it contains; and an encryption of d is stored alongside the processed set T_S . Searching for keywords in d is performed by

searching for the keyword in S , where if the search outputs 1 then the encryption of d is returned to the querier. Moreover, a trusted party is not needed to generate the share keys: we envision that each user will generate her own share keys whenever a document is shared with her. More specifically, when a data owner i shares a document d_i with a querier j , the data owner will send his data key K_i to the querier via a secure channel. The querier will then download the processed set T_i and the encryption of d_i from the server, and generate $\Delta_{j,i}$ herself. This use case also clarifies our assumption that sets are honestly processed: j can verify that T_i was honestly generated by processing S_i (which can be extracted from d_i) using K_i , and comparing to T_i . (Without loss of generality, ProcessSet is deterministic since any randomness can be provided in K_i .)

Notice that in our definition, the share key is (syntactically) “tied” to the set for which it was generated (Share depends not only on the data and query keys, but also on the processed set). This should be contrasted with the syntax used in [29, 30] in which the algorithm generating the share keys depends *only* on the data and query keys. Consequently, resultant schemes *inherently* guarantee no query privacy when malicious data owners share their sets with honest queriers (as discussed in the introduction). Indeed, when Δ is independent of the set then a malicious server colluding with the data owner can use the data key K to encrypt *any* set S of his choosing (in particular, a dictionary), then use Δ to search for the query in S . Since K was generated independently of the set, the correctness of the scheme guarantees that the output of search will be correct, and so the server can recover the queried keyword.

Similar to previous works in the field, our definition allows for some information leakage. Specifically, since the adversary is restricted to choosing sets and queries for which $w_{j,l}^0 \in S_i^0 \Leftrightarrow w_{j,l}^1 \in S_i^1$ for every $(j, i) \in E$ (see the last bullet in Step 1 of the security game), the scheme leaks the access pattern of which subset of documents is returned in response to each query. Additionally, since we require that each querier makes distinct queries, if a querier makes repeated queries then this might be leaked to the server. Finally, we note that our definition is *selective*: the adversary is required to specify in advance the sets, queries, and share graph. Possible extensions and generalizations include adaptive security, where the adversary can adaptively choose sets and queries, add edges to the share graph, and corrupt data owners; and simulation-based security, which guarantees that the view of every PPT adversary can be simulated given only the aforementioned leakage (namely, the access patterns and the sizes of the sets). We elaborate on these alternative definitions in Sect. 6.

4 MKSE with Fast Search

In this section we describe our MKSE scheme based on PRFs. Concretely, we will prove the following theorem.

Theorem 1 (MKSE (Sublinear Search)). *Assume that OWFs exist. Then there exists a secure MKSE scheme in which searching for keywords in a set S takes $\text{poly}(\lambda)$ time, where λ is the security parameter.*

Moreover, data and query keys, as well as queries, have length $\text{poly}(\lambda)$, and for a set S , its processed version and share keys have size $|S| \cdot \text{poly}(\lambda)$.

We first describe our construction, then analyze its properties.

Construction 1 (MKSE (Sublinear Search)). The construction uses a PRF F , and a symmetric-key encryption scheme (KeyGen , Enc , Dec), as building blocks.

- **DataKeyGen** (1^λ) outputs a symmetric key $K_{\text{SE}} \leftarrow \text{KeyGen}(1^\lambda)$.
- **QueryKeyGen** (1^λ) outputs a uniformly random PRF key $K_{\text{PRF}} \leftarrow \{0, 1\}^\lambda$.
- **ProcessSet** (K_{SE}, S) outputs $\text{Enc}_{K_{\text{SE}}}(S)$.
- **Share** ($K_{\text{SE}}, K_{\text{PRF}}, T = \text{Enc}_{K_{\text{SE}}}(S)$) operates as follows:
 - Generates a uniformly random string $r \leftarrow \{0, 1\}^\lambda$.
 - Decrypts $S \leftarrow \text{Dec}_{K_{\text{SE}}}(T)$.
 - Initializes $D = \emptyset$. For each keyword $w_i \in S$, computes $k'_i = F_{K_{\text{PRF}}}(w_i)$ and $d_i = F_{k'_i}(r)$, and adds d_i to D .
 - Inserts D into a perfect hash table [11] to obtain D' .
 - Outputs $\Delta = (r, D')$.
- **Query** (K_{PRF}, w) outputs $F_{K_{\text{PRF}}}(w)$.
- **Search** ($\Delta = (r, D'), q, T$) operates as follows:
 - Computes $d' = F_q(r)$.
 - Performs a hash table query on D' for d' , and outputs 1 if and only if d' was found.

The next claim states that Construction 1 is secure, and summarizes its parameters.

Claim 1. Assume that Construction 1 is instantiated with a PRF F , and a secure symmetric encryption scheme, then it is a secure MKSE scheme.

Moreover, data and query keys have length $\text{poly}(\lambda)$, and for a set S the processed set has size $|S| \cdot \text{poly}(\lambda)$. Furthermore, searching in a set S takes time $\text{poly}(\lambda)$ and queries have size $\text{poly}(\lambda)$.

Proof. The correctness of the scheme follows directly from the correctness of the underlying primitives, and its complexity follows directly from the construction and from the following theorem due to Fredman et al. [11].

Theorem 2 (Perfect hashing [11]). *Given a set D of n keys from a universe \mathcal{U} , there exists a method that runs in expected $O(n)$ time and constructs a lookup table D' of size $O(n)$ such that membership queries (i.e., given $x \in \mathcal{U}$, determine if $x \in S$) can be answered in constant time.*

We now argue that the scheme is secure.

For every $i \in \mathcal{D}$, let S_i^0, S_i^1 be the sets \mathcal{A} chose for data owner i , and let $\mathcal{W}_j^0, \mathcal{W}_j^1$ be the sets of queries \mathcal{A} chose for querier $j \in \mathcal{Q}$. Let F^1 denote the PRF called in Query and to compute k'_i in Share, and let F^2 be the PRF invoked to compute d_i in Share. Let $\text{view}_0, \text{view}_1$ denote the view of \mathcal{A} in the security game when $b = 0, 1$ (respectively). We show that $\text{view}_0 \approx \text{view}_1$ using a sequence of

hybrid distributions, and conclude that \mathcal{A} has only a $\text{negl}(\lambda)$ advantage in the security game. As the data keys, and encrypted sets, of corrupted data owners are identically distributed in both views (because $S_i^0 = S_i^1$ for every $i \in \mathcal{D}_c$), we can fix these values into $\text{view}_0, \text{view}_1$, and all hybrid distributions, without decreasing the computational distance. Moreover, if F is sufficiently expanding then with overwhelming probability, all images of the form $F_K^1(w)$, and $F_{k'_i}^2(r)$ (for query keys K , keywords w , set identifiers r , and k'_i) are distinct, so it suffices to bound the computational distance conditioned on this event. We now define the hybrids.

For $b \in \{0, 1\}$, let \mathcal{H}_0^b be the distribution obtained from view_b by replacing F^1 with a random function \mathcal{R} . (We think of \mathcal{R} as taking two inputs, the first being a querier index. Thus, \mathcal{R} defines a *family* $\{\mathcal{R}_j\}_j$ of functions, as does F^1 .) That is, for all queriers j , and $w_l \in \mathcal{W}_j^b$, we have $q'_{j,l} = \mathcal{R}(w_l)$ (the tag is used to denote queries in \mathcal{H}_0^b ; queries in view_b are untagged). Then $\text{view}_b \approx \mathcal{H}_0^b$ follows from the pseudorandomness of F by a standard hybrid arguments in which we replace the invocations of F^1 (used to generate the queries and share keys) of one querier at a time.

We now define \mathcal{H}_1^b to be identical to \mathcal{H}_0^b , except that F^2 is replaced with the random function \mathcal{R} (notice that here, the first input of \mathcal{R} corresponds to a query q'), and the keyword tokens in every share key $\Delta_{j,i}$ are generated as follows. For every $w_l \in S_i^b \cap \mathcal{W}_j^b$, the corresponding token $d_{i,j,l}$ is computed as $F_{q'_{j,l}}^2(r)$ (i.e., identically to how it is generated in \mathcal{H}_0^b ; this is needed since $q'_{j,l}$ appears in \mathcal{H}_1^b and so consistency of these tokens with F^2 can be efficiently checked). For every $w_l \in S_i^b \setminus \mathcal{W}_j^b$, $d'_{i,j,l}$ is chosen randomly subject to the constraint that $d'_{i,j,l} \notin \{F_{q'_{j,l'}}^2(r) : w_{l'} \in \mathcal{W}_j^b\}$. (This can be efficiently achieved by re-sampling, assuming F is sufficiently stretching.) All values in $\Delta_{j,i}$ are then hashed.

To show that $\mathcal{H}_0^b \approx \mathcal{H}_1^b$, we first define an intermediate hybrid $\mathcal{H}^{b,*}$ in which for every querier j , every data owner i , and every keyword $w_l \in S_i^b \setminus \mathcal{W}_j^b$, the token $d'_{i,j,l}$ in $\Delta_{j,i}$ is replaced with a random value, subject to the constraint that it is not in $\{F_{q'_{j,l'}}^2(r) : w_{l'} \in \mathcal{W}_j^b\}$, where r is the random identifier associated with $\Delta_{j,i}$. Then $\mathcal{H}_0^b \approx \mathcal{H}^{b,*}$ follows from the pseudorandomness of F by a standard hybrid argument in which we replace the tokens one at a time (and use the assumption that all images of F are unique).

To show that $\mathcal{H}_1^b \approx \mathcal{H}^{b,*}$, we define a series of sub-hybrids, replacing F^2 with a random function for a single query of a single querier at a time. (Notice that each query of each querier represents a unique key for F^2 in all share keys associated with that querier.) Concretely, denote $m = |\mathcal{Q}|$, and for every $j \in \mathcal{Q}$, let $l_j := |\mathcal{W}_j^b|$. For every $1 \leq j \leq m$ and $0 \leq l \leq l_j$, define $\mathcal{H}^{b,j,l}$ to be the distribution obtained from $\mathcal{H}^{b,*}$ by generating the queries of the first $j-1$ queriers, and the first l queries of querier j , with \mathcal{R} (instead of F^2), and generating the keyword tokens in share keys accordingly. Then $\mathcal{H}^{b,1,0} = \mathcal{H}^{b,*}$, and $\mathcal{H}^{b,m,l_m} = \mathcal{H}_1^b$. For every $1 \leq j \leq m$ and $1 \leq l \leq l_j$, $\mathcal{H}^{b,j,l} \approx \mathcal{H}^{b,j,l-1}$ by the

pseudorandomness of F^2 . Moreover, $\mathcal{H}^{b,j,0} = \mathcal{H}^{b,j-1,l_j-1}$ for every $1 < j \leq m$, so $\mathcal{H}^{b,1,0} \approx \mathcal{H}^{b,m,l_m}$ (since $m = \text{poly}(\lambda)$, and $l_j = \text{poly}(\lambda)$ for every $j \in \mathcal{Q}$).

Finally, let \mathcal{H}_2^b be identical to \mathcal{H}_1^b except that the encrypted sets of all honest data owners $i \notin \mathcal{D}_c$ encrypt $\mathbf{0}$ (instead of S_i^b). Then $\mathcal{H}_1^b \approx \mathcal{H}_2^b$ follows from the security of the encryption scheme by a standard hybrid argument in which the encrypted sets are replaced one at a time. Notice that $\mathcal{H}_2^0 = \mathcal{H}_2^1$ and so $\text{view}_0 \approx \text{view}_1$. \square

Remark 1. Notice that \mathcal{H}_2^b depends only on the share graph, the sets of corrupted data owners, the *sizes* of sets of honest data owners, and the access patterns; and can be efficiently generated given these values. This implies that the view of every PPT adversary can be efficiently simulated given only these values, namely, Construction 1 is *simulation*-secure (as defined in Sect. 6).

The proof of Theorem 1 now follows as a corollary from Claim 1.

Proof of Theorem 1. We instantiate Construction 1 with any sufficiently stretching PRF (e.g., $F : \{0, 1\}^\lambda \times \{0, 1\}^n \rightarrow \{0, 1\}^{2(\lambda+n)}$), whose existence follows from the existence of OWFs), and a secure symmetric encryption scheme (which can be constructed from F). Then the security of the scheme, as well as the length of data keys, query keys, and processed sets, follow directly from Claim 1. As for search time, since keywords have length $O(\lambda)$ then evaluating F takes $\text{poly}(\lambda)$ time, and the outputs have length $\text{poly}(\lambda)$. Searching in a set S takes 1 hash query and thus the overall time is $\text{poly}(\lambda)$ time, and queries have length $\text{poly}(\lambda)$. \square

5 MKSE with Short Share Keys

In this section we describe an MKSE scheme with short share keys which employs a program obfuscator as a building block. We first show (Sect. 5.1) a scheme based on differing-inputs obfuscation (diO), then show (Sect. 5.2) that a slightly modified construction can be based on *public-coin* differing-inputs obfuscation (pc-diO). We note that though there is evidence that *diO* for *general* circuits might not exist [2, 13], no such implausibility results are known for *pc-diO*.

5.1 MKSE from Differing-Inputs Obfuscation

We construct a secure MKSE scheme using a diO obfuscator for Turing Machines (TMs). Concretely, we prove the following for a universe \mathcal{U} of size $|\mathcal{U}| \leq \text{poly}(2^\lambda)$:

Theorem 3 (MKSE (Short Share Keys)). *Assume that CRHFs, and diO for TMs with polynomial blowup, exist. Then there exists a secure MKSE in which share keys have size $\text{poly}(\lambda)$, for a security parameter λ . Moreover, data and query keys, as well as queries, have length $\text{poly}(\lambda)$, and given a set S , its processed version has size $|S| \cdot \text{poly}(\lambda)$, and searching in it takes $\text{poly}(\lambda, |S|)$ time.*

The high-level idea of the constructions is to encrypt sets under their data key, and queries under the query key of the querier, using a standard (symmetric) encryption scheme. The share key will be an obfuscation of a program that has both keys hard-wired into it, and thus allows for searching (even though queries and sets are encrypted under different keys) by decrypting the ciphertexts and comparing the underlying keywords. However, to make this rough intuition work, we need to handle a few subtleties.

First, to obtain security, the program should take *the entire set* as input. Otherwise (i.e., if it operates on a single set element at a time), its output would reveal not only *whether* the queried keyword appears in the set, but also *where* it appears. To see why this violates security, consider the case in which the same set S_i is shared with two different queriers j and j' : the additional information of where in the set a query appears allows the server to check whether j and j' queried the same keyword (even when j, j' and data owner i are all honest). Notice that since the program takes the entire set as input, it cannot be represented as a circuit (since then share keys will not have sublinear size). Therefore, we implement the program as a *TM*, and use an obfuscator for TMs.

Second, as noted in the introduction, share keys should only allow searching for keywords *in the sets for which they were generated*. That is, a share key $\Delta_{j,i}$ between querier j and data owner i should be “tied” to the set S_i of i . We achieve this by hard-wiring a hash $h(S_i)$ of S_i into the program $P_{j,i}$ obfuscated in $\Delta_{j,i}$, where $P_{j,i}$ checks that its input set is consistent with the hash. Notice that the hard-wired hash prevents us from using an indistinguishability obfuscator [1]. Indeed, if i is honest then in the security game (Definition 1), the adversary chooses a pair S_i^0, S_i^1 of (possibly different) sets for i , and $\Delta_{j,i}$ has either $h(S_i^0)$ (in the game with $b = 0$) or $h(S_i^1)$ (in the game with $b = 1$) hard-wired into it. In particular, the underlying programs are not functionally equivalent, so we cannot rely on indistinguishability obfuscation, and need to use a stronger primitive. Concretely, our constructions rely on the existence of a diO, or a pc-diO, obfuscator. We proceed to describe the diO-based construction (the pc-diO-based construction is described in Sect. 5.2).

The last ingredient we need is a signature scheme, which will be used to sign queries. Specifically, a query for keyword w will consist of an encryption c of w , and a signature on c ; and share keys will have the corresponding verification key hard-wired into them. Intuitively, signatures are used to guarantee the server can only search for queries the querier *actually* issued, similar to the way the hashed set prevents the server from searching in sets that were not shared with the querier. Concretely, the signatures guarantee that the share keys in the security game for $b = 0$ and $b = 1$ are differing-inputs even given the entire view of the adversary, and allows us to rely on diO security. (Roughly, a pair of programs are differing-inputs if it is infeasible for a PPT algorithm to find an input on which their outputs differ.)

Remark 2. We note that if one is willing to change the MKSE syntax, allowing the server to return *encrypted* answers which the querier then decrypts, then a scheme with similar complexity could be constructed from Fully Homomorphic

Encryption (using Oblivious RAM or Private Information Retrieval). However, following previous works in the field [17, 23, 29] we focus on the setting in which the server gets the answers in the clear (and queriers do not need to decrypt). This may be crucial in some situations, e.g., when huge documents are associated with small keyword sets. In a solution based on Fully Homomorphic Encryption, the computation of a search is proportional to the total size of *all the huge documents*, while in our obfuscation-based MKSE scheme the work is proportional to the total number of *keywords*, and the size of the *returned* documents.

We now describe our MKSE scheme. As will become evident from the security proof, due to the technicalities of using diO security we will need a special type of encryption (which, nonetheless, can be constructed from any standard encryption scheme) that we call *double encryption*. It is similar to the encryption scheme used in the “2-key trick” of Naor and Yung [27] (to convert a CPA-secure encryption scheme into a CCA-secure one), except it does not use non-interactive zero-knowledge proofs to prove that ciphertexts encrypt the same value.

Definition 2 (Double encryption). *Let $\lambda \in \mathbb{N}$ be a security parameter. Given a symmetric encryption scheme $(\text{KeyGen}, \text{Enc}, \text{Dec})$, we define a double symmetric encryption scheme $\text{E}^2 = (\text{KeyGen}^2, \text{Enc}^2, \text{Dec}^2)$ as follows:*

- KeyGen^2 , on input 1^λ , generates $K_L \leftarrow \text{KeyGen}(1^\lambda)$ and $K_R \leftarrow \text{KeyGen}(1^\lambda)$, and outputs $K = (K_L, K_R)$.
- Enc^2 , on input a key $K = (K_L, K_R)$ and a message m , computes $c_L \leftarrow \text{Enc}(K_L, m)$ and $c_R \leftarrow \text{Enc}(K_R, m)$, and outputs $c = (c_L, c_R)$.
- Dec^2 , on input a key $K = (K_L, K_R)$ and a ciphertext $c = (c_L, c_R)$, outputs $\text{Dec}(K_L, c_L)$. (Notice that decryption disregards the “right” component of c .)

We are now ready to describe our MKSE scheme.

Construction 2 (MKSE (Short Share Keys)). The MKSE uses the following building blocks:

- an obfuscator \mathcal{O} ,
- a hash function h ,
- a double symmetric encryption scheme $(\text{KeyGen}, \text{Enc}, \text{Dec})$, and
- a signature scheme $(\text{KeyGen}_s, \text{Sign}, \text{Ver})$,

and is defined as follows:

- **DataKeyGen** (1^λ) generates a random encryption key $K \leftarrow \text{KeyGen}(1^\lambda)$ and outputs K .
- **QueryKeyGen** (1^λ) generates a random encryption key $K^u \leftarrow \text{KeyGen}(1^\lambda)$, and a random signing and verification key pair $(\text{sk}^u, \text{vk}^u) \leftarrow \text{KeyGen}_s(1^\lambda)$, and outputs $K^u = (K^u, \text{sk}^u, \text{vk}^u)$.
- **ProcessSet** (K, \mathcal{S}) encrypts each element $s \in \mathcal{S}$ as $c(s) \leftarrow \text{Enc}(K, s)$, and outputs $\{c(s) : s \in \mathcal{S}\}$.
- **Share** $(K, K^u = (K^u, \text{sk}^u, \text{vk}^u), T_S)$ generates $\tilde{P} \leftarrow \mathcal{O}(P_{K, (K^u, \text{vk}^u), h(T_S)})$ where $P_{K, (K^u, \text{vk}^u), h(T_S)}$ is the TM defined in Fig. 1, and outputs $\Delta = \tilde{P}$.

- **Query** ($K^u = (\mathbf{K}^u, \mathbf{sk}^u, \mathbf{vk}^u), w$) generates $c \leftarrow \text{Enc}(\mathbf{K}^u, w)$ and $\sigma \leftarrow \text{Sign}(\mathbf{sk}^u, c)$, and outputs (c, σ) .
- **Search** (Δ, q, T_S) outputs $\Delta(T_S, q)$.

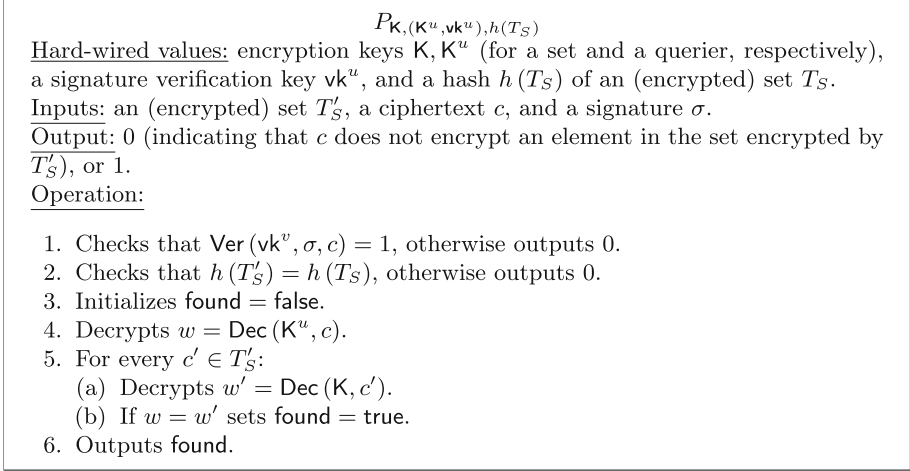


Fig. 1. Program $P_{\mathbf{K}, (\mathbf{K}^u, \mathbf{vk}^u), h(T_S)}$ used to generate share keys in Construction 2

The following claim states that if the obfuscator \mathcal{O} used in Construction 2 is a secure diO obfuscator, and all building blocks are secure, then Construction 2 is an MKSE scheme (as in Definition 1).

Claim 2 (MKSE (Short Share Keys)). If Construction 2 is instantiated with a secure diO obfuscator for TMs, and assuming the security of all building blocks, then Construction 2 is a secure MKSE.

Moreover, if the encryption and signature schemes have $\text{poly}(\lambda)$ -length keys, and incur a $\text{poly}(\lambda)$ overhead, then data and query keys, as well as queries, have length $\text{poly}(\lambda)$, and for a set S , its corresponding processed set has size $|S| \cdot \text{poly}(\lambda)$. Furthermore, if: (1) evaluating h on length- n inputs takes $H_T(n)$ time, and outputs a hash of length $H_\ell(n)$; and (2) there exist functions $s, T_{\mathcal{O}} : \mathbb{N} \rightarrow \mathbb{N}$ such that for every TM M , $|\mathcal{O}(M)| \leq s(|M|)$, and running $\mathcal{O}(M)$ on inputs of length n takes $T_{\mathcal{O}}(\text{TIME}(M, n))$ time, where $\text{TIME}(M, n)$ is the running time of M on length- n inputs; then running Search on a set S takes $T_{\mathcal{O}}(H_T(|S| \cdot \text{poly}(\lambda)) + |S| \cdot \text{poly}(\lambda))$ time, and share keys have size $s(H_\ell(|S| \cdot \text{poly}(\lambda)) \cdot \text{poly}(\lambda))$.

Remark 3. We note that the security of Construction 2 does not require the diO obfuscator to be secure with relation to *arbitrary* auxiliary inputs, but rather it is only required to guarantee security against a specific class of auxiliary inputs, as specified in the proof of Claim 2.

Proof of Claim 2. The correctness of the scheme follows directly from the correctness of the underlying primitives. We now argue that the scheme is secure.

Let \mathcal{A} be a PPT adversary in the security game of Definition 1, let K_i be the data key \mathcal{A} chose for data owner i , and let $\mathcal{W}^0, \mathcal{W}^1$ be the sets of queries \mathcal{A} chose (for all other values chosen by \mathcal{A} , we use the notation of Definition 1). We proceed through a sequence of hybrids. Recall that the view of \mathcal{A} in the security game consists of the encrypted sets $T_{S_i^b}$ for every $i \in \mathcal{D}$, queries q for every $w \in \mathcal{W}^b$, and for every edge $(j, i) \in E$, the obfuscated program $\Delta_{j,i}$. In particular, since the keys, and encrypted sets, of corrupted data owners are identically distributed when $b = 0, 1$ (because $S_i^0 = S_i^1$ for every $i \in \mathcal{D}_c$, and they are encrypted using the same keys), we can fix these values into all hybrid distributions, without decreasing the computational distance. Moreover, we assume without loss of generality that all data keys K_i chosen by \mathcal{A} are valid. We now define the hybrids.

view₀: view₀ is the view of \mathcal{A} in the security game with $b = 0$.

\mathcal{H}_0 : In hybrid \mathcal{H}_0 , the keys $K = (K_L, K_R), K^u = (K_L^u, K_R^u)$ in every obfuscated program $P_{K, (K^u, vk^u), h(T_{S^0})}$ are replaced with the keys $K' = (K_L, \mathbf{0}), K^{u'} = (K_L^u, \mathbf{0})$.

$\mathcal{H}_0 \approx \text{view}_0$ by the diO security of \mathcal{O} (and a standard hybrid argument over all obfuscated programs in $\text{view}_0, \mathcal{H}_0$), because the TMs in each of the share keys $\Delta_{j,i}$ in $\text{view}_0, \mathcal{H}_0$ are differing-inputs. Indeed, they are actually functionally equivalent (given *any* auxiliary input), since Dec^2 completely disregards the right ciphertext, and so replacing the right secret key with the all-0 string does not affect functionality.

\mathcal{H}_1 : In hybrid \mathcal{H}_1 , the encrypted set T_i of every honest data owner $i \notin \mathcal{D}_c$ is generated as the encryption of (S_i^0, S_i^1) with Enc^L (see Definition 3 below) instead of Enc^2 . (Notice that this also affects the share keys.)

To prove that $\mathcal{H}_0 \approx \mathcal{H}_1$, we will use the following lemma.

Lemma 1. *Let $\star \in \{L, R\}$. For every pair (m_L, m_R) of messages, the following distributions are computationally indistinguishable, when E^2, E^* use the same underlying encryption scheme $E = (\text{KeyGen}, \text{Enc}, \text{Dec})$.*

- \mathcal{D}_1 : generate $K = (K_L, K_R) \leftarrow \text{KeyGen}^2(1^\lambda)$ and $c \leftarrow \text{Enc}^2(K, m_\star)$, and output (K_\star, c) .
- \mathcal{D}_2 : generate $K = (K_L, K_R) \leftarrow \text{KeyGen}^*(1^\lambda)$ and $c \leftarrow \text{Enc}^*(K, (m_L, m_R))$, and output (K_\star, c) .

Proof. We prove the lemma for the case $\star = L$ (the case $\star = R$ is similar) by showing that indistinguishability follows from the security of the underlying scheme E . Given a distinguisher D between $\mathcal{D}_1, \mathcal{D}_2$, we construct a distinguisher D' (that has m_L hard-wired into it) between encryptions according to E of m_L, m_R . Given a ciphertext c , D' operates as follows: generates $K' \leftarrow \text{KeyGen}(1^\lambda)$, computes $c' \leftarrow \text{Enc}(K', m_L)$, and outputs $D(K', (c', c))$. Notice that if c encrypts m_L then the input to D is distributed according to \mathcal{D}_1 , otherwise it is distributed according to \mathcal{D}_2 , so the distinguishing advantage of D' is equal to that of D which (by the security of E) is negligible. \square

By a standard hybrid argument, Lemma 1 implies that polynomially many ciphertexts (generated by E^2 or by E^L , with the same or different keys), together with the keys of the “left” component, are computationally indistinguishable.

We prove $\mathcal{H}_0 \approx \mathcal{H}_1$ by reducing any distinguisher D between $\mathcal{H}_0, \mathcal{H}_1$ to a distinguisher D' between encryptions generated according to E^L or E^2 . We hard-wire into D' the querier keys and their queries, as well as the keys and encrypted sets of corrupted data owners, and the share keys associated with them. (This is possible because the encryption and signing keys of queriers, and their queries, are identically distributed in $\mathcal{H}_0, \mathcal{H}_1$, and independent of the encrypted sets; and since the share keys $\Delta_{j,i}$ for $i \in \mathcal{D}_c$ depend only on the keys of data owner i , querier j , and the encrypted set T_i , which are identically distributed in both hybrids.) D' operates as follows: given a sequence of ciphertexts (the encrypted sets of honest data owners), and the keys corresponding to the ciphertexts in the left components, D' honestly generates the hashes of encrypted sets of honest data owners, and uses the hard-wired querier keys, together with the keys for the left component in the ciphertexts of honest data owners, to generate the share keys between queriers and honest data owners. (Notice that since we have removed the key of the right component in ciphertexts of honest data owners, these are not needed to generate the share keys.) The values obtained in this way are distributed identically to \mathcal{H}_0 (if the input ciphertexts were generated with E^2) or \mathcal{H}_1 (if they were generated with E^L), so D' has the same distinguishing advantage as D .

We now define the next hybrids.

\mathcal{H}_2 : In hybrid \mathcal{H}_2 the queries $(w_{j,1}^0, \dots, w_{j,k_j}^0)$ of every querier $j \in \mathcal{Q}$ are generated using Enc^L with message $(w_{j,l}^0, w_{j,l}^1), 1 \leq l \leq k_j$, instead of Enc^2 .

$\mathcal{H}_1 \approx \mathcal{H}_2$ by a similar argument to the one used to show $\mathcal{H}_0 \approx \mathcal{H}_1$.

\mathcal{H}_3 : In hybrid \mathcal{H}_3 , the generation of share keys $\Delta_{j,i}$ is modified as follows: (1) the hard-wired keys are $(\mathbf{0}, K_{R,i}), (\mathbf{0}, K_{R,j})$, where $(K_{L,i}, K_{R,i}), (K_{L,i}^u, K_{R,j}^u)$ are the encryption keys of data owner i and querier j , respectively; and (2) the program P uses Dec^R (instead of Dec^2) to decrypt c and T'_S .

$\mathcal{H}_3 \approx \mathcal{H}_2$ by the diO security of \mathcal{O} , as we now show. Let d denote the number of share keys available to the adversary (i.e., $d = |E|$), and order them in some arbitrary way: $\Delta^1, \dots, \Delta^d$. We define a sequence of hybrids $\mathcal{H}^0, \dots, \mathcal{H}^d$, where in \mathcal{H}^l , the first l share keys are generated as in \mathcal{H}_3 (we denote these keys by Δ'_k), and all other share keys are generated as in \mathcal{H}_2 . We show that $\mathcal{H}^l \approx \mathcal{H}^{l-1}$ for every $1 \leq l \leq d$, and conclude that $\mathcal{H}_2 = \mathcal{H}^0 \approx \mathcal{H}^d = \mathcal{H}_3$.

Fix some $1 \leq l^* \leq d$, and let (j, i) be the edge for which Δ^{l^*} was generated. We fix all the keywords $\mathcal{W}^0, \mathcal{W}^1$, and the sets $S_{i'}^0, S_{i'}^1$ for $i' \in \mathcal{D}$, into $\mathcal{H}^{l^*}, \mathcal{H}^{l^*-1}$ (this is possible because these values are identical in both hybrids). We additionally fix the keys of every querier $j', j' \neq j$ and data owner $i', i' \neq i$, the queries that querier j' makes, the processed sets $T_{i'}, i' \neq i$, and share keys $\Delta_{j',i'}, \Delta'_{j',i'}$ (this is possible because these values are identically distributed in both hybrids). We now argue that $\Delta'_{j,i}, \Delta_{j,i}$ sampled in $\mathcal{H}^{l^*}, \mathcal{H}^{l^*-1}$ (respectively)

form a differing-inputs family of TMs with respect to the auxiliary information aux available to \mathcal{A} (and so $\mathcal{H}^{l^*} \approx \mathcal{H}^{l^*-1}$ by the diO security of \mathcal{O}). This auxiliary information consists of the values we have fixed into $\mathcal{H}^{l^*}, \mathcal{H}^{l^*-1}$, the public verification key vk_j^u for signatures of querier j , all queries querier j makes, the encrypted set T_i of data owner i , and all $\Delta_{j,i'}, \Delta_{j',i}$ for $i' \neq i, j' \neq j$ such that $(j, i'), (j', i) \in E$. Let $\mathcal{P}, \mathcal{P}'$ denote the distributions over programs obfuscated in $\Delta_{j,i}, \Delta'_{j,i}$, and let D be a PPT algorithm that obtains $(P, P') \leftarrow \mathcal{P} \times \mathcal{P}'$ and aux . In particular, notice that D knows the hash $h(T_i)$, and the encryption keys K_i, K_j^u (but not the secret signing key sk_j^u), since these appear in either P or P' . We show that D succeeds in finding a differing input only with negligible probability.

Consider first the inputs (for P, P') available to D in aux , i.e., the encrypted set T_i (which encrypts the elements of S_i^0 in the left components, and the elements of S_i^1 in the right components; this holds even if $i \in \mathcal{D}_c$ since in that case $S_i^0 = S_i^1$), and the queries of querier j . For every such query q there exists a pair (w_L, w_R) of keywords such that q is of the form $q = (c = (c_L, c_R), \sigma)$ where $c_\star \leftarrow \text{Enc}(K_{\star,j}^u, m_\star)$, $\star \in \{L, R\}$, $\sigma \leftarrow \text{Sign}(\text{sk}_j^u, c)$, and $w_L \in S_i^0 \Leftrightarrow w_R \in S_i^1$. (Here, $(\text{KeyGen}, \text{Enc}, \text{Dec})$ is the encryption scheme used as a building block in the double encryption scheme.) In particular, $P(q, T_i) = P'(q, T_i)$ since the checks in Steps (1)–(2) succeed in both cases, and Steps (4)–(5) return the same outcome (P searches for w_L in S_i^0 , since it decrypts using Dec^2 , whereas P' searches for w_R in S_i^1 , since it decrypts with Dec^R and the right component of T_i encrypts S_i^1 ; and $w_L \in S_i^0 \Leftrightarrow w_R \in S_i^1$). In particular, both programs have the same running time in this case.

Next, we claim that for every other possible input (T', c', σ') that D chooses (and does not appear in aux), $P(T', c', \sigma') = P'(T', c', \sigma') = 0$ except with negligible probability. Indeed, if $(c', \sigma') \neq q$ (where q is some query of querier j) then by the existential unforgeability of the signature scheme, with overwhelming probability σ' is not a valid signature for c' , so the check in Step (1) fails in both P, P' (in particular, both programs have the same running time in this case). Moreover, if $T' \neq T_i$ then by the collision resistance of h , with overwhelming probability $h(T') \neq h(T_i)$, so the check in Step (2) fails (and again, P, P' have the same running time). Therefore, P, P' are differing inputs with relation to the auxiliary information aux .

We now define the final set of hybrids.

\mathcal{H}_4 : In hybrid \mathcal{H}_4 , the queries $(w_{j,1}^1, \dots, w_{j,k_j}^1)$ of every querier $j \in \mathcal{Q}$ are generated using Enc^2 with message $(w_{j,l}^1, w_{j,l}^1)$, $1 \leq l \leq k_j$, instead of Enc^L with message $(w_{j,l}^0, w_{j,l}^1)$, $1 \leq l \leq k_j$.

$\mathcal{H}_3 \approx \mathcal{H}_4$ by a similar argument to the one used to prove $\mathcal{H}_1 \approx \mathcal{H}_2$.

\mathcal{H}_5 : In hybrid \mathcal{H}_5 , the encrypted set T_i of every honest data owner $i \notin \mathcal{D}_c$ is generated as the encryption of (S_i^1, S_i^1) with Enc^2 instead of with Enc^L .

$\mathcal{H}_4 \approx \mathcal{H}_5$ by a similar argument to the one used to prove $\mathcal{H}_0 \approx \mathcal{H}_1$.

\mathcal{H}_6 : In hybrid \mathcal{H}_6 , the obfuscated program for every edge $(j, i) \in E$ is generated as follows: (1) the hard-wired keys are $K = (K_{L,i}, K_{R,i}), K = (K_{L,j}^u, K_{R,j}^u)$,

instead of $K' = (\mathbf{0}, K_{R,i}), K^{u'} = (\mathbf{0}, K_{R,j}^u)$; and (2) Dec^2 is used for decryption (instead of Dec^R).

$\mathcal{H}_6 \approx \mathcal{H}_5$ by the diO security of \mathcal{O} , using a standard hybrid argument in which the obfuscated programs are replaced one at a time. When replacing the program for edge (j, i) from P' (in \mathcal{H}_5) to P (in \mathcal{H}_6), the PPT D (which should find a differing input) is given (as part of the auxiliary information aux) the sets $S_{i'}^1, i' \in \mathcal{D}$; the keywords in \mathcal{W}^1 and the corresponding queries; the encryption keys of all data owners $i', i' \neq i$ and querier $j', j' \neq j$; the signing and verification keys of all queriers $j', j' \neq j$; and all encrypted sets $T_{i'}, i' \in \mathcal{D}$. Also, from P, P' the distinguisher learns the encryption keys K_i, K_j^u , and the verification key vk_j^u . We show that except with negligible probability, D fails to find a differing input (the argument is similar to that used to prove $\mathcal{H}_3 \approx \mathcal{H}_2$).

The inputs (for P, P') available to D in aux , i.e., the encrypted set T_i , and queries $q = (c, \sigma)$ of querier j (where $c \leftarrow \text{Enc}^2(K_j^u, m)$ for some m , and $\sigma \leftarrow \text{Sign}(\text{sk}_j^u, c)$), are not differing-inputs (even though P' decrypts the right component of c , whereas P decrypts the left component) because both components of c encrypt m according to Enc (so both P, P' search for m in S_i^1). For every other possible input (T', c', σ') that D chooses, $P(T', c', \sigma') = P'(T', c', \sigma') = 0$ except with negligible probability: if $(c', \sigma') \neq q$ (where q is some query of querier j) then with overwhelming probability σ' is not a valid signature on c' (by the existential unforgeability of the signature scheme); whereas if $T' \neq T_i$ then with overwhelming probability $h(T') \neq h(T_i)$ (by the collision resistance of h). Consequently, $\mathcal{H}_5 \approx \mathcal{H}_6$. Since \mathcal{H}_6 is the view of \mathcal{A} in the security game with $b = 1$, we conclude that \mathcal{A} has only negligible advantage in the security game.

Finally, we analyze the complexity of the scheme. Data and query keys, which are simply encryption and signing keys, have size $\text{poly}(\lambda)$. Queries are ciphertext for length- $O(\lambda)$ keywords (since the universe is at most of size 2^λ), together with signatures on these ciphertexts. Similarly, a processed set consists of encryptions of each of its keywords, so its size is $|S| \cdot \text{poly}(\lambda)$. Regarding share keys, the TM has size $H_\ell(|S| \cdot \text{poly}(\lambda)) \cdot \text{poly}(\lambda)$ (since $|h(T_S)| \leq H_\ell(|S| \cdot \text{poly}(\lambda))$), and so by the assumption on the blowup caused by obfuscation, share keys have size $s(H_\ell(|S| \cdot \text{poly}(\lambda)) \cdot \text{poly}(\lambda))$. Finally, Search consists of running the obfuscated TM, which requires computing the hash ($H_T(|S| \cdot \text{poly}(\lambda))$ time), and performing $O(|S|)$ operations, each taking $\text{poly}(\lambda)$ time, so $\text{TIME}(M, |S|) = H_T(|S| \cdot \text{poly}(\lambda)) + |S| \cdot \text{poly}(\lambda)$, and consequently the running time is $T_{\mathcal{O}}(H_T(|S| \cdot \text{poly}(\lambda)) + |S| \cdot \text{poly}(\lambda))$. \square

The following encryption scheme was used to prove Claim 2:

Definition 3. *Given a symmetric encryption scheme $(\text{KeyGen}, \text{Enc}, \text{Dec})$, and $\star \in \{L, R\}$, define an encryption scheme $E^\star = (\text{KeyGen}^\star, \text{Enc}^\star, \text{Dec}^\star)$ as follows:*

- KeyGen^\star operates as KeyGen^2 from Definition 2. Namely, on input 1^λ it generates $K_L \leftarrow \text{KeyGen}(1^\lambda)$, $K_R \leftarrow \text{KeyGen}(1^\lambda)$, and outputs $K = (K_L, K_R)$.
- Enc^\star , on input a key $K = (K_L, K_R)$ and a message $m = (m_L, m_R)$, computes $c_L \leftarrow \text{Enc}(K_L, m_L)$ and $c_R \leftarrow \text{Enc}(K_R, m_R)$, and outputs $c = (c_L, c_R)$.

- Dec^* , on input a key $K = (K_L, K_R)$ and a ciphertext $c = (c_L, c_R)$, outputs $\text{Dec}(K_*, c_*)$.

The proof of Theorem 3 now follows as a corollary from Claim 2.

Proof of Theorem 3. We instantiate Construction 2 with the double encryption scheme of Definition 2, based on the encryption scheme whose existence follows from the existence of a CRHF; the hash function with a Merkle Hash Tree (MHT) hash based on the CRHF; and instantiate \mathcal{O} with the diO obfuscator. Then the security of the scheme, as well as the length of data and query keys, queries, and processed sets, follow directly from Claim 2. Regarding share keys, the MHT has $\text{poly}(\lambda)$ -length outputs, and \mathcal{O} causes only a polynomial blowup, so by Claim 2, share keys have length $\text{poly}(\lambda)$. Finally, generating the MHT for a set of size s takes time $s \cdot \text{poly}(\lambda)$, and so the runtime of Search is $\text{poly}(\lambda, |S|)$. \square

5.2 MKSE from Public-Coin Differing-Inputs Obfuscation

In this section we show that a slight modification of Construction 2 is secure assuming the underlying obfuscator is a pc -diO obfuscator for TMs. More specifically, we only need to use a signature scheme with some “special” properties. Concretely, we prove the following for a universe \mathcal{U} of size $|\mathcal{U}| \leq \text{poly}(2^\lambda)$:

Theorem 4 (MKSE from pc -diO (short share keys)). *Assume that OWPs, CRHFs, and pc -diO for TMs with polynomial blowup, exist. Then there exists a secure MKSE in which share keys have size $\text{poly}(\lambda)$, where λ is a security parameter. Moreover, data and query keys, as well as queries, have length $\text{poly}(\lambda)$, and given a set S , its processed version has size $|S| \cdot \text{poly}(\lambda)$, and searching in it takes $\text{poly}(\lambda, |S|)$ time.*

The reason we need to change the MKSE scheme outlined in Sect. 5.1 is that it cannot use a pc -diO obfuscator. (Roughly speaking, pc -diO guarantees indistinguishability of the obfuscated programs only as long as it is infeasible for a PPT adversary to find an input on which they differ, even given the randomness used to sample the programs.) Indeed, for every querier j and data owner i , the randomness used to sample the program $P_{j,i}$ (i.e., the program obfuscated in the share key $\Delta_{j,i}$) includes the signing key of querier j . This allows one to sign arbitrary messages, meaning the obfuscated programs in the security game when $b = 0$ and $b = 1$ are *not* differing-inputs. (The program $P_{j,i}$ for querier j and data owner i contains $h(S_i^0)$ when $b = 0$, and $h(S_i^1)$ when $b = 1$, so a differing input would be a query on any keyword contained in one and not the other. The query can be efficiently generated since the encryption and signing keys appear in the randomness used to sample $P_{j,i}$.)

To overcome this issue, we introduce a new signature primitive which we call *dual-mode* signatures. Roughly, a dual-mode signature scheme is an existentially-unforgeable signature scheme associated with an additional `SpecialGen` algorithm that given a list of messages, generates “fake” signatures on these messages, and a “fake” verification key under which they can be verified. These “fake” signatures and key are computationally indistinguishable from honestly generated

signatures and verification key, and the “fake” verification key cannot be used to successfully sign other messages, even given the randomness used to generate the “special mode” verification key and signatures. (This rules out the trivial construction in which `SpecialGen` simply runs the key generation and signing algorithms.) Due to space limitations, we defer the formal definition to the full version [19].

One can think of the `SpecialGen` algorithm as a way of “puncturing” the signing key from the procedure that generates the verification key and signatures. We use this viewpoint to prove security based on pc-diO security and dual-mode signatures: we first replace the actual verification key used in $P_{j,i}$, and the signatures in the queries of j , with ones generated by `SpecialGen`; and then use pc-diO security to replace the obfuscated program from one containing $h(S_i^0)$ to one containing $h(S_i^1)$. (Notice that now the randomness used to sample $P_{j,i}$ does *not* contain the signing key, so one cannot sign queries that j did not issue.)

In the full version [19], we construct dual-mode signatures from OWPs and CRHFs:

Theorem 5. *Assume that OWPs and CRHFs exist. Then there exists a dual-mode signature scheme. Moreover, there exists a polynomial $p(\lambda)$ such that signatures on length- n messages have length $5p(\lambda) \cdot (n + 1)$, signing keys have length $2p(\lambda) + \lambda$, and verification keys have length $2p(\lambda)$.*

We instantiate Construction 2 with a pc-diO obfuscator and the dual-mode signatures of Theorem 5. The properties of the resultant scheme are summarized in the following claim (whose proof is similar to that of Claim 2).

Claim 3 (MKSE (Short Share Keys) from pc-diO). If Construction 2 is instantiated with a secure pc-diO obfuscator for TMs and a secure dual-mode signature scheme, and assuming the security of all building blocks, then Construction 2 is a secure MKSE.

Moreover, if on messages of length n the dual-mode signature scheme outputs signing and verification keys of length $\text{poly}(\lambda)$, and signatures of length $n \cdot \text{poly}(\lambda)$, then the following holds for the MKSE scheme for universe \mathcal{U} . Data and query keys have length $\text{poly}(\lambda)$, queries have length $\log |\mathcal{U}| \cdot \text{poly}(\lambda)$, and for a set S , its corresponding processed set has size $|S| \cdot \text{poly}(\lambda)$. Furthermore, if: (1) evaluating h on length- n inputs takes $H_T(n)$ time, and outputs a hash of length $H_\ell(n)$; and (2) there exist functions $s, T_{\mathcal{O}} : \mathbb{N} \rightarrow \mathbb{N}$ such that for every TM M , $|\mathcal{O}(M)| \leq s(|M|)$, and running $\mathcal{O}(M)$ on inputs of length n takes $T_{\mathcal{O}}(\text{TIME}(M, n))$ time, where $\text{TIME}(M, n)$ is the running time of M on length- n inputs; then: running `Search` on a set S takes $T_{\mathcal{O}}(H_T(|S| \cdot \text{poly}(\lambda)) + |S| \cdot \text{poly}(\lambda) + \text{poly}(\lambda, \log |\mathcal{U}|))$ time, and share keys have size $s(H_\ell(|S| \cdot \text{poly}(\lambda)) \cdot \text{poly}(\lambda))$.

Proof. The correctness and complexity of the scheme is proven similarly to Claim 2. (The only difference is in the length of queries, which contain a signature on a keyword $w \in \mathcal{U}$, and the running time of `Search`, which needs to verify the signature. In both cases, the increase in complexity is caused because signatures on $w \in \mathcal{U}$ have length $\log |\mathcal{U}| \cdot \text{poly}(\lambda)$.)

The security proof proceeds in a sequence of hybrids similar to the proof of Claim 2, but introduces additional complications due to using a weaker obfuscator primitive. More specifically, let \mathcal{A} be a PPT adversary in the security game of Definition 1, and let view_b , $b \in \{0, 1\}$ denote its view in the security game with bit b . We define hybrids $\mathcal{H}_0, \mathcal{H}_1$, and \mathcal{H}_2 as in the proof of Claim 2, and $\text{view}_0 \approx \mathcal{H}_0$ by the same arguments. Indeed, as discussed there, the obfuscated programs in $\text{view}_0, \mathcal{H}_0$ are differing inputs in relation to *every* auxiliary input, and in particular when this auxiliary input is the randomness used by the sampler to sample the programs. $\mathcal{H}_0 \approx \mathcal{H}_2$ because the indistinguishability argument did not use diO security.

Next, we define a new hybrid \mathcal{H}'_2 in which the signatures on the queries \mathcal{W}_j^0 of every querier j , and his verification key vk_j , are generated using the SpecialGen algorithm (instead of the KeyGen and Sign algorithms). We show that $\mathcal{H}'_2 \approx \mathcal{H}_2$ by the indistinguishability of standard and special modes of the dual-mode signature scheme. We condition both hybrids on the values of the sets S_i^0, S_i^1 of data owners, their data keys, the processed sets, the encryption keys of the queriers, the keywords they search for, and their encryptions. (This is possible by an averaging argument, since these values are identically distributed in both hybrids.) Let m denote the number of queriers, then we define a sequence of hybrids $\mathcal{H}^0, \dots, \mathcal{H}^m$, where in \mathcal{H}^j , the signatures and verification key of the first j queriers are generated using SpecialGen, and the signatures and verification keys of all other queriers are honestly generated (using KeyGen and Sign). We prove that $\mathcal{H}^j \approx \mathcal{H}^{j-1}$ for every $1 \leq j \leq m$, and conclude that $\mathcal{H}_2 = \mathcal{H}^0 \approx \mathcal{H}^m = \mathcal{H}'_2$.

Fix some j . Given a distinguisher D between $\mathcal{H}^j, \mathcal{H}^{j-1}$, we construct a distinguisher D' (with the same distinguishing advantage) between the real and special-mode verification key and signatures on the ciphertexts encrypting the keywords in \mathcal{W}_j^0 , and conclude that $\mathcal{H}^j \approx \mathcal{H}^{j-1}$ by the indistinguishability of standard and special modes property. We hard-wire into D' the signing, verification keys, and queries of every $j' \neq j$, as well as all share keys $\Delta_{j',i}$ for $i \in \mathcal{D}$ (this is possible because these values are identically distributed in $\mathcal{H}^j, \mathcal{H}^{j-1}$ and so we can fix them into both hybrids). Given a verification key vk , and a list L of signatures on the ciphertexts of querier j , D' generates for every edge $(j, i) \in E$ the program $P_{K, (K^u, \text{vk}), h(T)}$ (where K, K^u , and $h(T)$ are taken from the hard-wired values), and uses \mathcal{O} to generate the obfuscated program $\Delta_{j,i}$. Then, D' generates the queries of querier j by concatenating the corresponding signature to each ciphertext of j . Together with the hard-wired values, this gives the entire hybrid, and D' runs D on the hybrid, and outputs whatever D outputs. Notice that if vk and the signatures were honestly generated, then the input to D is distributed according to \mathcal{H}^{j-1} , otherwise it is distributed according to \mathcal{H}^j , so D' and D have the same distinguishing advantage.

Next, we define \mathcal{H}_3 as in the proof of Claim 2 (but notice that the verification keys in every $\Delta_{j,i}$ were generated using SpecialGen), and claim that $\mathcal{H}'_2 \approx \mathcal{H}_3$ by the pc -diO security of \mathcal{O} . We define the hybrids $\mathcal{H}^0, \dots, \mathcal{H}^d$ as in the argument

that $\mathcal{H}_2 \approx \mathcal{H}_3$ in the proof of Claim 2 (except that we use \mathcal{H}'_2 instead of \mathcal{H}_2), and show that $\mathcal{H}^l \approx \mathcal{H}^{l-1}$ for every $1 \leq l \leq d$.

Fix some $1 \leq l^* \leq d$, and let (j, i) be the edge for which Δ^{l^*} was generated. We hard-wire the sets $S_{i'}^0, S_{j'}^1, i' \in \mathcal{D}$, and all the keywords that queriers ask about (in both the 0-experiment and the 1-experiment), into $\mathcal{H}^{l^*}, \mathcal{H}^{l^*-1}$ (this is possible because these values are identical in both hybrids). We additionally hard-wire the keys of every querier $j', j' \neq j$ and data owner $i', i' \neq i$, the encrypted sets $T_{i'}, i' \neq i$, the queries of querier j' , and the share keys $\Delta_{j',i'}, \Delta'_{j',i'}$ ($\Delta_{j',i'}$ denotes a key in \mathcal{H}'_2 , $\Delta'_{j',i'}$ denotes a key in \mathcal{H}_3). (This is possible because these values are identically distributed in both hybrids.) We show that $\Delta_{j,i}, \Delta'_{j,i}$ sampled in $\mathcal{H}^{l^*-1}, \mathcal{H}^{l^*}$ (respectively) form a public-coin differing-inputs family of TMs (and conclude that $\mathcal{H}^{l^*-1} \approx \mathcal{H}^{l^*}$ by the pc-diO security of \mathcal{O}).

Let $\mathcal{P}, \mathcal{P}'$ denote the distributions over programs obfuscated in $\Delta_{j,i}, \Delta'_{j,i}$, and let D be a PPT algorithm that obtains $(P, P') \leftarrow \mathcal{P} \times \mathcal{P}'$ and r , where r is the randomness used to sample P, P' . We assume the “worst-case” scenario in which all the values we have fixed into $\mathcal{H}^{l^*}, \mathcal{H}^{l^*-1}$ are known to D . Notice that from the randomness r of the sampler, D learns the encryption keys K_i, K_j^u (the left component of these keys is needed to generate P , whereas the right component is needed to generate P'), as well as the encrypted set T_i , the verification key vk_j^u for signatures of querier j , and the queries of j (which consist of encryptions of keywords, and signatures on these encryptions; the signatures were generated together with the verification key by `SpecialGen`). (We note that from these values D can compute on its own the share keys $\Delta_{j',i}, \Delta'_{j',i}$ for $(j', i) \in E$, and $\Delta_{j,i}, \Delta'_{j,i}$ for $(j, i) \in E$.) We show that D succeeds in finding a differing input only with negligible probability.

Consider first the inputs (for P, P') which D knows (from the hard-wired values, or what it can deduce from r), i.e., the encrypted set T_i (which encrypts the elements of S_i^0 in the left components, and the elements of S_i^1 in the right components; for $i \in \mathcal{D}_c$ this holds since $S_i^0 = S_i^1$), and the queries of querier j . For every such query q there exists a pair (w_L, w_R) of keywords such that q is of the form $q = (c = (c_L, c_R), \sigma)$ where $c_\star \leftarrow \text{Enc}(K_{\star,j}^u, m_\star)$, $\star \in \{L, R\}$, σ is a valid signature on c (generated by `SpecialGen`), and $w_L \in S_i^0 \Leftrightarrow w_R \in S_i^1$. In particular, $P(q, T_i) = P'(q, T_i)$ except with negligible probability since except with negligible probability, the checks in Steps (1)–(2) succeed in both cases (by indistinguishability of the standard and special modes of the signature scheme, σ is indistinguishable from a valid signature on c , which by the correctness of the signature scheme, would pass verification), and Steps (4)–(5) return the same outcome (P searches for w_L in S_i^0 , since it decrypts using Dec^2 , whereas P' searches for w_R in S_i^1 , since it decrypts with Dec^R and the right component of T_i encrypts S_i^1 ; and $w_L \in S_i^0 \Leftrightarrow w_R \in S_i^1$).

Next, we claim that for every other possible input (T', c', σ') that D chooses, $P(T', c', \sigma') = P'(T', c', \sigma') = 0$ except with negligible probability. Indeed, if $(c', \sigma') \neq q$ (where q is some query of querier j) then by the property that special-mode keys cannot sign additional messages, with overwhelming probability σ' is not a valid signature for c' , so the check in Step (1) fails in both P, P' . Moreover,

if $T' \neq T_i$ then by the collision resistance of h , with overwhelming probability $h(T') \neq h(T_i)$, so the check in Step (2) fails. Therefore, P, P' are public-coin differing-inputs.

We now define the last set of hybrids. (These hybrids differ from the corresponding hybrids in the proof of Claim 2 only in that the verification keys and signatures are generated in the special mode.)

\mathcal{H}_4 : In hybrid \mathcal{H}_4 , the queries $(w_{j,1}^1, \dots, w_{j,k_j}^1)$ of every querier $j \in \mathcal{Q}$ are generated using Enc^2 with message $(w_{j,l}^1, w_{j,l}^1), 1 \leq l \leq k_j$, instead of Enc^L with message $(w_{j,l}^0, w_{j,l}^1), 1 \leq l \leq k_j$.

$\mathcal{H}_3 \approx \mathcal{H}_4$ by a similar argument to the one used to prove $\mathcal{H}_1 \approx \mathcal{H}_2$.

\mathcal{H}_5 : In hybrid \mathcal{H}_5 , the encrypted set T_i of every honest data owner $i \notin \mathcal{D}_c$ is generated as the encryption of (S_i^1, S_i^1) with Enc^2 instead of with Enc^L .

$\mathcal{H}_4 \approx \mathcal{H}_5$ by a similar argument to the one used to prove $\mathcal{H}_0 \approx \mathcal{H}_1$.

\mathcal{H}_6 : In hybrid \mathcal{H}_6 , the obfuscated program for every edge $(j, i) \in E$ is generated as follows: (1) the hard-wired keys are $\mathbf{K} = (\mathbf{K}_{L,i}, \mathbf{K}_{R,i}), \mathbf{K} = (\mathbf{K}_{L,j}^u, \mathbf{K}_{R,j}^u)$, instead of $\mathbf{K}' = (\mathbf{0}, \mathbf{K}_{R,i}), \mathbf{K}^{u'} = (\mathbf{0}, \mathbf{K}_{R,j}^u)$; and (2) Dec^2 is used for decryption (instead of Dec^R).

We show that $\mathcal{H}_6 \approx \mathcal{H}_5$ follows from the pc-diO security of \mathcal{O} by a standard hybrid argument in which the obfuscated programs are replaced one at a time. When replacing the program for edge (j, i) from P' (in \mathcal{H}_5) to P (in \mathcal{H}_6), we hard-wire into the PPT \mathbf{D} (which should find a differing input) the sets $S_{i'}^1$ for every $i' \in \mathcal{D}$, the keywords searched for (in the 1-experiment) by all queriers, the encryption keys of all data owners $i', i' \neq i$ and queriers $j', j' \neq j$, the signing, verification keys, and queries of all queriers $j', j' \neq j$, and all encrypted sets $T_{i'}, i' \neq i$. Also, from the randomness r of the sampler (of P, P'), \mathbf{D} learns the encryption keys $\mathbf{K}_i, \mathbf{K}_j^u$, the (special-mode) verification key $\text{vk}_{s,j}^u$, the encryptions of the keywords which querier j searches for, together with the signatures on these ciphertexts, and the encrypted set T_i .

We claim that \mathbf{D} finds a differing input only with negligible probability. The argument is similar to that used to prove $\mathcal{H}_3 \approx \mathcal{H}_2$. The inputs (for P, P') available to \mathbf{D} (from the hard-wired values, and the randomness of the sampler), i.e., the encrypted set T_i , and queries $q = (c, \sigma)$ of querier j , where $c \leftarrow \text{Enc}^2(\mathbf{K}_j^u, m)$ for some m , and σ is a signature for c (generated using SpecialGen) are not differing-inputs (even though P' decrypts the right component of c , whereas P decrypts the left component) because both components of c encrypt m according to Enc (so both P, P' search for m in S_i^1), where Enc is the encryption scheme underlying $\text{Enc}^2, \text{Enc}^L, \text{Enc}^R$. For every other possible input (T', c', σ') that \mathbf{D} chooses, $P(T', c', \sigma') = P'(T', c', \sigma') = 0$ except with negligible probability: if $(c', \sigma') \neq q$ (where q is some query of querier j) then with overwhelming probability σ' is not a valid signature on c' (by the property that special-mode signatures cannot sign additional messages); whereas if $T' \neq T_i$ then with overwhelming probability $h(T') \neq h(T_i)$ (by the collision resistance of h).

In our final new hybrid \mathcal{H}'_6 , the signatures on the queries \mathcal{W}_j^0 of every querier j , and his verification key vk_j , are honestly generated (using the `KeyGen` and `Sign` algorithms). Then $\text{view}_1 = \mathcal{H}'_6 \approx \mathcal{H}_6$ by the same arguments used to show $\mathcal{H}'_2 \approx \mathcal{H}_2$. \square

Theorem 4 now follows as a corollary from Claim 3 and Theorem 5, and since the existence of symmetric encryption follows from the existence of CRHFs.

6 Extensions and Open Problems

In this section we discuss possible extensions of our MKSE definition and constructions, and point out a few open problems in the field.

We have focused on a *selective, indistinguishability-based* MKSE notion (Definition 1). One could also consider several other formulations, as we now discuss.

Simulation-based security. First, one can consider a *selective simulation-based notion*, in which the real-world view of any PPT adversary can be efficiently simulated given only “minimal” information. More specifically, at the onset of the execution the adversary chooses (as in Definition 1) sets of queriers, data owners, and corrupted data owners; a share graph; keyword sets for all data owners; data keys for corrupted data owners; and a (possibly empty) set of distinct keyword queries for each querier. The simulator is then given the sets and data keys of corrupted data owners; the *sizes* of the sets of honest data owners; the share graph; and for each keyword query w of querier j , and every edge (j, i) in the graph, whether $w \in S_i$ or not (where S_i is the set of data owner i). The simulator then generates a complete simulated adversarial view, namely processed sets for all data owners, share keys for all edges in the share graph, and queries for every keyword query. Intuitively, we say that an MKSE is *simulation-secure* if for every PPT adversary there exists a PPT simulator as above, such that the real and simulated views are computationally indistinguishable. The PRF-based MKSE (Construction 1) is simulation-secure (see Remark 1), and we leave it as an open problem to determine whether the MKSE with short share keys (Construction 2, based on diO or pc-diO) is simulation-secure.

A natural question that arises in this context is whether indistinguishability-based security (as in Definition 1) implies simulation-based security (as outlined above). One approach towards tackling this question is to describe an algorithm that, given the input of the simulator (as specified above), generates an assignment for the sets of the honest data owners, and for all keyword queries, in a way that is consistent with the outcome of searching for these keywords. Concretely, this approach reduces the task of constructing a simulator to the following graph problem: given a bipartite graph $G = (L, R, E)$; a set $\{n_v : v \in R\}$ of natural numbers; and for every $u \in L$, a set of coloring of the edges touching u in two colors (blue and red), assign a set $S_v \subseteq \mathcal{U}$ to every $v \in R$ (recall that \mathcal{U} is a universe of possible keywords), and a value $w_u^c \in \mathcal{U}$ to every $u \in L$ and every coloring c of the edges that touch u , such that the following holds:

1. For every $v \in R$, $|S_v| = n_v$.
2. For every $u \in L$, and two colorings c_1, c_2 of the edges touching u , $w_u^{c_1} \neq w_u^{c_2}$.
3. For every $u \in L$, every coloring c of the edges that touch u , and every edge $(u, v) \in E$, $c(u, v) = \text{blue}$ if and only if $w_u^c \in S_v$, where $c(u, v)$ is the color of the edge (u, v) in the coloring c .

(We note that Item 1 guarantees that sets have the “right” size; Item 2 guarantees that the queries made by each querier are distinct; and Item 3 guarantees that the assignments to the sets and keywords searched for are consistent.) At a high level, the main challenge is in finding an assignment for the set that would be consistent over the queries of *multiple* queriers, while simultaneously satisfying the restriction on the size of the set. Intuitively, this issue does not arise in the PRF-based construction since each share key $\Delta_{j,i}$ encodes the entire set, and the Search algorithm does not use the processed set at all (so issues of consistency across *different* queriers do not arise).

Adaptive security. Another possible dimension of generalizing Definition 1 is to consider an (either indistinguishability-based or simulation-based) adaptive definition. At a high level, in this setting the adversary may adaptively generate queriers, data owners (with their sets and data keys), edges in the share graph, and keyword queries, and immediately receives the resultant values. (For example, when an adversary specifies a new data owner and his data key and set, he receives the corresponding processed set; when he adds an edge to the share graph, he receives the corresponding share key, etc.) The security requirement should hold as long as at the end of the execution, the data sets, share graph, and queries satisfy the restrictions imposed by the (selective) security definition (in a simulation-based definition, the only restriction is that the queries of each querier are distinct; in an indistinguishability-based definition there are further restrictions as specified in Definition 1).

Natural approaches towards proving adaptive security, even for the PRF-based construction (Construction 1), seem to run into “selective opening type” issues: in the security proof, we would naturally want to replace the pseudorandom images of the PRF F in share keys with random values, however we do not a-priori know which values the adversary will ask to be “opened” (by making a keyword query for a keyword in the set corresponding to the share key; recall that these queries constitute a key for F). Consequently, we cannot a-priori determine which values should remain pseudorandom (so that they can later be opened). However, we can show that Construction 1 is adaptively simulation-secure *in the Random Oracle model*, namely when all evaluations of F are replaced with calls to the random oracle (replacing $F_K(x)$ with a call to $\text{RO}(K, x)$). The random oracle circumvents such “selective opening type” issues since any (randomly assigned) output of the random oracle can later be “explained” by consistently assigning the random oracle outputs at other (related) points.

Efficiency and security tradeoffs. Finally, an interesting avenue for future research is exploring the tradeoffs between efficiency of the MKSE scheme, and the underlying assumptions. Our MKSE with short share keys (Construction 2)

indicates the hardness of proving an unconditional lower bound on the size of share keys, since it would require ruling out the existence of diO for a *specific* class of samplers. However, it does not rule out the possibility of constructing an MKSE scheme with short share keys based on weaker assumptions (such as the existence of iO for TMs, or ideally, on the existence of OWFs). More generally, one could ask how the search time, and size of share keys, relate to each other; and if there is a lower bound on “search time plus share key size”.

Acknowledgments. We thank the anonymous PKC reviewers for suggesting to use hashing to reduce search time to $O(1)$ in Construction 1. This work was supported by NSF grants CNS-1314722, CNS-1413964, TWC-1664445 and TWC-1646671. The third author was supported in part by The Eric and Wendy Schmidt Postdoctoral Grant for Women in Mathematical and Computing Sciences.

References


1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: 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
2. 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
3. Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. ACM Comput. Surv. **47**(2), 18:1–18:51 (2014)
4. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015, pp. 668–679. ACM Press, October 2015
5. Cash, D., Jaeger, J., Jarecki, S., Jutla, C.S., Krawczyk, H., Rosu, M.-C., Steiner, M.: Dynamic searchable encryption in very-large databases: data structures and implementation. In: NDSS 2014. The Internet Society, February 2014
6. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
7. Chen, X., Li, J., Huang, X., Li, J., Xiang, Y., Wong, D.S.: Secure outsourced attribute-based signatures. IEEE Trans. Parallel Distrib. Syst. **25**(12), 3285–3294 (2014)
8. Cui, B., Liu, Z., Wang, L.: Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. IEEE Trans. Comput. **65**(8), 2374–2385 (2016)
9. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM CCS 2006, pp. 79–88. ACM Press, October/November 2006
10. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. J. Comput. Secur. **19**(3), 367–397 (2011)
11. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with $O(1)$ worst case access time. In: FOCS 1982, pp. 165–169 (1982)

12. Fuller, B., Varia, M., Yerukhimovich, A., Shen, E., Hamlin, A., Gadepally, V., Shay, R., Mitchell, J.D., Cunningham, R.K.: SoK: cryptographically protected database search. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, 22–26 May 2017, San Jose, CA, USA, pp. 172–191 (2017)
13. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 518–535. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_29
14. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003)
15. Goldreich, O.: The Foundations of Cryptography - Basic Techniques, vol. 1. Cambridge University Press, Cambridge (2001)
16. Goldreich, O.: The Foundations of Cryptography - Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
17. Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., Shmatikov, V.: Breaking web applications built on top of encrypted data. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1353–1364. ACM Press, October 2016
18. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. In: 2017 IEEE Symposium on Security and Privacy, pp. 655–672. IEEE Computer Society Press, May 2017
19. Hamlin, A., Shelat, A., Weiss, M., Wichs, D.: Multi-key searchable encryption, revisited. Cryptology ePrint Archive, Report 2018/018 (2018)
20. 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
21. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, pp. 965–976. ACM Press, October 2012
22. Kellaris, G., Kollios, G., Nissim, K., O’Neill, A.: Generic attacks on secure outsourced databases. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1329–1340. ACM Press, October 2016
23. Kiayias, A., Oksuz, O., Russell, A., Tang, Q., Wang, B.: Efficient encrypted keyword search for multi-user data sharing. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 173–195. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45744-4_9
24. Li, J., Chen, X., Li, M., Li, J., Lee, P.P.C., Lou, W.: Secure deduplication with efficient and reliable convergent key management. *IEEE Trans. Parallel Distrib. Syst.* **25**(6), 1615–1625 (2014)
25. Li, J., Li, J., Chen, X., Jia, C., Liu, Z.: Efficient keyword search over encrypted data with fine-grained access control in hybrid cloud. In: Xu, L., Bertino, E., Mu, Y. (eds.) NSS 2012. LNCS, vol. 7645, pp. 490–502. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34601-9_37
26. Liu, Z., Wang, Z., Cheng, X., Jia, C., Yuan, K.: Multi-user searchable encryption with coarser-grained access control in hybrid cloud. In: 2013 4th International Conference on Emerging Intelligent Data and Web Technologies, pp. 249–255 (2013)
27. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990, pp. 427–437 (1990)
28. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015, pp. 644–655. ACM Press, October 2015

29. Popa, R.A., Stark, E., Valdez, S., Helfer, J., Zeldovich, N., Balakrishnan, H.: Building web applications on top of encrypted data using Mylar. In: Proceedings of 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, pp. 157–172 (2014)
30. Popa, R.A., Zeldovich, N.: Multi-key searchable encryption. Cryptology ePrint Archive, Report 2013/508 (2013)
31. Pouliot, D., Wright, C.V.: The shadow nemesis: inference attacks on efficiently deployable, efficiently searchable encryption. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1341–1352. ACM Press, October 2016
32. Van Rompay, C., Molva, R., Önen, M.: A leakage-abuse attack against multi-user searchable encryption. In: PoPETs 2017, no. 3, p. 168 (2017)
33. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society Press, May 2000
34. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: NDSS 2014. The Internet Society, February 2014
35. Tang, Q.: Nothing is for free: security in searching shared and encrypted data. IEEE Trans. Inf. Forensics Secur. **9**(11), 1943–1952 (2014)
36. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of file-injection attacks on searchable encryption. In: 25th USENIX Security Symposium, USENIX Security 2016, pp. 707–720 (2016)
37. Zhao, F., Nishide, T., Sakurai, K.: Multi-user keyword search scheme for secure data sharing with fine-grained access control. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 406–418. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31912-9_27



Fully Homomorphic Encryption from the Finite Field Isomorphism Problem

Yarkin Doröz¹✉, Jeffrey Hoffstein², Jill Pipher², Joseph H. Silverman² , Berk Sunar¹, William Whyte³, and Zhenfei Zhang³

¹ Worcester Polytechnic Institute, Worcester, MA, USA
{ydoroz, sunar}@wpi.edu

² Brown University, Providence, RI, USA
{jhoff, jpipher, jhs}@math.brown.edu

³ OnBoard Security, Wilmington, MA, USA
{wwhyte, zzhang}@onboardsecurity.com

Abstract. If q is a prime and n is a positive integer then any two finite fields of order q^n are isomorphic. Elements of these fields can be thought of as polynomials with coefficients chosen modulo q , and a notion of length can be associated to these polynomials. A non-trivial isomorphism between the fields, in general, does not preserve this length, and a short element in one field will usually have an image in the other field with coefficients appearing to be randomly and uniformly distributed modulo q . This key feature allows us to create a new family of cryptographic constructions based on the difficulty of recovering a secret isomorphism between two finite fields. In this paper we describe a fully homomorphic encryption scheme based on this new hard problem.

Keywords: Finite field isomorphism · Fully homomorphic encryption
Lattice-based cryptography

1 Introduction

Let q be a prime, let \mathbb{F}_q be the finite field with q elements, and let $\mathbf{f}(x) \in \mathbb{F}_q[x]$ and $\mathbf{F}(y) \in \mathbb{F}_q[y]$ be irreducible monic polynomials of degree n . Then

$$\mathbb{X} := \mathbb{F}_q[x]/(\mathbf{f}(x)) \quad \text{and} \quad \mathbb{Y} := \mathbb{F}_q[y]/(\mathbf{F}(y)) \tag{1}$$

are isomorphic fields with q^n elements. Given knowledge of $\mathbf{f}(x)$ and $\mathbf{F}(y)$, it is easy to write down an explicit isomorphism $\mathbb{X} \rightarrow \mathbb{Y}$ and its inverse. We normalize mod q polynomials by choosing their coefficients between $-\frac{1}{2}q$ and $\frac{1}{2}q$, and then we define the size of a polynomial to be the magnitude of its largest

Y. Doröz and B. Sunar—Funding for this research was in part provided by the US National Science Foundation CNS Award #1561536.

J. Hoffstein and J. H. Silverman—Fundings for this research were in part provided by the US National Science Foundation CNS Award #1349908 and #1561709.

coefficient. It is then an observation that, except in trivial cases, the isomorphism $\mathbb{X} \rightarrow \mathbb{Y}$ does not respect the Archimedian property of size. Indeed, when \mathbf{f} and \mathbf{F} are distinct monic irreducible polynomials, we have observed that polynomials within a sphere of small radius (with respect to the L^∞ or L^2 norm) in \mathbb{X} appear to be essentially uniformly distributed in \mathbb{Y} . We record this observation formally, and construct arguments for its veracity in Sect. 2.2.1.

Observation 1. *Let $\mathcal{M}_{n,q}$ be the set of all degree n monic irreducible polynomials mod q and fix $1 \leq \beta < q/2$. Sample $\mathbf{f} \in \mathbb{F}_q[x]$ and $\mathbf{F} \in \mathbb{F}_q[y]$ uniformly from $\mathcal{M}_{n,q}$, and construct \mathbb{X} , \mathbb{Y} and the associated isomorphism $\phi : \mathbb{X} \rightarrow \mathbb{Y}$ as in (1). Let χ_β be a distribution that produces samples with bounded length less than β . Then the image in \mathbb{Y} of a collection of polynomials in \mathbb{X} sampled from χ_β is computationally hard to distinguish from a collection of polynomials sampled uniformly in \mathbb{Y} . By a proper choice of parameters, the ability to distinguish such a collection can be made arbitrarily difficult.*

Remark 1. We will refer to elements of \mathbb{X} or \mathbb{Y} as *short* if they have infinity norm less than β , where generally β will be less than $q/4$.

We will find it essential to choose \mathbf{f} from a subset of $\mathcal{M}_{n,q}$ consisting of monic irreducible polynomials of degree n whose coefficients have absolute value less than or equal to 1. Observation 1 appears to remain true, even when restricted to this subset of $\mathcal{M}_{n,q}$, and the security of our proposed homomorphic scheme will rest on:

Observation 2. *Observation 1 remains true if $\mathbf{f} \in \mathbb{F}_q[x]$ is chosen from the subset of polynomials in $\mathcal{M}_{n,q}$ whose coefficients have a max absolute value 1.*

In this paper we base two distinct, but related, problems on Observation 2.

Definition 1. (FFI). *Finite Field Isomorphism Problems:* Let k be a positive integer. Let $\mathbb{X}, \mathbb{Y}, \phi, \chi_\beta$ be as above. Let $\mathbf{a}_1(x), \dots, \mathbf{a}_k(x), \mathbf{b}_1(x)$ be samples from χ_β , and $\mathbf{A}_i = \phi(\mathbf{a}_i)$ and $\mathbf{B}_1 = \phi(\mathbf{b}_1)$ be the corresponding images. Also sample $\mathbf{B}_2(y)$ uniformly from \mathbb{Y} .

Computational FFI problem: Given $\mathbb{Y}, \mathbf{A}_1(y), \dots, \mathbf{A}_k(y)$, recover $\mathbf{f}(x)$ and/or $\mathbf{a}_1(x), \dots, \mathbf{a}_k(x)$.

Decisional FFI problem: Given $\mathbb{Y}, \mathbf{A}_1(y), \dots, \mathbf{A}_k(y), \mathbf{B}_1$ and \mathbf{B}_2 , with one of $\mathbf{B}_1, \mathbf{B}_2$ an image of a sample from χ_β , identify the image with a probability greater than $1/2$.

Clearly, the decisional FFI problem can be solved if the computational FFI problem can be solved, and if Observation 1 is correct, then the decisional FFI problem can be made arbitrarily hard. We will demonstrate that if a certain lattice reduction problem of dimension roughly $2n$ can be solved, then the decisional FFI problem can be solved, and this lattice reduction problem can be made arbitrarily hard. We do not, however, have a reduction showing that ability to solve the decisional problem implies the ability to solve a lattice reduction problem. In other words, the strongest attacks we have found on the decisional problem

are via lattice reduction arguments, but we cannot rule out the possibility of other, potentially stronger, attacks.

Our plan is to build a somewhat homomorphic encryption scheme based on the decisional FFI problem. This will have double exponential noise growth, but will also have the advantage of being able to handle a reasonable number of multiplications (and additions) of moderate sized integers. We will then analyze the noise performance, and introduce a bit-decomposition-based noise management scheme that allows us to reduce the noise growth to single exponential. This will yield a bootstrappable, thus a fully homomorphic encryption scheme.

We will encode numbers, i.e. messages, as short elements in \mathbb{X} , with noise added for semantic security, and view their corresponding images in \mathbb{Y} as ciphertexts. This will create a symmetric encryption algorithm, which will be somewhat homomorphic in the following sense: Polynomials in elements of \mathbb{X} can be evaluated, and lifted to polynomials over $\mathbb{Z}[x]/(\mathbf{f}(x))$ as long as their coefficients do not exceed $q/2$ in absolute value. Knowledge of these output polynomials will allow the user with knowledge of $\mathbf{f}(x)$ to recover the value of the polynomial over \mathbb{Z} , and the output of the computation. The corresponding ciphertext polynomials in \mathbb{Y} can be evaluated by anyone with knowledge of the public key $\mathbf{F}(y)$, and substantial reduction modulo q will occur. Decryption will occur by mapping isomorphically back to \mathbb{X} , and the correct result will be output as long as the coefficients do not exceed $q/2$ in absolute value.

This is where an important point arises. In 1996, (eventually published in [25]), NTRU introduced the idea that if two short polynomials in $\mathbb{Z}[x]$ are multiplied, and the result is reduced modulo $x^n - 1$, then the reduced product is also (moderately) short. This observation has been used, in the years since then, in a variety of cryptographic constructions. In this paper we make use of a variation on this observation: This property remains true for a considerably larger class of polynomials than $x^n \pm 1$. In particular, if $\mathbf{f}(x)$ is chosen to be monic, of degree n , and have coefficients from the set $\{-1, 0, 1\}$, then a short polynomial times a short polynomial remains moderately short when reduced modulo $\mathbf{f}(x)$. If parameters are chosen properly, the search space for $\mathbf{f}(x)$ can be made arbitrarily large, making it impractical to locate $\mathbf{f}(x)$ by a brute force search.

The symmetric system sketched above can be converted into a public key encryption scheme using the standard technique of publishing a list of encryptions of 0 and adding short linear combinations of these encryptions as noise. Its semantic security can be seen to be based on the decisional FFI problem, not on the presumably harder computational FFI problem. It is not immediately obvious that this is the case, as all ciphertexts of messages will be images of short vectors in \mathbb{X} , but in the simple instantiation we will present here, it can be shown that this is true. (See Theorem 1 in Sect. 3.2.4.)

1.1 Subfield Attack

Despite major advances over the past few years the biggest challenge preventing the deployment of FHE schemes in real life applications is efficiency. To address the efficiency bottleneck, many optimizations were proposed including

some that take advantage of specialization of the underlying field/ring structure. Such specializations enable efficient batched parallel evaluations, make it possible to choose parameters that support highly efficient number theoretical transforms, and in some cases even reduce the size of evaluation keys.

However, such customizations may potentially introduce weaknesses in the security assumptions of the schemes. A recent family of attacks proposed by Albrecht et al. [29], by Cheon et al. [8], and by Kirchner and Fouque [27] exploit the special structure, namely subfields, in ring based FHE schemes. Furthermore, the attack in [27] also works when the underlying ring does not admit subfields. Moving to a subfield with a Norm mapping as in [29], or a Trace mapping as in [8] or the Gentry-Szydło mapping [22] as in [27] will reduce the dimension of the lattice. Then, via a projection, also named zero-forcing in the original May-Silverman description [30], the Kirchner-Fouque method is able to create a lattice with an even smaller dimension, at the cost of reducing the number of unique shortest vectors in the lattice.

This set of attacks demonstrated that several NTRU based FHEs with medium size parameters are no longer secure. Specifically, if the NTRU scheme is constructed with the DSPR security assumption, which is the case in some of the NTRU based FHE schemes [3, 28], the assumed security level of the scheme can be significantly reduced. While the authors suggest more caution on parameter selection by avoiding specialized fields in this particular case, there could be further attacks that exploit specialized parameters. It has become quite clear that we need more generic constructions that avoid specialized structures as much as possible. Furthermore, we need diversity in the FHE constructions, i.e. FHEs that remain secure even if other conjectured hard problems, e.g. DSPR or Approximate GCD, are shown to be weaker than expected.

These are among the goals of the FHE scheme proposed in this paper: The proposed construction is based on the DFFI problem; a new problem we propose and analyze here for the first time. The proposed construction avoids specializations. The FHE scheme is based on a fixed prime q and a class of short generic private keys $\mathbf{f}(x)$ with the property that $\mathbf{f}(x)$ is monic, irreducible mod q , and the Galois group of the associated finite field $\mathbb{Z}_q[x]/(\mathbf{f}(x))$ is C_n .

With such choice of parameters it is safe to claim that attacks in [8, 29] no longer apply due to the lack of subfields. In addition, as one shall see in Sect. 2.4, the unique shortest vectors in this class of lattices are not sparse vectors with many 0s, and they are not cyclic rotations of each other. Therefore, the projection method will not work either. Thus we also assert that attack in [27] is not applicable either.

Remark 2. The security of the finite field homomorphic encryption scheme presented here is based on the decisional problem (DFFI). It may be possible to construct a homomorphic encryption scheme that solely depends on the computational problem, (CFFI), but in the interest of simplicity we will not pursue this here. It is certainly possible to construct a signature scheme, based on the CFFI, and this will appear elsewhere.

1.2 A Sketch of the Main Ideas

Messages, which are integers, will be mapped to elements of \mathbb{X} by some method. These elements will be sparse, low weight polynomials, $m(x)$, of degree at most $n - 1$. For each message encryption, a sparse low weight, e.g. trinary, polynomial $r(x)$ of degree at most $n - 1$ will be chosen at random. A polynomial $p(x)$ will be fixed as a public parameter. This polynomial will have coefficients with small infinity norm. Two useful possibilities for $p(x)$ are $p(x) = 2$, and $p(x) = x - 2$. We will illustrate below with the example $p(x) = x - 2$. To encode an integer $1 \leq m < 2^n$, write m in base two as $m = b_0 + 2b_1 + \dots + 2^{n-1}b_{n-1}$, and represent m by $m(x) = b_0 + b_1x + \dots + b_{n-1}x^{n-1}$. Thus $m(2) = m$. An encoding of $m(x)$ in \mathbb{X} will be done as follows:

- Choose $r(x)$ at random from a given distribution of sparse, binary or trinary, polynomials of degree less than n .
- The encoded message is $e_m(x) := m(x) + p(x)r(x) \pmod{f(x)}$. As the coefficients of $p(x)$ and $r(x)$ are very small, and $f(x)$ is chosen as described above, the reduction of $m(x) + p(x)r(x) \pmod{f(x)}$ will have coefficients that remain small relative to q . In other words, the lift of $e_m(x)$ from \mathbb{X} to an element of $\mathbb{Z}[x]/(f(x))$ with coefficients in the interval $(-q/2, q/2]$ will have no reduction modulo q occurring.

Encryption of $e_m(x)$ is done by mapping $e_m(x)$ to its isomorphic image $E_m(y)$ in \mathbb{Y} , using the isomorphism $\mathbb{X} \rightarrow \mathbb{Y}$ that is known to the encryptor. The somewhat homomorphic property for multiplication is seen as follows: Given $e_{m_1}(x) = m_1(x) + p(x)r_1(x)$ and $e_{m_2}(x) = m_2(x) + p(x)r_2(x)$, the product is given by

$$\begin{aligned} e_{m_1}(x)e_{m_2}(x) &= m_1(x)m_2(x) + p(x)r_1(x)m_2(x) + p(x)r_2(x)m_1(x) + p(x)^2r_1(x)r_2(x) \\ &= m_1(x)m_2(x) + p(x)[r_1(x)m_2(x) + r_2(x)m_1(x) + p(x)r_1(x)r_2(x)] \pmod{(f(x), q)}. \end{aligned} \tag{2}$$

The key observation is that since the coefficients of $e_{m_1}(x)$ and $e_{m_2}(x)$ are small compared to q , the product, even after reduction mod $f(x)$, will still have coefficients that are small compared to q . As a result, if the reduced product $e_{m_1}(x)e_{m_2}(x)$ is lifted from \mathbb{X} to $\mathbb{Z}[x]/(f(x))$ with coefficients chosen from the interval $(-q/2, q/2]$, then the coefficients will be the same as if the computation had taken place over $\mathbb{Z}[x]/(f(x))$.

A similar comment applies to $e_{m_1}(x) + e_{m_2}(x)$. Because the mapping between \mathbb{X} and \mathbb{Y} is a field isomorphism, it follows that

$$E_{m_1}(y)E_{m_2}(y) = E_{m_1m_2}(y) \text{ and } E_{m_1}(y) + E_{m_2}(y) = E_{m_1+m_2}(y).$$

This means that a polynomial function of elements of \mathbb{X} can be computed on the isomorphic images of these elements in \mathbb{Y} and the output mapped back to \mathbb{X} , and, as long as the coefficients in the corresponding \mathbb{X} computation remain in the interval $(-q/2, q/2]$, the image of the output in \mathbb{X} can be lifted to $\mathbb{Z}[x]/(f(x))$ without any loss of information.

The key question then is how to recover $m(x)$ from a polynomial of the form $m(x) + p(x)r(x)$ in \mathbb{X} . After a computation is performed, as seen in (2) above, the output in \mathbb{X} will still have this form, although the coefficients of $m(x)$ and $r(x)$ may be considerably larger than binary or trinary. As long as they have not passed $q/2$ in absolute value, the lift to $\mathbb{Z}[x]/(f(x))$ will not involve any mod q reduction. The decryption process, then consists of:

- Map the output of the computation in \mathbb{Y} back to \mathbb{X} . It will have the form $m'(x) + p(x)r'(x)$, for unknown polynomials $m'(x)$ and $r'(x)$
- This can be further lifted to $\mathbb{Z}[x]$ by viewing of it as $m'(x) + p(x)r'(x) + s(x)f(x)$ for some also unknown polynomial $s(x)$
- Compute the resultant of $f(x)$ and $p(x)$. This is the ideal in $\mathbb{Z}[x]$ generated by $p(x)$ and $f(x)$ which, in the case $p(x) = x - 2$, is simply $f(2)$. Also, $m'(x) + p(x)r'(x) + s(x)f(x)$ reduced mod $f(x)$ and $x - 2$ is $m(2) \bmod f(2)$. Thus, as long as m is less than $f(2)$, $m = m(2)$ will be recovered exactly.

The process breaks down when the size of any coefficient of the computation exceeds $q/2$ in absolute value. Note that the collection of all $p(x)r(x)$ in \mathbb{X} is all possible encodings of 0, and their images in \mathbb{Y} are all possible encryptions of 0. As we are in a field, not a ring, the ideal generated by all such $p(x)r(x)$ is, of course, all of \mathbb{Y} .

1.3 Related Work

The first Fully Homomorphic Encryption (FHE) scheme was constructed by Gentry [17, 19] in 2009, answering a problem that had remained open for over three decades. Gentry’s scheme is based on ideal lattices and the security assumptions are based on hard problems in lattices. A key innovation in Gentry’s construction is *bootstrapping*, which allows a party to refresh the noise level in a ciphertext without having access to a secret key. Despite its success, bootstrapping has remained the bottleneck in FHE implementations. After Gentry’s original scheme, many other constructions based on a variety of hardness assumptions followed that aimed to improve the efficiency of FHE.

One such construction based on the learning-with-errors (LWE) problem was proposed by Brakerski and Vaikuntanathan [6]. The security of the scheme is based on the hardness of short vector problems. The LWE-based construction was later improved by Brakerski, Gentry and Vaikuntanathan (BGV) in [5] using a *modulus switching* technique that slows the noise accumulation drastically. Modulus switching is applied at each multiplicative level, which prevents exponential noise growth. Thereby the noise remains fixed throughout the homomorphic evaluation levels. Later, a new noise management technique was introduced by Brakerski [4], applicable to LWE schemes, that decreases noise growth from quadratic to linear using tensor products. Gentry et al. [20] demonstrated that it is possible to perform deep homomorphic evaluations by providing the first AES evaluation implemented using the BGV scheme embodied in a software library called HELIB [23]. The authors optimize the design using the SIMD technique introduced in [31] to batch multiple messages and process parallel AES

operations. Another FHE construction based on the assumed hardness of the *Integer Approximate-GCD* problem was proposed by van Dijk et al. [12]. This work was followed by Coron et al. [10], where the public key size was reduced from $\lambda\mathcal{O}(\kappa^{10})$ to $\mathcal{O}(\kappa^7)$ where κ is the security parameter. In [11] the public key size was further reduced from $\mathcal{O}(\kappa^7)$ to $\mathcal{O}(\kappa^5)$ and modulus switching methods were adapted to the integer scheme. Another follow up work by Coron et al. [9] implements a variant of van Dijk et al.'s scheme using the scale invariant property introduced earlier by Brakerski [4].

Another leveled FHE scheme was presented by López-Alt, Tromer, Vaikuntanathan (LTV) in [28]. It is based on a variant of NTRU [25] constructed earlier by Stehlé and Steinfeld [32]. The scheme is a multi-party scheme that is capable of processing homomorphic functions for various users each with their individual keys. The authors use the *relinearization* technique introduced in [6] and also adapt modulus switching to mitigate the noise growth, thus keeping the growth linear in size over the levels. To compute relinearization, the scheme requires evaluation keys, which increases the memory requirement and becomes prohibitive especially in *deep* evaluations. The NTRU variant by Stehlé and Steinfeld [32] was later modified and implemented by Bos et al. in [3]. Their scheme, named YASHE, adopts the tensor product technique in [4] and achieves a scale-invariant scheme with limited noise growth on homomorphic operations. Also, with the use of the tensor product technique, the authors managed to improve the security of the LTV scheme [28] by using much higher levels of noise and thereby removed the Decisional Small Polynomial Ratio (DSPR) assumption. Instead, the scheme relies only on standard lattice reductions as in [32]. However, as the authors also note, the YASHE scheme requires a large evaluation key and a complicated key switching procedure. In [3] the authors introduce a modification (YASHE') to their scheme to eliminate the problems of expensive tensor product calculations and large evaluation keys. However, this modification re-introduces the DSPR assumption. Another modified LTV-FHE implementation, along with AES evaluation, was presented by Doröz et al. in [13]. The security of their scheme depends on the DSPR and R-LWE assumptions as in [28]. Their implementation uses the relinearization and modulus switching methods as in [28] to cope with noise, and it introduced a specialized ring structure to significantly reduce the evaluation key size. Since both the YASHE' and LTV-FHE schemes rely on the DSPR problem, both are vulnerable to the Subfield Attack [29].

Motivated by the large evaluation key requirements come by complex noise management techniques such as *relinearization*, *modulus switching*, and *bootstrapping* employed by earlier FHE schemes Gentry et al. [21] proposed a new scheme based on the approximate eigenvector problem. The system uses matrix additions and multiplications, which makes it asymptotically faster. At first, they constructed the GSW scheme as a *somewhat homomorphic scheme*, since for a depth L circuit with B -bounded parameters, the noise grows with a double exponential B^{2^L} . To convert the scheme into a leveled FHE, they introduced a *Flattening* operation that decomposes the ciphertext entries into bits. The secret key is also kept in a special powers-of-two form. With these modifica-

tions, the noise performance is improved significantly. For a depth L circuit with B -bounded secret key entries and 1-bounded (flattened) ciphertexts, the error magnitude is at most $(N + 1)^L B$ for $N = \log(q)(n + 1)$. However, ciphertexts still require a considerable amount space, roughly $\Theta(n^2 \log(q)^2)$, and as noted by GSW [21], in practice their scheme may not be as efficient as existing leveled schemes. More recently, the Flattening technique was adapted by Doröz and Sunar to NTRU in a new FHE scheme called F-NTRU [14]. Similar to the GSW scheme, F-NTRU does not require evaluation keys or key switching. More significantly, the scheme eliminates the DSPR assumption and relies only on the standard R-LWE assumption which makes it the only NTRU variant FHE scheme immune to the Subfield Attack.

1.4 Paper Organization

In Sect. 2 we formally introduce the finite field isomorphisms problem, state hardness assumptions, and study lattice and non-lattice techniques to establish the difficulty of the problem against known techniques. We then show how to construct a fully homomorphic public-key encryption scheme in Sect. 3 by first building a somewhat homomorphic encryption scheme and then by converting it into a bootstrappable scheme via a new bit decomposition based noise management scheme. In Sect. 4, we conclude our paper.

In the appendices, we discuss how to construct field representations \mathbb{X} and \mathbb{Y} and the necessary isomorphisms $\mathbb{X} \rightarrow \mathbb{Y}$ and $\mathbb{Y} \rightarrow \mathbb{X}$ (Sect. A), we give a more detailed noise analysis (Sect. B), we perform security analysis and give estimates on the parameters (Sect. C), and we give test results for our observation 2 (Sect. D).

2 The Finite Field Isomorphism (FFI) Problem

2.1 Preliminaries

We begin by formally introducing some notation that has already been used in the previous section. Additional notation will be introduced at the start of Sect. 3. For given degree n monic irreducible polynomials $\mathbf{f}(x) \in \mathbb{F}_q[x]$ and $\mathbf{F}(y) \in \mathbb{F}_q[y]$, we create two copies of \mathbb{F}_q^n , which we denote by $\mathbb{X} := \mathbb{F}_q[x]/(\mathbf{f}(x))$ and $\mathbb{Y} := \mathbb{F}_q[y]/(\mathbf{F}(y))$. In general, polynomials denoted by lower case letters will be polynomials in \mathbb{X} , and their isomorphic images in \mathbb{Y} will be denoted with the corresponding capital letters. The vector form of a polynomial is simply the vector consisting of its coefficients. We often identify polynomials and vectors when there is no ambiguity. Consider a polynomial $\mathbf{a}(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} \in \mathbb{X}$. We will informally say that $\mathbf{a}(x)$ is *short* if for all i , the congruence class $a_i \bmod q$ reduced into the interval $(-q/2, q/2]$ is small relative to q . An important class of such polynomials are those satisfying $a_i \in \{-1, 0, 1\}$; these are called *ternary polynomials*. We denote by $\|\mathbf{a}\| = \|\mathbf{a}\|_\infty := \max |a_i|$ and $\|\mathbf{a}\|_2 := (a_0^2 + \dots + a_{n-1}^2)^{1/2}$ the L^∞ and L^2

norms of \mathbf{a} , respectively, where it is understood that the coefficients of \mathbf{a} are always normalized to lie in the interval $(-q/2, q/2]$. Denote by $\mathcal{M}_{n,q}$ the set of all degree n monic irreducible polynomials mod q . When there is no ambiguity, we will suppress the subscripts.

2.2 Discussions and Proofs

2.2.1 Arguments for the Truth of Observation 1

Lemma 1. *For large n , for any fixed $\mathbf{f}(x) \in \mathbb{F}_q[x]$, and any given degree $n - 1$ polynomial $\phi(y) \in \mathbb{F}_q[y]$, there will exist, with probability approaching 1, a unique monic irreducible $\mathbf{F}(y) \in \mathbb{F}_q[y]$ such that the map $x \rightarrow \phi(y)$ induces an isomorphism between $\mathbb{F}_q[x]/(\mathbf{f}(x))$ and $\mathbb{F}_q[y]/(\mathbf{F}(y))$.*

Proof. As $\mathbb{F}_{q^n}/\mathbb{F}_q$ is Galois, any irreducible polynomial with one root must split completely, implying that $\mathbf{f}(x)$ has n distinct roots in $\mathbb{F}_q[y]/(\mathbf{F}(y))$, and similarly, that no two monic irreducible polynomials of degree n in $\mathbb{F}_q[x]$ can share a root. Fix a degree n monic irreducible polynomial $\mathbf{f}(x) \in \mathbb{F}_q[x]$. By the prime number theorem for function fields, for fixed q and large n , $|\mathcal{M}_{n,q}|$, i.e., the number of distinct irreducible monic polynomials over $\mathbb{F}_q[x]$, is asymptotic to q^n/n ; see [26, Chap. 7, Sect. 2, Corollary 2]. It follows that for any polynomial $\mathbf{f} \in \mathcal{M}_{n,q}$ there are asymptotically q^n/n distinct isomorphic images of $\mathbb{F}_q[x]/(\mathbf{f}(x))$ and hence also q^n/n potential \mathbf{F} . Choose at random a degree $n - 1$ polynomial $\phi(y) \in \mathbb{F}_q[y]$. There are exactly $(q - 1)q^{n-1}$ such polynomials. There are also, asymptotically, a total of $n \times q^n/n = q^n$ isomorphisms between $\mathbb{F}_q[x]/(\mathbf{f}(x))$ and all possible $\mathbb{F}_q[y]/(\mathbf{F}(y))$, where $\mathbf{F}(y)$ varies over all distinct monic irreducible polynomials. These are given by sending x to each of the n distinct roots of each $\mathbf{F}(y)$. With probability approaching 1 (for large q), these sets have the same order, and as one is contained in the other, they are asymptotically equal. \square

This provides evidence for the truth of Observations 1 for the following reason. Suppose one chooses, independently, a private monic irreducible $\mathbf{f}(x)$, and a $\phi(y)$, with the coefficients of $\phi(y)$ chosen randomly and uniformly from \mathbb{F}_q . Then with high probability there will be a corresponding (monic, irreducible) $\mathbf{F}_1(y)$ and a short polynomial $\mathbf{a}(x)$ will be mapped to $\mathbf{A}(y) = \mathbf{a}(\phi(y))$ reduced modulo $\mathbf{F}_1(y)$. As the coefficients of $\phi(y)$ are random and uniformly distributed modulo q it is reasonable to assume that the coefficients of $\mathbf{A}(y)$ will be similarly uniformly distributed modulo q . Unfortunately, because of the highly non-linear aspect of this mapping, it appears to be hard to construct a proof of this. The polynomial $\mathbf{F}_1(y)$ can be used as the public key. However, it may be convenient to use a polynomial of a simpler form, such as $\mathbf{F}_2(y) = y^n - y - 1$ to make computations easier for the public party. In this case the composite isomorphism

$$\mathbb{F}_q[x]/(\mathbf{f}(x)) \rightarrow \mathbb{F}_q[y]/(\mathbf{F}_1(y)) \rightarrow \mathbb{F}_q[y]/(\mathbf{F}_2(y))$$

can be used for encryption. It is again reasonable to assume, though hard to prove, that the composite mapping continues to cause coefficients of images of short polynomials to be uniformly distributed modulo q .

Remark 3. Because of Observation 2, that non-trivial isomorphisms send short polynomials in \mathbb{X} to uniformly distributed elements of \mathbb{Y} , we believe that there are no easy cases of CFFI. Hence, similar to hard lattice problems such as those described in [1], we suspect that there may well be an average-case/worst-case equivalence for the computational finite field isomorphism problem. However, research in this direction is beyond the scope of the present paper and clearly requires considerable further study.

2.2.2 Arguments for the Truth of Observation 2

In order to build a multiplicative homomorphic encryption scheme we require that products of short elements in \mathbb{X} are also short. Hence, we cannot simply sample $\mathbf{f}(x)$ uniformly from $\mathcal{M}_{n,q}$. Instead, we will sample $\mathbf{f}(x)$ uniformly from $\mathcal{M}_{n,q}$ with the requirement that $\|\mathbf{f}(x)\|$ is bounded.

In order to estimate the size of the search space for $\mathbf{f}(x)$, we will rely on the following very reasonable assumption:

Assumption 1. *Monic irreducible polynomials are uniformly distributed over $\mathbb{F}_q[x]$.*

This assumption implies that Observation 2 is true. It also implies (together with the argument that $|\mathcal{M}_{n,q}|$ is on the order of q^n/n) that for $1 \leq \beta \leq \frac{1}{2}q$ there are approximately $(2\beta)^n/n$ distinct irreducible monic polynomials $\mathbf{a}(x)$ over $\mathbb{F}_q[x]$ satisfying $\|\mathbf{a}(x)\| \leq \beta$. This quantifies the size of the set of all possible \mathbf{f} and enables us to verify that with well chosen parameters it is large enough to be robust against a brute force search.

This shortness of $\mathbf{f}(x)$ is exploited via the following useful property:

Property 1. If $\mathbf{f}(x)$ is short, and if $\mathbf{a}(x)$ and $\mathbf{b}(x)$ are short elements of \mathbb{X} , then the product $\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)$ is also a reasonably short element of \mathbb{X} .

As remarked earlier, Property 1 has been widely exploited in ideal and lattice-based cryptography, especially with $\mathbf{f}(x) = x^n \pm 1$, starting with the original NTRUEncrypt [25].

2.3 An Algorithm to Find an Isomorphism

We explain how to find suitable polynomials $\mathbf{f}(x)$ and $\mathbf{F}(y)$ and an explicit isomorphism $\mathbb{F}_q[x]/(\mathbf{f}(x)) \mapsto \mathbb{F}_q[y]/(\mathbf{F}(y))$. We need to find four polynomials $(\mathbf{f}, \mathbf{F}, \phi, \psi)$ satisfying:

- $\mathbf{f}(x) \in \mathbb{F}_q[x]$ is irreducible monic of degree n with $\|\mathbf{f}(x)\| \leq \beta$.
- $\mathbf{F}(y) \in \mathbb{F}_q[y]$ is irreducible monic of degree n with random coefficients.
- $\phi(y) \in \mathbb{F}_q[y]$ and $\psi(x) \in \mathbb{F}_q[x]$ have degree less than n .
- $\mathbf{F}(y) \mid \mathbf{f}(\phi(y))$.
- $\phi(\psi(x)) \equiv x \pmod{\mathbf{f}(x)}$.

The algorithm for finding such an isomorphism is shown in Algorithm 1.

Algorithm 1. Finite Field Isomorphism Generation

-
- 1: Sample $\mathbf{f}(x)$ and $\mathbf{F}(y)$ as required.
 - 2: Find a root of $\mathbf{f}(x)$ in the finite field $\mathbb{F}_q[y]/(\mathbf{F}(y)) \cong \mathbb{F}_{q^n}$ and lift this root to a polynomial $\phi(y) \in \mathbb{F}_q[y]$ of degree less than n .
 - 3: Construct the unique polynomial $\psi(x) \in \mathbb{F}_q[x]$ of degree less than n satisfying $\psi(\phi(y)) \equiv y \pmod{\mathbf{F}(y)}$.
 - 4: **return** $\mathbf{f}(x)$, $\mathbf{F}(y)$, $\phi(y)$ and $\psi(x)$.
-

Remark 4. We note again that the secret polynomial $\mathbf{f}(x)$ and the public polynomial $\mathbf{F}(y)$ are chosen *independently*, so in particular, knowledge of $\mathbf{F}(y)$ reveals no information about $\mathbf{f}(x)$. In other words, any polynomial satisfying the norm bound is a potential candidate for $\mathbf{f}(x)$. The attacker only begins to acquire information about $\mathbf{f}(x)$ when she is given isomorphic images in \mathbb{Y} of (short) polynomials in \mathbb{X} . Further, the fact that there are no security issues in the choice of $\mathbf{F}(y)$, other than the requirement that it be irreducible in $\mathbb{F}_q[y]$, means that $\mathbf{F}(y)$ may be chosen to simplify field operations in the quotient field $\mathbb{F}_q[y]/(\mathbf{F}(y))$. For example, one could take $\mathbf{F}(y)$ to be a trinomial. The point is that the attacker can always replace your $\mathbf{F}(y)$ with her choice of $\mathbf{F}'(y)$, since she can easily construct an isomorphism from $\mathbb{F}_q[y]/(\mathbf{F}(y))$ to $\mathbb{F}_q[y]/(\mathbf{F}'(y))$.

We now discuss the steps in the generation algorithm in more details. In Step 2, we are required to find a root of a polynomial $\mathbf{f}(x)$ in a finite field \mathbb{F}_{q^n} that is given explicitly as a quotient $\mathbb{F}_q[y]/(\mathbf{F}(y))$. There are fast polynomial-time algorithms for doing this.¹ We note that in our set-up, the polynomial $\mathbf{f}(x)$ is irreducible of degree n , so any one of its roots generates the field \mathbb{F}_{q^n} , and since any two fields with q^n elements are isomorphic, it follows that $\mathbf{f}(x)$ must have a root in $\mathbb{F}_q[y]/(\mathbf{F}(y))$. Further, since $\mathbb{F}_{q^n}/\mathbb{F}_q$ is Galois, any irreducible polynomial with one root must split completely, so in fact $\mathbf{f}(x)$ has n distinct roots in $\mathbb{F}_q[y]/(\mathbf{F}(y))$. We may take $\phi(y) \pmod{\mathbf{F}(y)}$ to be any one of these roots.

In Step 3, we need to construct $\psi(x)$. We describe three ways to do this. All are efficient. Method 2 is always faster than method 1. It is not clear which is the more efficient between methods 2 and 3.

1. One can compute the roots of $\mathbf{F}(y)$ in $\mathbb{F}_q[x]/(\mathbf{f}(x))$. As above, there will be n distinct roots, and one of them will be the desired $\psi(x)$.
2. One can compute a root of $\phi(y) - x$ in the field $\mathbb{F}_q[x]/(\mathbf{f}(x))$.
3. One can use linear algebra as described in Appendix A.

2.4 Known Approaches to Recovering the Secret Isomorphism

In this section, we explore two possible methods to solve the finite field isomorphism problem. Such an isomorphism will be described as an n -by- n matrix M . The first approach is based on lattice reduction. The second approach is a highly non-linear attack of unknown but, we believe, high difficulty.

¹ For example, Pari-GP [33] provides the routine `polrootsff`.

2.4.1 Lattice Attack of ($\dim \approx 2n$)

In this subsection we describe a lattice attack that uses a transcript of ciphertexts. We formulate this abstractly by saying that there is an unknown n -by- n matrix M with mod q coefficients, and there are known vectors $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k$ with the property that the unknown vectors $M\mathbf{A}_i \bmod q$ are small for all $i = 1, 2, \dots, k$.

For the computational isomorphism problem we would need to recover the rows of M exactly, and place them in the correct order. However, to solve the decisional problem it would suffice to search for a single row of M . The dimension of an attack lattice can be further reduced. To accomplish this, let \mathbf{m} be some (unknown) row of M , say the j^{th} row, and let $b_i = \mathbf{m} \cdot \mathbf{A}_i$ for $i = 1, 2, \dots, k$, be the corresponding (unknown) small values of the indicated dot products. Then

$$A = (\mathbf{A}_1 \mid \mathbf{A}_2 \mid \dots \mid \mathbf{A}_k), a = (\mathbf{a}_1 \mid \mathbf{a}_2 \mid \dots \mid \mathbf{a}_k), \mathbf{b}_j = (b_1, b_2, \dots, b_k),$$

and we set $D = \begin{pmatrix} A \\ qI \end{pmatrix}$. Thus A and a are two n -by- k matrices, and D is an $(n+k)$ -by- k matrix. The vector \mathbf{b}_j is a k dimensional “slice” consisting of the j^{th} coordinates of the \mathbf{a}_i , which are the inverse images in \mathbb{X} of the \mathbf{A}_i . Let $\mathcal{L}(D)$ denote the row span of D , so $\dim \mathcal{L}(D) = k$. Then $\mathcal{L}(D)$ contains the short row vector of \mathbf{b}_j . If we choose k sufficiently large, then the vectors \mathbf{b}_j will stand out as unusually short, relative to the Gaussian heuristic, and a successful lattice reduction argument would recover them, or short linear combinations of them. This means that an attacker with sufficient lattice reduction resources could solve the decisional FFI problem, in the following way. Suppose the attacker is provided with a list of \mathbf{A}_i , images in \mathbb{Y} of short vectors in \mathbb{X} , and a vector \mathbf{B} , which might or might not be the image in \mathbb{Y} of a short vector in \mathbb{X} . Considering

$$(\mathbf{A}_1 \mid \mathbf{A}_2 \mid \dots \mid \mathbf{A}_k \mid \mathbf{B}),$$

a successful lattice reduction could produce a slice through the j^{th} coordinates. If each $\mathbf{A}_i = (a_{i,1}, a_{i,2}, \dots, a_{i,n})^T$ then $(a_{1,j}, a_{2,j}, \dots, a_{k,j}, b_j)$ will be in $\mathcal{L}(D)$. If \mathbf{B} is the image of a short vector in \mathbb{X} then $(a_{1,j}, a_{2,j}, \dots, a_{k,j}, b_j)$ will have all short entries, say, around β in absolute value, and a successful lattice reduction argument should recover it. If \mathbf{B} is not the image of a short vector in \mathbb{X} then $(a_{1,j}, a_{2,j}, \dots, a_{k,j}, b_j)$ will have k short entries and one entry that is random mod q . If the vector, with this new final entry were recovered by lattice reduction, it is highly unlikely that the random length of the final entry would be on the order of β , and, as q will be considerably larger than k , it is also highly unlikely that this output would be shorter than the gaussian heuristic expected vector. This would enable the decision problem to be solved with greater than 50% probability. The technical estimates are given in the remainder of this section.

Since $\|\mathbf{a}\| \leq \beta$, the length of the target vector is roughly $\|\mathbf{a}\|_2 \asymp \beta\sqrt{k}$. The determinant of $\mathcal{L}(D)$ is the gcd of the k -by- k minors of the matrix D . Each such minor includes at least $k-n$ rows from the bottom part of the matrix, which gives a factor of q^{k-n} to each k -by- k minor. Since the entries of A are more-or-less random, it is likely that $\det \mathcal{L}(D)$ is some small multiple of q^{k-n} . Hence the Gaussian expected shortest vector in $\mathcal{L}(D)$ has length roughly

$$\gamma(\mathcal{L}(D)) \asymp \sqrt{\frac{\dim \mathcal{L}(D)}{2\pi e}} (\text{Det } \mathcal{L}(D))^{1/\dim \mathcal{L}(D)} = \sqrt{\frac{k}{2\pi e}} \cdot (q^{k-n})^{1/k}.$$

To analyze the hardness of recovering this vector via lattice reductions, we focus on the k -th root of the ratio between the Gaussian expected length and the unique shortest vectors:

$$\left(\frac{q^{\frac{k-n}{k}}}{\beta\sqrt{2\pi e}} \right)^{\frac{1}{k}}.$$

This attack appears to be optimal when $k \approx 2n$. In the meantime, analyses in [7, 16] suggest that recovering this vector is hard for BKZ 2.0 algorithm when $q^{\frac{1}{4n}}\beta^{-\frac{1}{2n}} \lesssim 1.005$.

Remark 5. This lattice is a little different from those used in instantiating the unique shortest vector problem, as in our lattice, there are roughly n unique shortest non-zero vectors of similar length. Previous results in [15, 16] show that the hardness of finding a short vector in q -ary lattices that contain many unique shortest vectors depends not on the gap, but rather on the ratio between the Gaussian heuristic and the actual length of the shortest vector. We conjecture a similar property applies to our lattice.

2.4.2 A Non-Lattice Attack on Small Solutions

There are two pieces of structure lurking within the isomorphism $\mathbb{X} \rightarrow \mathbb{Y}$ that are not used in the lattice attack described in Sect. 2.4.1:

1. The map $\mathbb{X} \rightarrow \mathbb{Y}$ is a field isomorphism between two copies of \mathbb{F}_{q^n} , not merely an \mathbb{F}_q -vector space isomorphism between two copies of \mathbb{F}_q^n ;
2. The secret polynomial $\mathbf{f}(x)$ used to define one of the copies of \mathbb{F}_{q^n} has small coefficients. (And the attacker may, in principle, take $\mathbf{F}(y)$ to be any irreducible polynomial that she chooses.)

In this section we explain how to exploit these properties to formulate an attack that requires finding small solutions to systems of higher degree multivariable polynomial equations. We note that solving such systems appears to be exponentially difficult. The polynomials $\mathbf{f}(x)$ and $\mathbf{F}(y)$ almost, but not quite, determine the polynomials $\phi(y)$ and $\psi(x)$ used to define the isomorphism

$$\mathbb{F}_q[x]/(\mathbf{f}(x)) \cong \mathbb{F}_q[y]/(\mathbf{F}(y)).$$

More precisely, if $x \rightarrow \phi'(y)$ is some other isomorphism, then necessarily

$$\phi'(y) = \phi(y)^{q^t} \pmod{\mathbf{F}(y)} \quad \text{for some } 0 \leq t < d.$$

This follows immediately from the fact that $\text{Gal}(\mathbb{F}_{q^d}/\mathbb{F}_q)$ is cyclic of order d , generated by the q -power Frobenius map. Alternatively, the possible values for $\phi(y)$ are exactly the roots of $\mathbf{f}(x)$ in the field $\mathbb{F}_q[y]/(\mathbf{F}(y))$, so in any case there are exactly d possible $\phi(y)$'s. As stated in Remark 4, an attacker knows no useful

information about $\mathbf{f}(x)$ until she acquires an image, since as already noted, the public value $\mathbf{F}(y)$ is chosen independently of $\mathbf{f}(x)$. We assume that the attacker is given the value of an arbitrary number of images. As per Definition 1, the attacker is given $\mathbf{A}_1, \dots, \mathbf{A}_k \in \mathbb{Y}$ with the promise that $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathbb{X}$ are small, in other words:

$$\mathbf{A}_i(y) = \mathbf{a}_i(\phi(y)) \bmod \mathbf{F}(y), \quad (3)$$

where \mathbf{a}_i has small coefficients. The Eq. (3) contain $2n$ quantities that are unknown to the attacker, namely the coefficients of \mathbf{a} and ϕ . Of these, the coefficients of \mathbf{a} are small, so she can try to eliminate the coefficients of ϕ . We note that (3) really gives n equations for the coefficients, since both sides are polynomials of degree $n-1$. Unfortunately, this doesn't quite allow her to eliminate all n of the coefficients of ϕ . If she uses both $\mathbf{A}_1(y)$ and $\mathbf{A}_2(y)$, then she obtains $2n$ equations for the $3n$ unknowns consisting of the coefficients of \mathbf{a}_1 , \mathbf{a}_2 , and ϕ . So using elimination theory (as a practical matter, using Gröbner basis algorithms), she can eliminate the coefficients of ϕ and obtain a system of n equations for the $2n$ coefficients of \mathbf{a}_1 and \mathbf{a}_2 . These are highly non-linear equations over the field \mathbb{F}_q , so the attacker is faced with the problem of finding an \mathbb{F}_q -point with small coordinates on a high degree n -dimensional subvariety of \mathbb{F}_q^{2n} . As far as we are aware, there are no algorithms to solve such problems that are faster than an exhaustive (or possibly collision-based) search. Indeed, there does not appear to be an efficient algorithm to solve the decision problem of whether a small solution exists.

We note that the attacker may continue eliminating variables until eventually arriving at a single equation in \mathbb{F}_q^{n+1} . But this is likely to be counter-productive, since it greatly increases the degree of the underlying equation while discarding the information that the eliminated variables are small. Alternatively, the attacker can use one element in \mathbb{Y} and the knowledge that there is a polynomial $\mathbf{f}(x)$ with small coefficients that satisfies

$$\mathbf{f}(\phi(y)) = 0 \bmod \mathbf{F}(y). \quad (4)$$

Thus (3) and (4) again provide $2n$ equations, this time for the $3n$ coefficients of \mathbf{a} , \mathbf{f} , and ϕ . The first two polynomials have small coefficients, so eliminating the coefficients of ϕ again yields an n -dimensional subvariety in \mathbb{F}_q^{2n} on which the attacker must find a small point.

3 Fully Homomorphic Encryption Based on DFFI

In this section we use the approach of López-Alt et al. [28] to show how to turn our scheme into a fully homomorphic encryption scheme. First, we present Gentry's definitions and theorems on fully homomorphic encryption [17, 18]. Later, we show that our scheme satisfies the definitions on somewhat homomorphism, but it does not reach the circuit depth required for evaluating decryption circuit homomorphically. We resolve the issue by turning our scheme into a leveled

homomorphic encryption scheme using a technique to reduce the noise growth from doubly exponential to singly exponential. We then describe our leveled homomorphic scheme and show that it is fully homomorphic by showing that it is able to evaluate its decryption circuit homomorphically.

3.1 Fully Homomorphic Encryption Definitions

We give the definitions of fully homomorphic encryption and leveled homomorphic encryption.

Definition 31 (*C*-Homomorphic Encryption [6]). *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a class of functions with security parameter κ . A scheme \mathcal{E} is *C*-homomorphic if for any sequence of functions $f_\kappa \in \mathcal{C}_\kappa$ and respective inputs $\mu_1, \dots, \mu_\ell \in \{0, 1\}$ (where $\ell = \ell(\kappa)$), it is true that*

$$PR[\mathcal{E}.\text{Dec}_{sk}(\mathcal{E}.\text{Eval}_{evk}(f, c_1, \dots, c_\ell)) \neq f(\mu_1, \dots, \mu_\ell)] = \text{negl}(\kappa),$$

where $(pk, evk, sk) \leftarrow \mathcal{E}.\text{KeyGen}(1^\kappa)$ and $c_i \leftarrow \mathcal{E}.\text{Enc}_{pk}(\mu_i)$.

Definition 32 (Fully Homomorphic Encryption [28]). *An encryption scheme \mathcal{E} is fully homomorphic if it satisfies the following properties:*

Correctness: \mathcal{E} is *C*-homomorphic for the class \mathcal{C} of all circuits.

Compactness: *The computational complexity of \mathcal{E} 's algorithms is polynomial in the security parameter κ , and in the case of the evaluation algorithm, i.e. the size of the circuit.*

Now as given in [28], we continue with the leveled homomorphic encryption definition that is taken from [5]. It is a modified definition of fully homomorphic encryption (Definition 32) into a leveled homomorphic encryption scheme. It removes the requirement that the scheme is able to evaluate all possible circuits and instead imposes a circuit depth D . It requires the scheme to be able to evaluate all circuits (including the decryption circuit) that are depth at most D .

Definition 33 (Leveled Homomorphic Encryption [28]). *Let $\mathcal{C}^{(D)}$ be the class of all circuits of depth at most D (that use some specified complete set of gates). We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(D)} : D \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $D \in \mathbb{Z}^+$, it satisfies the following properties:*

Correctness: $\mathcal{E}^{(D)}$ is $\mathcal{C}^{(D)}$ -homomorphic.

Compactness: *The computational complexity of $\mathcal{E}^{(D)}$'s algorithms is polynomial in the security parameter κ and D , and in the case of the evaluation algorithm, the size of the circuit. We emphasize that this polynomial must be the same for all D .*

3.2 Somewhat Homomorphic FF-Encrypt Construction

We present a somewhat homomorphic version of our FF-Encrypt construction. We first give the details of our construction, and then we prove that our scheme is able to evaluate homomorphic circuits (multiplications and additions) of bounded depth.

3.2.1 Preliminaries

Here we give some preliminary notation and information that we use for the construction of our homomorphic schemes:

- The error distribution χ is a truncated Gaussian distribution $D_{\mathbb{Z}_r^n}$ with standard deviation r .
- The random polynomials $\mathbf{r}(x)$ are ephemeral short noise polynomials that are sampled from χ .
- The message space uses a fixed polynomial $\mathbf{p}(x)$, which we take for this instantiation to be the number 2.
- The message $\mathbf{m}(x)$ consists of a monomial with a single coefficient that is chosen from $\{0, 1\}$.

Polynomial Multiplication Noise in \mathbb{X} . The noise of the product of two polynomials is significantly affected by the choice of the polynomial $\mathbf{f}(x)$. Two factors that affect noise growth are the choice of the coefficient bound β_f for $\mathbf{f}(x)$ and the degree $d := \deg(\mathbf{f}'(x))$, where we write $\mathbf{f}(x) = x^n + \mathbf{f}'(x)$. The noise bound for the product of two β -bounded polynomial $\mathbf{a}(x)$ and $\mathbf{b}(x)$ for $d < n/2$ satisfies

$$\|\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)\|_\infty \leq n[(d+1)^2 + 1]\beta^2. \quad (5)$$

A detailed noise analysis for general $\mathbf{f}(x)$ is given in Appendix B.

3.2.2 Secret-Key Instantiation

The secret key version of our Somewhat Homomorphic Finite Field scheme uses the following four algorithms:

- SHFF-SK.Keygen(1^κ):
 - Input a security parameter κ .
 - Generate a parameter set $\Xi = \{n, q, \beta\}$ as a function of κ .
 - Use Algorithm 1 (from the FF-Encrypt paper) to generate a finite field homomorphism $\{\mathbf{f}, \mathbf{F}, \psi, \phi\}$.
 - Output $\{\mathbf{f}, \mathbf{F}, \psi, \phi\}$. Also output $\mathbf{p}(x)$ and $\gamma > 0$.
- SHFF-SK.Enc($\mathbf{f}, \mathbf{F}, \phi, \mathbf{m}$):
 - Encode a plaintext by some method into a short polynomial $\mathbf{m}(x) \in \mathbb{X}$;
 - Sample a polynomial $\mathbf{r}(x) \in \mathbb{X}$ from the distribution χ_β .
 - Compute $\mathbf{C}(y) = \mathbf{p}(\phi(y))\mathbf{r}(\phi(y)) + \mathbf{m}(\phi(y)) \bmod \mathbf{F}(y)$.
 - Output $\mathbf{C}(y)$ as the ciphertext.
- SHFF-SK.Dec($\mathbf{f}, \psi, \mathbf{C}$):
 - For a ciphertext $\mathbf{C}(y)$, compute $\mathbf{c}'(x) = \mathbf{C}(\psi(x))$.
 - Output $\mathbf{m}'(x) = \mathbf{c}'(x) \bmod (\mathbf{p}(x), \mathbf{f}(x))$.
- SHFF-SK.Eval($C, C_1, C_2, \dots, C_\ell$):
 - The circuit C is represented by two input binary arithmetic circuits with gates $\{+, \times\}$. Then, we can evaluate the circuit C homomorphically, since we can perform homomorphic addition and homomorphic multiplication.

3.2.3 Public-Key Instantiation

The public key version of our Somewhat Homomorphic Finite Field scheme is similar to the secret key instantiation in most aspects. We use a subset sum problem to instantiate the public key version. The scheme uses the following four algorithms:

- SHFF-PK.Keygen(1^κ):
 - Perform the key generation as in secret key instantiation SHFF-SK.Keygen(1^κ).
 - Choose two integers S, s which $\binom{S}{s} > 2^\kappa$ for security parameter κ .
 - Set $c_i = \text{SHFF-SK.Enc}(\mathbf{f}, \mathbf{F}, \phi, 0)_i$, create an array of zero encryptions $\text{pk} = \mathcal{S} = \{\mathbf{C}_0(y), \mathbf{C}_1(y), \dots, \mathbf{C}_{S-1}(y)\}$.
- SHFF-PK.Enc(\mathcal{S}, \mathbf{m}):
 - Choose s random encryptions of zero $\mathbf{C}_i(y)$ from \mathcal{S} and compute their summation with message $\mathbf{C}(y) = \sum_{i=\text{rand}(\mathcal{S})} \mathbf{C}_i(y) + \mathbf{M}(y)$ in which \mathbf{M} is the representation of the message m in \mathbb{Y} .
 - Output $\mathbf{C}(y)$ as the ciphertext.
- SHFF-PK.Dec($\mathbf{f}, \psi, \mathbf{C}$):
 - Compute and output SHFF-SK.Dec($\mathbf{f}, \psi, \mathbf{C}$).
- SHFF-PK.Eval($C, \mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_\ell$):
 - Compute and output SHFF-SK.Eval($C, \mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_\ell$).

The noise and depth performance of this scheme is captured by the following Lemma.

Lemma 2. *The encryption scheme*

$$\mathcal{E}_{\text{SHFF}} = (\text{SHFF.KeyGen}, \text{SHFF.Enc}, \text{SHFF.Dec}, \text{SHFF.Eval})$$

described above is somewhat homomorphic for circuits having depth less than $D < \log \log q - \log(3 \log n)$ where $q = 2^{n^\varepsilon}$ with $\varepsilon \in (0, 1)$, and χ is a β -bounded Gaussian distribution for random sampling.

Proof. We denote the encryptions of two messages \mathbf{m}_1 and \mathbf{m}_2 by $\mathbf{C}_1(y)$ and $\mathbf{C}_2(y)$. Then we want the noise of the ciphertexts after an addition or a multiplication to be smaller than $q/2$ so that it can be correctly decrypted.

Addition. Set $\mathbf{C}(y) = \mathbf{C}_1(y) + \mathbf{C}_2(y)$. Dropping y from the notation, we have $\mathbf{C} = (\sum \mathbf{p}(\phi)\mathbf{r}_1(\phi) + \mathbf{m}_1(\phi)) + (\sum \mathbf{p}(\phi)\mathbf{r}_2(\phi) + \mathbf{m}_2(\phi))$. Apply $\psi(x)$ as the first step of the decryption $\mathbf{C}(x) = (\sum \mathbf{p}(x)\mathbf{r}_1(x) + \mathbf{m}_1(x)) + (\sum \mathbf{p}(x)\mathbf{r}_2(x) + \mathbf{m}_2(x))$. Then the infinity norm of $\mathbf{C}(x)$ is $\|\mathbf{C}(x)\|_\infty = 2s\beta'$.

Multiplication. We compute

$$\begin{aligned} \mathbf{C} &= \left(\sum \mathbf{p}(\phi)\mathbf{r}_1(\phi) + \mathbf{m}_1(\phi) \right) \cdot \left(\sum \mathbf{p}(\phi)\mathbf{r}_2(\phi) + \mathbf{m}_2(\phi) \right) \\ &= \sum \mathbf{p}(\phi)^2 \mathbf{r}_1(\phi)\mathbf{r}_2(\phi) + \sum \mathbf{p}(\phi)\mathbf{r}_1(\phi)\mathbf{m}_2(\phi) \\ &\quad + \sum \mathbf{p}(\phi)\mathbf{r}_2(\phi)\mathbf{m}_1(\phi) + \mathbf{m}_1(\phi)\mathbf{m}_2(\phi). \end{aligned}$$

We calculate the infinity norm of $\mathbf{C}(x)$ using Eq. 5,

$$\|\mathbf{C}(x)\|_\infty = n((d+1)^2 + 1)(s\beta')^2 + 2s\beta'.$$

Multiplicative Level D . For D -level homomorphic operations, we need to compute the bound of $\|(\mathbf{p}(x)\mathbf{r}(x) + \mathbf{m}(x))^{2^D}\|_\infty$. Since $\mathbf{p}(x)\mathbf{r}(x) \gg \mathbf{m}(x)$, this is essentially equal to $\|(\mathbf{p}(x)\mathbf{r}(x))^{2^D}\|_\infty$. This gives an error bound equal to $(nd')^{2^D-1}(s\beta')^{2^D}$ with $d' = (d+1)^2+1$. We want this noise to be smaller than $q/2$, so we impose the inequality $(nd')^{2^D-1}(s\beta')^{2^D} < q/2$. Taking the logarithms, we rewrite this as $(2^D - 1) \log(nd') + (2^D) \log(s\beta') < \log q - 1$. Taking logarithm again yields $D + \log(\log(nd') + \log(s\beta')) < \log(\log q + \log(nd') - 1)$. We can simplify this inequality by noting that $d' \approx n^2/4$, which makes $\log(nd') \approx 3 \log(n) > \log(s\beta')$ and $\log(q) > 3 \log(n)$. Omitting small terms, we obtain

$$D < \log \log q - \log(3 \log n)$$

Taking $q = 2^{n^\epsilon}$, our upper bound for the multiplicative depth D is $\mathcal{O}(\epsilon \log n)$. \square

3.2.4 Security

Our construction relies on two security assumptions. The first assumption is the hardness of the Decisional Finite Field Isomorphism problem, which ensures that small norm elements in \mathbb{X} are mapped to random-looking elements in \mathbb{Y} . The mapping function is secret, and an attacker has to find some way of identifying images of short objects in \mathbb{X} in order to break the scheme. The second assumption is the difficulty of the subset sum problem that is used to generate encryptions of 0 to add to encryptions of messages. We will choose s ciphertexts from a list length S , so the pair of parameters (S, s) should give reasonable combinatorial security, e.g., $\binom{S}{s} > 2^{256}$. Beyond combinatorial security, solving this subset sum problem and identifying an encryption of 0 can be translated into a lattice reduction problem in the rows of an S by $S + n$ matrix, which can be made arbitrarily difficult. In particular $S > 2n$ should suffice. We prove the semantic security via the following theorem.

Theorem 1. *If there is an algorithm \mathcal{A} that breaks the semantic security with parameter $\Xi = \{n, q, \beta\}$ and $\mathbf{p}(x) = p$, i.e., if one inputs of any public keys $(\mathbf{C}_1, \dots, \mathbf{C}_k)$, a ciphertext \mathbf{D} which encrypts a message m of either 0 or 1, and \mathcal{A} outputs the message m with probability $1/2 + \epsilon$ for some non-negligible $\epsilon > 0$, then there exist another algorithm \mathcal{B} that solves the decisional FFI with parameter $\{n, q, \beta/p\}$ with probability $1/2 + \epsilon$.*

Proof. Notice that if the input $(\mathbf{C}_1, \dots, \mathbf{C}_k, \mathbf{D})$ to algorithm \mathcal{A} is invalid (either \mathbf{D} cannot be written as subset sum of \mathbf{C}_i , or \mathbf{D} does not encrypt 0 or 1), it will either output an error or output 0 or 1 with equal probability. On the other hand, if the input is valid, it will output the correct m with probability $1/2 + \epsilon$.

Now we can use \mathcal{A} to build an algorithm \mathcal{B} as follows. Let $\mathbf{A}_1, \dots, \mathbf{A}_k, \mathbf{B}_1, \mathbf{B}_2$ be the input to the decisional FFI problem. Upon receiving those inputs, algorithm \mathcal{A} calls algorithm \mathcal{B} with a “public key” $(p\mathbf{B}_1, p\mathbf{A}_2, \dots, p\mathbf{A}_k)$ and a ciphertext $\mathbf{0}$. Therefore, if \mathbf{B}_1 has short images in \mathbb{X} , then $(p\mathbf{B}_1, p\mathbf{A}_2, \dots, p\mathbf{A}_k)$ is a legit public key, while if \mathbf{B}_1 is uniformly sampled in $\mathbb{Z}_q[x]$, then the probability of $(p\mathbf{B}_1, p\mathbf{A}_2, \dots, p\mathbf{A}_k)$ been a legitimate public key is negligible, roughly $(\frac{\beta}{pq})^n$.

Notice that $\mathbf{0}$ is a subset sum of the “public key” regardless if the “public key” is legitimate or not. So from \mathcal{A} ’s point of view, $\mathbf{0}$ is a legit ciphertext that encrypts 0 if \mathbf{B}_1 has a short image. Upon receiving those public key and ciphertext, \mathcal{A} will return 0 with probability $1/2 + \epsilon$ if \mathbf{B}_1 has a short image. It will return error or random if \mathbf{B}_1 doesn’t. Thus \mathcal{B} solves the decisional FFI with probability $1/2 + \epsilon$. \square

For completeness sake, we also show that if one can solve the Decisional FFI, one can also break the semantic security. Given a ciphertext \mathbf{C} with an image $\mathbf{C} = \mathbf{p}\mathbf{r} + \ell\mathbf{m}$, one can compute $\mathbf{p}^{-1}\mathbf{C} \bmod q$ (assuming \mathbf{p} is an integer, say 2) which has a reverse image $\mathbf{r} + \mathbf{p}^{-1}\ell\mathbf{m}$. If $m = 0$, this quantity will be short. If $m = 1$, this quantity will be of length $\|\mathbf{p}^{-1}\ell \mathbf{r} \bmod q\|$. This is highly probable to be large, as if, say, $\mathbf{p} = 2$, then $\|\mathbf{p}^{-1} \bmod \mathbf{r} \bmod q\|$ will probably be of a size that takes random values mod q as ℓ varies.

3.3 From Somewhat to Fully Homomorphic Encryption

We give the definitions of bootstrappable scheme and weak circular security [17, 18]. Later, we use these two definitions to describe the bootstrapping theorem.

Definition 34 (Bootstrappable Scheme [18]). *Let $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})$ be a \mathcal{C} -homomorphic encryption scheme, and let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as*

$$\begin{aligned} f_{\text{add}}^{c_1, c_2}(\text{sk}) &= \text{Dec}(\text{sk}, c_1) \quad \text{XOR} \quad \text{Dec}(\text{sk}, c_2), \\ f_{\text{mult}}^{c_1, c_2}(\text{sk}) &= \text{Dec}(\text{sk}, c_1) \quad \text{AND} \quad \text{Dec}(\text{sk}, c_2). \end{aligned}$$

Then we say that \mathcal{E} is bootstrappable if $\{f_{\text{add}}^{c_1, c_2}, f_{\text{mult}}^{c_1, c_2}\}_{c_1, c_2} \subseteq \mathcal{C}$, i.e., if \mathcal{E} can homomorphically evaluate f_{add} and f_{mult} .

Definition 35 (Weak Circular Security [18]). *A public-key encryption scheme $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec})$ is weakly circular secure if it is IND-CPA secure even for an adversary with auxiliary information containing encryptions of all secret key bits: $\{\text{Enc}(\text{pk}, \text{sk}[i])\}_i$. In other words, no polynomial-time adversary can distinguish an encryption of 0 from an encryption of 1, even given this additional information.*

Theorem 2. *Let \mathcal{E} be a bootstrappable scheme that is also weakly circular secure. Then there exists a fully homomorphic encryption scheme \mathcal{E}' .*

In its current construction, our scheme is not bootstrappable, because it **cannot reach the required multiplicative depth for decryption**. For details on the evaluation of the depth of decryption circuit, see Sect. 3.3.5. The current scheme is only able to compute circuits with depth $\varepsilon \log(n)$. In order to convert our scheme into a bootstrappable one, in the next section we introduce a multiplication method with better noise management. This helps to significantly improve the depth of the circuits that the scheme can evaluate.

3.3.1 Regular Multiplication

A straightforward multiplication in the SHFF scheme causes the noise to grow doubly exponentially $(nd')^{2^D-1}(\mathfrak{s}\beta')^{2^D}$ with respect to the level D . To reduce the growth to singly exponential, we introduce a multiplication technique similar to the flattening in [21]. In rest of this section for notational simplicity, we drop x and y and represent elements of \mathbb{X} with lowercase letters and elements of \mathbb{Y} with uppercase letters, e.g., $\mathbf{r} \in \mathbb{X}$ and $\mathbf{R} \in \mathbb{Y}$ satisfy $\mathbf{r}(\phi(y)) = \mathbf{R}(y)$. We first consider the product for two ciphertexts, $\mathbf{C}_1 = \sum \mathbf{P}\mathbf{R}_1 + \mathbf{M}_1$ and $\mathbf{C}_2 = \sum \mathbf{P}\mathbf{R}_2 + \mathbf{M}_2$. To ease notation we write $\overline{\mathbf{R}} = \sum \mathbf{R}$. Then $\mathbf{C}_1 \cdot \mathbf{C}_2 = \mathbf{P}^2 \overline{\mathbf{R}}_1 \overline{\mathbf{R}}_2 + \mathbf{P}\overline{\mathbf{R}}_1 \mathbf{M}_2 + \mathbf{P}\overline{\mathbf{R}}_2 \mathbf{M}_1 + \mathbf{M}_1 \mathbf{M}_2$.

Remark 6. Obviously this method creates a significant noise term $\mathbf{P}^2 \overline{\mathbf{R}}_1 \overline{\mathbf{R}}_2 + \mathbf{P}\overline{\mathbf{R}}_1 \mathbf{M}_2 + \mathbf{P}\overline{\mathbf{R}}_2 \mathbf{M}_1$. If we map it back to \mathbb{X} , the norm of the noise is bounded by $\|\mathbf{p}^2 \mathfrak{s}^2 \mathbf{r}^2 + 2\mathfrak{p}\mathfrak{s}\mathbf{r}\|$ for $m \in \{0, 1\}$.

We look at the steps more closely. If we expand the second ciphertext $\mathbf{C}_2(y)$ and do not expand $\mathbf{C}_1(y)$, we obtain $\mathbf{C}_1 \cdot \mathbf{C}_2 = \mathbf{P}\overline{\mathbf{R}}_2 \mathbf{C}_1 + \mathbf{C}_1 \mathbf{M}_2$. Here $\mathbf{C}_1 \mathbf{M}_2$ gives the desired message product, with the side effect that the $\mathbf{P}\overline{\mathbf{R}}_2 \mathbf{C}_1$ term adds a significant amount of noise. To curb the noise growth, we have to find a way to evaluate $\mathbf{C}_1 \mathbf{M}_2$ while avoiding $\mathbf{P}\overline{\mathbf{R}}_2 \mathbf{C}_1$.

3.3.2 Multiplication with Noise Management

In this section we explain the idea behind computing the ciphertext product while avoiding the noisy $\mathbf{P}\overline{\mathbf{R}}_2 \mathbf{C}_1$ term. To achieve this we change the format of the ciphertexts and define two ciphertext operands: the Left-Hand-Side (LHS) and the Right-Hand-Side (RHS).

LHS Operand: The LHS-operand format is simply a matrix formed by bit decomposition of the ciphertext. We write $\hat{\mathbf{C}}_{\text{BD}}^m$ for the bit decomposition matrix of the ciphertext $\mathbf{C} = \mathbf{P}\overline{\mathbf{R}} + \mathbf{M}$ with message $\mathbf{m}(x)$. We denote the elements of the matrix by $\mathbf{C}_{i,j} = \hat{\mathbf{C}}_{\text{BD}}^m[i][j]$ for $0 < i < n$ and $0 < j < \ell$. More precisely, in the matrix, the entry $\mathbf{C}_{i,j}$ denotes the j^{th} bit of the i^{th} coefficient of \mathbf{C} . From this point on, we denote matrices by using a hat on top of the letters, e.g., $\hat{\mathbf{C}}$ means that it is a matrix.

RHS Operand: We create an n -by- ℓ matrix $\hat{\mathbf{C}}$, where each entry is a ciphertext that holds the message \mathbf{m} with a specific construction. For simplicity we drop the indices on $\overline{\mathbf{R}}$, so each $\overline{\mathbf{R}}$ represents a different sample. Then, the entries of the matrix are computed as $\hat{\mathbf{C}}^m[i][j] = \mathbf{P}\overline{\mathbf{R}}_{i,j} + 2^i \psi(\phi)^j \mathbf{M}$ for $0 \leq i <$

n and $0 \leq j < \ell$. Note that with each new row, we multiply the message by 2, and for each new column, we increase the power of $\psi(\phi)$. Since $y = \psi(\phi)$, this matrix is equal to $\hat{\mathbf{C}}^m[i][j] = \mathbf{P}\bar{\mathbf{R}}_{i,j} + 2^i y^j \mathbf{M}$ for $0 \leq i < n$ and $0 \leq j < \ell$.

One-Sided Homomorphic Multiplication: In the first method we use an LHS operand and an RHS operand to create an LHS operand, i.e., $\text{LHS} = \text{LHS} \times \text{RHS}$. The homomorphic product is computed by computing a component-wise product followed by a summation over the products:

$$\begin{aligned} \langle \hat{\mathbf{C}}_{\text{BD}}^{m_1}, \hat{\mathbf{C}}^{m_2} \rangle &= \sum_{i < n} \sum_{j < \ell} \mathbf{C}_{i,j} \cdot (\mathbf{P}\bar{\mathbf{R}}_{i,j} + 2^j y^i \mathbf{M}_2) \\ &= \sum \sum \mathbf{P}\bar{\mathbf{R}}_{i,j} + \mathbf{P}\bar{\mathbf{R}}_1 \mathbf{M}_2 + \mathbf{M}_1 \mathbf{M}_2. \end{aligned}$$

If we look more closely, each column in the component-wise product creates an encrypted version of the coefficients of the ciphertext \mathbf{C}_1 . The result of the product is a standard FF-Encrypt ciphertext. To continue using the result, we apply bit decomposition BD to obtain an LHS ciphertext. An LHS operand can be computed from a regular ciphertext on the fly via bit-decomposition. An RHS operand must be constructed before it is given to the cloud/server. This means that the ciphertext size grows by a factor of $n\ell$ for RHS operands only.

Remark 7. Noise growth in multiplications is significantly reduced compared to the earlier method. Using this one-sided multiplication approach and having fresh ciphertexts on the right-hand side, with flattening we obtain a new noise bound of $n\ell \|psr\|$. Therefore the noise growth is no longer doubly exponential, and we can support deep evaluations with reasonably sized parameters as long as we restrict evaluations to be one sided evaluations. This may be achieved by expressing the circuit first using NAND gates and then evaluating left to right similar to GSW.

Remark 8. Another significant contribution is that we eliminate polynomial multiplications and only perform polynomial additions. This way, the effect of $\mathbf{f}(x)$ is omitted for noise analysis, i.e., it does not have any effect on noise.

Lemma 3. *Let n be the polynomial degree, let $q = 2^{m^\epsilon}$ be the modulus, let $\chi = D_{\mathbb{Z}^n, r}$ be the β -bounded Gaussian distribution, and let D be the multiplicative level. Then, the proposed One-Sided Homomorphic Multiplication algorithm has noise bound $(2^D - 1)(n\ell + 1) \|psr\| = O(2^D n \log q)$ for fixed s and β .*

Generic Homomorphic Multiplication: This second method uses two RHS operands to do multiplication and achieves an RHS product as the result of the multiplication, i.e., $\text{RHS} = \text{RHS} \times \text{RHS}$. The multiplication is similar to the multiplication algorithm for LHS and RHS operands. We represent an element (ciphertext) in the RHS operand matrix as $\mathbf{C}^m[k][l]$ (k^{th} row and l^{th} column). In order to compute all the elements in the matrix we compute the following:

$$\begin{aligned} \mathbf{C}^{m_1 \cdot m_2}[k][l] &= \langle \hat{\mathbf{C}}_{\text{BD}}^{m_1}[k][l], \hat{\mathbf{C}}^{m_2} \rangle = \sum_{i < n} \sum_{j < \ell} \mathbf{C}_{i,j}[k][l] \cdot (\mathbf{P}\bar{\mathbf{R}}_{i,j} + 2^j y^i \mathbf{M}_2) \\ &= \sum \sum \mathbf{P}\bar{\mathbf{R}}_{i,j} + \mathbf{P}\bar{\mathbf{R}}_1 \mathbf{M}_2 + 2^k y^l \mathbf{M}_1 \mathbf{M}_2. \end{aligned}$$

Here we compute an element of the matrix using same approach that we used for LHS-RHS multiplication. We take an element in the matrix at any location (k, l) and apply the bit decomposition of that element $C_{BD}^{m_1}[k][l]$. Later, we compute component-wise products, which gives us the ciphertext result at location (k, l) in the result matrix. One $RHS \times RHS$ multiplication requires $n\ell$ multiplications of $LHS \times RHS$ type. Also, multiplication does not require one-sided evaluation as in the One-Sided Homomorphic Multiplication method. Since we can create an RHS operand, we can evaluate an arbitrary circuit, which gives an advantage over One-Sided Homomorphic Multiplication. The noise growth in multiplications is still low, but it accumulates as we compute depth D multiplication using a binary tree multiplication. This leads to a worse noise growth compared to LHS-RHS multiplication. But just as in method 1, we have still eliminated the effect of $f(x)$ on noise.

Lemma 4. *Let n be the polynomial degree, let $q = 2^{n^\epsilon}$ be the modulus, let $\chi = D_{\mathbb{Z}^n, r}$ is the β -bounded Gaussian distribution, and let D be the multiplicative level. Then, the proposed Generic Homomorphic Multiplication algorithm has noise bound $(n\ell + 1)^D \|psr\| = O((n \log q)^D)$ for fixed s and β .*

3.3.3 Leveled Homomorphic Public Key Scheme Instantiation

We construct a leveled homomorphic scheme using the noise management technique described above and the SHFF-PKscheme. Here we list the primitive functions of the Leveled Homomorphic Public Key scheme:

- LHFF-PK.Keygen(1^κ):
 - Compute SHFF-PK.Keygen(1^κ).
- LHFF-PK.Enc(\mathcal{S}, \mathbf{m}):
 - We form n by ℓ ciphertext matrix \hat{C} by computing its elements $C(y)[i][j] = \text{SHFF-PK.Enc}(\mathcal{S}, 2^i \psi^j \mathbf{m})$ for $i < \ell$ and $j < n$.
 - Output \hat{C} as the ciphertext.
- LHFF-PK.Dec($\mathbf{f}, \psi, \hat{C}$):
 - Compute SHFF-PK.Dec($\mathbf{f}, \psi, C[0][0]$).
- LHFF-PK.Eval($C, \hat{C}_1, \hat{C}_2, \dots, \hat{C}_\ell$):
 - We follow a similar approach to that we used in SHFF-SK. We show that the homomorphic properties are preserved under the binary circuit evaluation with gates $\{+, \times\}$. This proves that any circuit C can be evaluated using two gates with two binary inputs.

Homomorphic Addition (+). Homomorphic addition of two ciphertext matrices \hat{C}_1 and \hat{C}_2 is evaluated by performing a matrix addition, $\hat{C} = \hat{C}_1 + \hat{C}_2$. Namely, we compute the elements of the ciphertext matrix at each location (k, l) by computing $C(y)[k][l] = C_1(y)[k][l] + C_2(y)[k][l] \pmod{F(y)}$. The summation at each location preserves the ciphertext matrix property, $C[k][l] = (P\bar{R}_1 + 2^k y^l M_1) + (P\bar{R}_2 + 2^k y^l M_2)$, which simplifies to $C[k][l] = P(\bar{R}_1 + \bar{R}_2) + 2^k y^l (M_1 + M_2)$. This shows that the ciphertext property of the

matrix holds. Also, the first element $C[0][0]$ is decryptable and gives us the result of the summation.

Homomorphic Multiplication (\times). Homomorphic multiplication is evaluated using the multiplication method that is explained in Sect. 3.3.2. A matrix ciphertext multiplication preserves its format, which allows it to continue the homomorphic process. This may be seen by comparing the format of a fresh ciphertext and a product of ciphertexts. First we recall the format of an element of a fresh ciphertext: $C^{m_1}[k][l] = PR_1 + 2^k y^l M_1$. Next we recall the result of multiplication using multiplication method 2:

$$C^{m_1 \cdot m_2}[k][l] = \langle \hat{C}_{BD}^{m_1}[k][l], \hat{C}^{m_2} \rangle = \sum \sum PR_{i,j} + PR_1 M_2 + 2^k y^l M_1 M_2.$$

When we compare the ciphertext elements, it is clear that in a multiplication, we preserve the ciphertext matrix format while computing the multiplication, i.e., $2^k y^l M_1 M_2$. Also, in order to decrypt successfully, we need only decrypt the first element $C[0][0]$ of the matrix.

Multiplicative Level D . We capture the multiplicative depth of the leveled homomorphic scheme as follows.

Lemma 5. *The encryption scheme*

$$\mathcal{E}_{LH}\{\text{LHFF - PK.KeyGen, LHFF - PK.Enc, LHFF - PK.Dec, LHFF - PK.Eval}\}$$

described above is leveled homomorphic for circuits having depth $D = O(n^\varepsilon/\log n)$ where $q = 2^{n^\varepsilon}$ with $\varepsilon \in (0, 1)$, and χ is a β -bounded Gaussian distribution for random sampling.

Proof. In order to determine an upper bound for depth D , we use the noise bound that is calculated in Sect. 3.3.2. The noise has a bound $(n \log q + 1)^D \|pr\|$, which is equal to $(n \log q + 1)^D (s\beta')$. We require that this be smaller than $q/2$, which gives an upper bound for multiplicative level D in the form $(n \log q + 1)^D (s\beta') < q/2$. Taking the logarithm of both sides gives $D \log(n \log q + 1) + \log(s\beta') < \log q - 1$. Since $1 \ll n \log q$, using $q = 2^{n^\varepsilon}$ yields

$$D < \frac{n^\varepsilon - 1 - \log(s\beta')}{\log n + \varepsilon \log n}.$$

In big- \mathcal{O} notation, this gives an upper bound of the form $O(n^\varepsilon/\log n)$. \square

3.3.4 Security

The construction of the leveled homomorphic encryption is based on the Somewhat Homomorphic Finite Field Encryption scheme. Since there is not any significant change that affects the security, the leveled version of our construction is based on the same security assumptions as SHFF-PK: the hardness of the Decisional FFI and the subset sum problems.

Lemma 6. *Let n be the polynomial degree, let $q = 2^{n^\varepsilon}$ be the modulus, and let $\chi = D_{\mathbb{Z}^n, r}$ be a Gaussian distribution. Then, the proposed leveled homomorphic encryption scheme*

$$\mathcal{E}_{\text{LH}}\{\text{LHFF} - \text{PK.KeyGen}, \text{LHFF} - \text{PK.Enc}, \text{LHFF} - \text{PK.Dec}, \text{LHFF} - \text{PK.Eval}\}$$

is secure under the assumptions of hardness of the Decisional Finite Field Isomorphism problem and the subset sum problem.

3.3.5 Bootstrapping

In order to demonstrate that \mathcal{E} is fully homomorphic, we show that the depth of the decryption circuit can be homomorphically achieved by our scheme. First we look at the depth of the decryption circuit.

Decryption Circuit Depth. We recall that decryption is given by evaluating $\mathbf{c}'(x) = \mathbf{C}(\boldsymbol{\psi}(x)) \pmod{\mathbf{p}(x), \mathbf{f}(x)}$. Denoting the coefficients of $\mathbf{C}(y)$ by ζ_i , this can be expanded as $\mathbf{c}'(x) = \zeta_0 + \zeta_1\boldsymbol{\psi}(x) + \zeta_2\boldsymbol{\psi}(x)^2 + \dots + \zeta_{n-1}\boldsymbol{\psi}(x)^{n-1} \pmod{\mathbf{f}(x), \mathbf{p}(x)}$. Modular reduction by $\mathbf{f}(x)$ can be avoided by pre-computing $\boldsymbol{\psi}'^{(i)}(x) = \boldsymbol{\psi}(x)^i \pmod{\mathbf{f}(x)}$. This turns decryption into summation of polynomials are multiplied by scalars, $\mathbf{c}'(x) = \sum_{i < n} \zeta_i \boldsymbol{\psi}'^{(i)}(x)$. Let \mathbf{c}'_j be the coefficients of the result $\mathbf{c}'(x)$. Then each coefficient is evaluated by computing $\mathbf{c}'_j = \sum_{i < n} \zeta_i \boldsymbol{\psi}'^{(i)}_j$ where $\boldsymbol{\psi}'^{(i)}_j$ denotes the j^{th} coefficient of $\boldsymbol{\psi}'^{(i)}$.

In [6, Lemma 4.5] the authors prove that evaluating the sum of n elements with $\log q$ bits results in circuit depth $O(\log n + \log \log q)$. They also show that they can do modular reduction mod q with circuit depth $O(\log n + \log \log q)$. Since $\mathbf{p}(x)$ is small, say $\mathbf{p}(x) = 2$, we can perform modular reduction mod \mathbf{p} by taking the first bit, which does not require any circuit. Therefore, the bootstrapping operation has an upper bound $O(\log n + \log \log q)$.

Theorem 3. *Let χ is a β -bounded distribution for $\beta = \text{poly}(n)$, and let $q = 2^{n^\varepsilon}$ for $0 < \varepsilon < 1$. Then there exists a fully homomorphic encryption scheme based on the leveled homomorphic encryption scheme $\mathcal{E} = \text{LHFF} - \text{PK}$ with the assumptions that scheme is secure under the Decisional Finite Field Isomorphism Problem and that it is weakly circular secure.*

Proof. The decryption circuit requires $O(\log n + \log \log q)$ depth, and our scheme can compute $O(n^\varepsilon / \log n)$ depth circuits (Lemma 5). Therefore, the following inequality is sufficient in order to be bootstrappable:

$$\mathcal{Y}(\log n + \log \log q) < n^\varepsilon / \log n$$

where $\mathcal{Y} > 0$ is used to capture the constants in the circuit. Since $0 < \varepsilon < 1$, in worst case scenario we obtain $2\mathcal{Y} < \log q / \log^2 n$. □

4 Conclusion

In this work we proposed a new conjectured hard problem: the finite field isomorphism problem. Informally, the FFI problem asks one to construct an explicit

isomorphism between two representations of a finite field, given only access to long (large norm) representations of field elements and the assurance of the existence of a representation where each of these elements has a short (low norm) expression. We formalized the FFI problem and study the effectiveness of various approaches, including lattice attacks and non-lattice algebraic techniques, for recovering the secret isomorphism. Relying on the assumed hardness of the decisional-FFI problem, we first presented a secret-key somewhat homomorphic encryption scheme. This was extended, using a subset-sum problem technique, to a public-key scheme. We briefly analyze the noise performance of both schemes and introduced a bit-decomposition-based noise managements scheme that allows us to reduce the noise growth to single exponential. This yielded a bootstrapable, and thus a fully homomorphic encryption scheme.

A Constructing the Inverse Isomorphism

The map defined by $x \mapsto \phi(y)$ is a field isomorphism. It follows that there is an inverse isomorphism, and that inverse isomorphism is determined by the image of y . So we write the inverse isomorphism as

$$y \mapsto \psi(x) = \sum_{i=0}^{n-1} c_i x^i, \quad (6)$$

and our goal is to determine the c_i coefficients. We know that the composition $y \mapsto \psi(x) \mapsto \psi(\phi(y))$ gives an automorphism of $\mathbb{F}_q[y]/(\mathbf{F}(y))$, so

$$\psi(\phi(y)) \equiv y \pmod{\mathbf{F}(y)}. \quad (7)$$

Hence it suffices to determine the (unique) polynomial $\psi(x)$ of degree less than n satisfying (7). Using the expression (6) for $\psi(x)$, we want to find c_i so that $\sum_{i=0}^{n-1} c_i \phi(y)^i \equiv y \pmod{\mathbf{F}(y)}$. We write each power $\phi(y)^i$ modulo $\mathbf{F}(y)$ as a polynomial of degree less than n . In other words, we use the known values of $\phi(y)$ and $\mathbf{F}(y)$ to write $\phi(y)^i = \sum_{j=0}^{n-1} a_{ij} y^j \pmod{\mathbf{F}(y)}$ for $0 \leq i < n$. Substituting this into $\psi(\phi(y))$ yields $\psi(\phi(y)) = \sum_{i=0}^{n-1} c_i \phi(y)^i \equiv \sum_{i=0}^{n-1} c_i \sum_{j=0}^{n-1} a_{ij} y^j \pmod{\mathbf{F}(y)} \equiv \sum_{j=0}^{n-1} \left(\sum_{i=0}^{n-1} a_{ij} c_i \right) y^j \pmod{\mathbf{F}(y)}$. Hence ψ will satisfy (7) if we choose c_0, \dots, c_{n-1} to satisfy $\sum_{i=0}^{n-1} a_{ij} c_i = 1$ if $j = 1$, and $\sum_{i=0}^{n-1} a_{ij} c_i = 0$ if $j = 0$. This is a system of n equations for the n variables c_0, \dots, c_{n-1} over the finite field \mathbb{F}_q , hence is easy to solve, which gives the desired polynomial $\psi(y)$.

B Noise Analysis

To estimate the noise, we need to find the effect of modular reduction operation (with $\mathbf{f}(x)$) on the norm. One way is to use Barrett's Reduction algorithm. In Barrett's algorithm, a precomputed factor $\mathbf{M}(x) = x^{2n}/\mathbf{f}(x)$ plays a key role in estimating the quotient of the division with the modulus. Therefore,

determining $\mathbf{M}(x)$ will give us the main contributing factor to the noise level. Our goal is to bound the norm of the factor $\mathbf{M}(x)$ as tightly as possible. We start by rearranging $\mathbf{M}(x) = \lfloor x^{2n} / \mathbf{f}(x) \rfloor = \left\lfloor \frac{x^n}{1 + \frac{\mathbf{f}'(x)}{x^n}} \right\rfloor$. Note that $\deg(\mathbf{f}'(x)) < n$ and the floor operator simply truncates the polynomial beyond the constant term. This allows us to write the Taylor Series expansion (polynomial equivalent for $1/(1+x)$) as follows $\mathbf{M}(x) = \left\lfloor x^n + \sum_{i=1}^{i=\ell} (-1)^i \frac{\mathbf{f}'(x)^i}{x^{(i-1)n}} \right\rfloor$. Set $d = \deg(\mathbf{f}'(x))$. Then, each element in the series contributes up to a polynomial degree in the summation. It is important to notice that since $n > d$ each term in the expansion of $\mathbf{M}(x)$ the degree is bounded by d (except of course the x^n term. Therefore $\deg(\mathbf{M}(x) - x^n) \leq d$. In the series expansion a power $\mathbf{f}'(x)^i$ contributes to the series as long as $(i-1)n \leq id$. For larger i values the new additive term is simply truncated away, i.e. has no effect on $M(x)$. Therefore in the summation we only need to consider up to a degree ℓ which is determined as follows $\ell = \lfloor n/(n-d) \rfloor$. In the special case of $d < n/2$ we have $\ell = 1$ and $\mathbf{M}(x) = 1 - \mathbf{f}'(x)$ and $\beta_M = \beta_f$. In the general case, to bound the norm of $\mathbf{M}(x)$, we have to find the largest possible value for each term in the expansion. Assume that we sample $\mathbf{f}'(x)$ from a β -bounded distribution. We first assume $\beta = 1$ and later generalize the worst and average case bounds to cover arbitrary β values.

B.1 Worst Case Analysis

For clarity we first consider the first few terms in the expansion and then generalize the contribution to an arbitrary term:

$\mathbf{f}'(\mathbf{x})$: Since this is a fresh polynomial, the coefficients are sampled from a β -bounded distribution. For $\beta = 1$ in the worst case all coefficients are set to 1, i.e. $\mathbf{f}'(x) = x^d + x^{d-1} + x^{d-2} + \dots + x^1 + 1$.

$\mathbf{f}'(\mathbf{x})^2 / \mathbf{x}^n$: Assume we compute the square of $\mathbf{f}'(x)$ using as schoolbook multiplication. It is easy to see that starting from the middle degree d , the coefficients of the result decrease as we go to lower and higher degrees. In other words, the coefficients of $\mathbf{f}'(x)^2$ are symmetric around the middle degree. Since $\beta = 1$, we can write the polynomial as $x^{2d} + 2x^{2d-1} + \dots + (d+1)x^d + \dots + 2x + 1$. The division by x^n eliminates the first n terms. This results in following polynomial $x^{2d-n} + 2x^{2d-n-1} + 3x^{2d-n-2} + \dots + (2d-n+1)x^0$. Since $d < n$ then $2d-n < d$ and thus the largest coefficient is the constant coefficient with value $(2d-n+1)$.

$\mathbf{f}'(\mathbf{x})^i / \mathbf{x}^{(i-1)n}$: We are now ready to generalize the approach to find the largest coefficient for a degree i . When computing $\mathbf{f}'(x)^i = \mathbf{f}'(x)^{i-1} \cdot \mathbf{f}'(x)$ since it is divided by $x^{(i-1)n}$, we only use the last $id - (i-1)n + 1$ coefficients of $\mathbf{f}'(x)^{i-1}$. We multiply $\mathbf{f}'(x)^{i-1}$ with each coefficient of $\mathbf{f}'(x)$ and only take the last $id - (i-1)n + 1$ coefficients. If $\beta_{i-1} = \max(\mathbf{f}'(x)^{i-1})$, then we add $id - (i-1)n + 1$ of $B_{i-1} \cdot \beta$ which makes the upper bound $(id - (i-1)n + 1) \cdot \beta_{i-1} \cdot \beta$. If we apply this recursively to compute for previous values of i , we achieve an upper bound $(id - (i-1)n + 1)^{i-1}$ for $\beta = 1$.

B.2 Worst Case for Arbitrary β

$\mathbf{f}'(\mathbf{x})^i/\mathbf{x}^{(i-1)n}$. We use the general formula as explained in the section above. For the current i we have $(id - (i-1)n + 1) \cdot \beta_{i-1} \cdot \beta$ as the upper bound. For any β , recursively we have β^i so the upper bound will be $(id - (i-1)n + 1)^{i-1} \cdot \beta^i$. The overall bound on $\mathbf{M}(x)$ is therefore $B_M = \|\mathbf{M}(x)\| \leq \sum_{i=1, \dots, \ell} (id - (i-1)n + 1)^{i-1} \beta^i$ where $B_0 = \beta$ and as established before $\ell = \lfloor n/(n-d) \rfloor$. Our goal is to bound the norm $\|\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)\|$ using Barrett Reduction. We assume both $\|\mathbf{a}(x)\|, \|\mathbf{b}(x)\| \leq \beta$ and $\deg(\mathbf{f}(x)) = n$. We compute the worst case noise bound using the following steps:

- Step 1. Compute $\mathbf{M}(x) = \lfloor x^{2n}/\mathbf{f}(x) \rfloor$ ($\mathbf{M}(x)$ is the quotient of the division). Also assume $\|\mathbf{M}(x)\| = \beta_M$.
- Step 2. Compute regular product $\mathbf{c}(x) = \mathbf{a}(x)\mathbf{b}(x)$. $\|\mathbf{c}(x)\| = n\beta^2$.
- Step 3. Estimate quotient of $\mathbf{c}(x)/\mathbf{f}(x)$ (dropping (x) for brevity) $\mathbf{q}_1 = \lfloor \mathbf{c}/x^n \rfloor$. Since we take half of \mathbf{c} , worst case noise still remains: $\|\mathbf{q}_1\| = n\beta^2$.
 $\mathbf{q}_2 = \mathbf{M}\mathbf{q}_1$. This yields $\|\mathbf{q}_2\| = (d+1) \cdot \beta_M \cdot n\beta^2 = n(d+1)\beta_M\beta^2$
 $\mathbf{q}_3 = \lfloor \mathbf{q}_2/x^n \rfloor$. Worst case noise remains the same as \mathbf{q}_2 : $\|\mathbf{q}_3\| = n(d+1)\beta_M\beta^2$
- Step 4. Fix the result using the lower half of $\mathbf{c}(x)$
 $\mathbf{r}_1 = \mathbf{c} \bmod x^n$, thus $\|\mathbf{r}_1\| = n\beta^2$,
 $\mathbf{r}_2 = \mathbf{q}_3\mathbf{f} \bmod x^n$ $\|\mathbf{r}_2\| = n \cdot (d+1)^2 \beta_M \beta^2 \cdot \beta_f$, where we choose $\|\mathbf{f}(x)\| = \beta_f$.
 $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2 = \mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)$. This gives us an overall bound of
 $\|\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)\| \leq n\beta^2 + n(d+1)^2\beta^2\beta_M\beta_f$

For $d < n/2$ and $\beta_f = 1$, we have $\beta_M = 1$ and the worst case norm simplifies to $\|\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)\| \leq n[(d+1)^2 + 1]\beta^2$. In the average case the noise norm can be approximated by $\|\mathbf{a}(x)\mathbf{b}(x) \bmod \mathbf{f}(x)\|_{avg} \approx n^{1/2}\beta^2 + n^{1/2}(d+1)\beta^2\beta_M\beta_f$.

C Sample Parameters and Their Security Estimates

In Table 1 we present some parameters for the somewhat homomorphic encryption scheme. The proposed parameter set does not take into account our noise management technique. We compute the levels (circuit depth) by doing straightforward multiplications. In all 5 examples, we choose $\beta = 2$ and $d = n/2$ (recall that d is the degree of $\mathbf{f}'(x)$ where $\mathbf{f}(x) = x^n + \mathbf{f}'(x)$). For each level we give a noise estimate and also give a maximum selectable q size.

To estimate the cost of BKZ 2.0, we follow the cryptanalysis in [2, 24]. We use Tables 2 and 3 to estimate the block size and the number of nodes for a given root Hermite factor. Then we use the following formula ([24], which is an interpolation of data reported in [7] to get the cost of BKZ 2.0).

$$\text{BKZCost}(dim, b, rounds) = \text{LogNodes}(b) + \log_2(\text{dimension} \cdot \text{rounds}) + 7.$$

We remark that in [2] the authors proposed to use (quantum) sieving, rather than enumeration with extreme pruning, to estimate the cost of BKZ 2.0. In this analysis, we stick to the original estimation model, to show a proof-of-concept that practical parameters can be derived for our scheme. We leave the parameter derivation under the more conservative model in [2] to future work.

Table 1. Sample parameters for somewhat homomorphic encryption

Level	n	$\log \textit{noise}$	$\log \textit{max}(q)$	Ciphertext size	Root of Ratio	BKZ 2.0 cost
1	256	13	15	0.4 KB	1.0060	$>2^{145}$
2	2048	50	83	12.5 KB	1.0065	$>2^{135}$
3	4096	127	161	63.5 KB	1.0066	$>2^{136}$
4	8192	293	317	293 KB	1.0066	$>2^{137}$
5	32768	698	1250	2.7 MB	1.0066	$>2^{139}$

Table 2. Required blocksize for target root Hermite factor [7]

Target root Hermite factor	1.01	1.009	1.008	1.007	1.006
Approximate block size	85	106	133	168	216

Table 3. Upper bounds on \log_2 number of nodes enumerated in one call to enumeration subroutine of BKZ 2.0 [7].

Block size b	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250
LogNodes(b)	39	44	49	54	60	66	72	78	84	96	99	105	111	120	127	134

D Testing Results for Observation 2

We test the soundness of Observation 2 as follows. We setup toy size isomorphisms with $n \in \{20, 30, 40, 80\}$ and $q \in \{1031, 2053, 2^{20} + 7\}$. For each test we generate a long transcript of elements in \mathbb{X} and \mathbb{Y} ; We examine the distribution of the coefficients in \mathbb{Y} and compare it with uniform distribution; We show that the Renyi divergence between our distribution and a uniform distribution scales properly with $\log_2(q/n)$. Two example distribution of the coefficients are shown

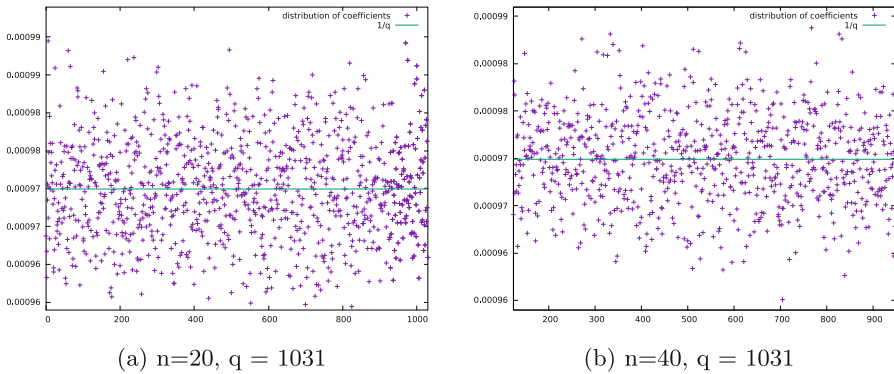


Fig. 1. Testing results for Observation 2

in Fig. 1. We compute the Renyi divergence with $\alpha = 2$. Our results shows that our distribution is less than 2^{-14} away from a uniform distribution for out toy example with $n = 20$ and $q = 1031$.

Table 4. Renyi divergence

q	$n = 20$	$n = 30$	$n = 40$	$n = 80$
1031	$2^{-14.3}$	$2^{-14.8}$	$2^{-15.3}$	$2^{-16.2}$
2053	$2^{-13.3}$	$2^{-13.9}$	$2^{-14.3}$	$2^{-15.3}$
$2^{20}+7$	$2^{-4.3}$	$2^{-4.8}$	$2^{-5.3}$	$2^{-6.2}$

We summarize the testing result in Table 4. As one can see the exponent of the divergence is linear in $\log_2(q/n)$. We estimate that for moderate $n \approx q$ the divergence is around 2^{-11} .

References

1. Ajtai, M.: The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract). In: Thirtieth Annual ACM Symposium on the Theory of Computing (STOC 1998), pp. 10–19 (1998)
2. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 327–343 (2016). <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim>
3. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45239-0_4
4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_50
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 18, p. 111 (2011)
6. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
7. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_1
8. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low level encoding of zero. Cryptology ePrint Archive, Report 2016/139 (2016). <https://eprint.iacr.org/2016/139>
9. Coron, J.-S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 311–328. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_18

10. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_28
11. Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_27
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
13. Doröz, Y., Hu, Y., Sunar, B.: Homomorphic AES evaluation using the modified LTV scheme. Des. Codes Cryptogr. 1–26 (2015). <https://doi.org/10.1007/s10623-015-0095-1>
14. Doröz, Y., Sunar, B.: Flattening NTRU for evaluation key free homomorphic encryption. Cryptology ePrint Archive, Report 2016/315 (2016). <http://eprint.iacr.org/2016/315>
15. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
16. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_3, <http://dl.acm.org/citation.cfm?id=1788414.1788417>
17. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009). crypto.stanford.edu/craig
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009)
19. Gentry, C., Halevi, S.: Implementing Gentry’s fully-homomorphic encryption scheme. In: EUROCRYPT, pp. 129–148 (2011)
20. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_49
21. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
22. Gentry, C., Szydlo, M.: Cryptanalysis of the revised NTRU signature scheme. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 299–320. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_20
23. Halevi, S., Shoup, V.: HELib, homomorphic encryption library (2012)
24. Hoffstein, J., Pipher, J., Schanck, J.M., Silverman, J.H., Whyte, W., Zhang, Z.: Choosing parameters for NTRUEncrypt. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_1
25. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054868>

26. Ireland, K., Rosen, M.: A Classical Introduction to Modern Number Theory. Springer, Heidelberg (1990). <https://doi.org/10.1007/978-1-4757-2103-4>
27. Kirchner, P., Fouque, P.-A.: Revisiting lattice attacks on overstretched NTRU parameters. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 3–26. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_1
28. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing, STOC 2012, pp. 1219–1234. ACM (2012). <https://doi.org/10.1145/2213977.2214086>
29. Albrecht, M., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes. Cryptology ePrint Archive, Report 2016/127 (2016). <http://eprint.iacr.org/>
30. May, A., Silverman, J.H.: Dimension reduction methods for convolution modular lattices. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 110–125. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44670-2_10
31. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014). <https://doi.org/10.1007/s10623-012-9720-4>
32. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_4
33. The PARI Group, Bordeaux: PARI/GP version 2.7.0 (2014). <http://pari.math.u-bordeaux.fr/>

Public-Key Encryption



Hybrid Encryption in a Multi-user Setting, Revisited

Federico Giacon¹(✉), Eike Kiltz¹, and Bertram Poettering²

¹ Ruhr University Bochum, Bochum, Germany
{federico.giacon,eike.kiltz}@rub.de

² Royal Holloway, University of London, London, UK
bertram.poettering@rhul.ac.uk

Abstract. This paper contributes to understanding the interplay of security notions for PKE, KEMs, and DEMs, in settings with multiple users, challenges, and instances. We start analytically by first studying (a) the tightness aspects of the standard hybrid KEM+DEM encryption paradigm, (b) the inherent weak security properties of all deterministic DEMs due to generic key-collision attacks in the multi-instance setting, and (c) the negative effect of deterministic DEMs on the security of hybrid encryption.

We then switch to the constructive side by (d) introducing the concept of an *augmented data encapsulation mechanism* (ADEM) that promises robustness against multi-instance attacks, (e) proposing a variant of hybrid encryption that uses an ADEM instead of a DEM to alleviate the problems of the standard KEM+DEM composition, and (f) constructing practical ADEMs that are secure in the multi-instance setting.

Keywords: Hybrid encryption · Multi-user security · Tightness

1 Introduction

HYBRID ENCRYPTION AND ITS SECURITY. Public-key encryption (PKE) is typically implemented following a hybrid paradigm: To encrypt a message, first a randomized key encapsulation mechanism (KEM) is used to establish— independently of the message—a fresh session key that the receiver is able to recover using its secret key; then a deterministic data encapsulation mechanism (DEM) is used with the session key to encrypt the message. Both KEM and DEM output individual ciphertexts, and the overall PKE ciphertext is just their concatenation. Benefits obtained from deconstructing PKE into the two named components include easier implementation, deployment, and analysis. An independent reason that, in many cases, makes separating asymmetric from symmetric techniques actually necessary is that asymmetric cryptographic components

The full version can be found in the IACR eprint archive as article 2017/843 (<https://eprint.iacr.org/2017/843>).

can typically deal only with messages of limited length (e.g., 2048 bit messages in RSA-based systems) or of specific structure (e.g., points on an elliptic curve). The paradigm of hybrid encryption, where the message-processing components are strictly separated from the asymmetric ones, side-steps these disadvantages.

Hybrid encryption was first studied on a formal basis in [11]. (Implicitly the concept emerged much earlier, for instance in PGP email encryption.) The central result on the security of this paradigm is that combining a secure KEM with a secure DEM yields a secure PKE scheme. Various configurations of sufficient definitions of ‘secure’ for the three components have been proposed [11, 16, 18], with the common property that the corresponding security reductions are tight.

MULTI-USER SECURITY OF PKE AND KEMS. Classic security definitions for PKE, like IND-CPA and IND-CCA, formalize notions of confidentiality of a single message encrypted to a single user. (For public-key primitives, we identify (receiving) users with public keys.) This does not well-reflect real-world requirements where, in principle, billions of senders might use the same encryption algorithm to send, concurrently and independently of each other, related or unrelated messages to billions of receivers. Correspondingly, for adequately capturing security aspects of PKE that is deployed at large scale, generalizations of IND-CPA/CCA have been proposed that formalize indistinguishability in the face of multiple users and multiple challenge queries [4] (the goal of the adversary is to break confidentiality of one message, not necessarily of all messages). On the one hand, fortunately, these generalized notions turn out to be equivalent to the single-user single-challenge case [4] (thus supporting the relevance of the latter). On the other hand, and unfortunately, all known proofs of this statement use reductions that are not tight, losing a factor of $n \cdot q_e$ where n is the number of users and q_e the allowed number of challenge queries per user. Of course this does not mean that PKE schemes with tightly equivalent single- and multi-user security cannot exist, and indeed [1, 4, 12, 15, 17, 19, 20] expose examples of schemes with tight reductions between the two worlds.

The situation for KEMs is the same as for PKE: While the standard security definitions [11, 16] consider exclusively the single-user single-challenge case, natural multi-user multi-challenge variants have been considered and can be proven—up to a security loss with factor $n \cdot q_e$ —equivalent to the standard notions.

MULTI-INSTANCE SECURITY OF DEMS. Besides scaled versions of security notions for PKE and KEMs, we also consider similarly generalized variants of DEM security. More specifically, we formalize a new¹ security notion for DEMs that assumes multiple independently generated instances and allows for one challenge encapsulation per instance. (For secret key primitives, we identify instances with secret keys.) The single-challenge restriction is due to the fact that overall we are interested in KEM+DEM composition and, akin to the single-instance

¹ We are not aware of prior work that explicitly develops multi-instance security models for DEMs; however, [22] (and others) discuss the multi-instance security of symmetric encryption, and [7] considers the multi-instance security of (nonce-based) AE.

case [11], a one-time notion for the DEM is sufficient (and, as we show, necessary) for proving security of the hybrid. As for PKE and KEMs, the multi-instance security of a DEM is closely coupled to its single-instance security; however, generically, if N is the number of instances, the corresponding reduction loses a factor of N .

A couple of works [8, 22] observe that DEMs that possess a specific technical property² indeed have a lower security in the multi-instance setting than in the single-instance case. This is shown via attacks that assume a number of instances that is so large that, with considerable probability, different instances use the same encapsulation key; such key collisions can be detected, and message contents can be recovered. Note that, strictly speaking, the mentioned type of attack does *not* imply that the reduction of multi-instance to single-instance security is necessarily untight, as the attacks crucially depend on the DEM key size which is a parameter that does not appear in above tightness bounds. We finally point out that the attacks described in [8, 22] are not general but target only specific DEMs. In this paper we show that the security of *any* (deterministic) DEM degrades as the number of considered instances increases.

1.1 Our Contributions

This paper contributes to understanding the interplay of security notions for PKE, KEMs, and DEMs, in settings with multiple users, challenges, and instances. We start analytically by first studying (a) the tightness aspects of the standard hybrid KEM+DEM encryption paradigm, (b) the inherent weak security properties of deterministic DEMs in the multi-instance setting, and (c) the negative effect of deterministic DEMs on the security of hybrid encryption. We then switch to the constructive side by (d) introducing the concept of an *augmented data encapsulation mechanism* (ADEM) that promises robustness against multi-instance attacks, (e) proposing a variant of hybrid encryption that uses an ADEM instead of a DEM to alleviate the problems of the standard KEM+DEM composition, and (f) constructing secure practical ADEMs. We proceed with discussing some of these results in more detail, in the order in which they appear in the paper.

STANDARD KEM+DEM HYBRID ENCRYPTION. In Sect. 3 we define syntax and security properties of PKE, KEMs, and DEMs; we also recall hybrid encryption. Besides unifying the notation of algorithms and security definitions, the main contribution of this section is to provide a new multi-instance security notion for DEMs that matches the requirements of KEM+DEM hybrid encryption in the multi-user multi-challenge setting. That is, hybrid encryption is secure, tightly, if KEM and DEM are simultaneously secure (in our sense). We further show that

² The cited work is not too clear about this property; loosely speaking the condition seems to be that colliding ciphertexts of the same message under random keys can be used as evidence that also the keys are colliding. One example for a DEM with this property is CBC encryption.

any attack on the multi-instance security of the DEM tightly implies an attack on the multi-user multi-challenge security of the hybrid scheme. This implication is particularly relevant in the light of the results of Sect. 4, discussed next.

GENERIC KEY-COLLISION ATTACKS ON DETERMINISTIC DEMS. In Sect. 4 we study two attacks that target arbitrary (deterministic) DEMs, leveraging on the multi-instance setting and exploiting the tightness gap between single-instance and multi-instance security. Concretely, inspired by the key-collision attacks (also known as birthday-bound attacks) from [7, 8, 22], in Sects. 4.1 and 4.2 we describe two attacks against arbitrary DEMs that break indistinguishability or even recover encryption keys with success probability $N^2/|\mathcal{K}|$, where N is the number of instances and \mathcal{K} is the DEM's key space. (The reason for specifying two attacks instead of just one is that deciding which one is preferable may depend on the particular DEM.) As mentioned above, in hybrid encryption these attacks carry over to the overall PKE.

What are the options to thwart the described attacks on DEMs? One way to avoid key-collision attacks in practice is of course to increase the key length of the DEM. This requires the extra burden of also changing the KEM (it has to output longer keys) and hence might not be a viable option. (Observe that leaving the KEM as-is but expanding its key to, say, double length using a PRG is *not* going to work as our generic DEM attacks would immediately kick in against that construction as well.) Another way to go would be to randomize the DEM. Drawbacks of this approach are that randomness might be a scarce resource (in particular on embedded systems, but also on desktop computers there is a price to pay for requesting randomness³), and that randomized schemes necessarily have longer ciphertexts than deterministic ones. In Sects. 5 to 7 we explore an alternative technique to overcome key-collision attacks in hybrid encryption without requiring further randomness and without requiring changing the KEM. We describe our approach in the following.

KEM+ADEM HYBRID ENCRYPTION. In Sect. 5 we introduce the concept of an augmented data encapsulation mechanism (ADEM). It is a variant of a DEM that takes an additional input: the tag. The intuition is that ADEMs are safer to use for hybrid encryption than regular DEMs, in particular in the presence of session-key collisions: Even if two keys collide, security is preserved if the corresponding tags are different. Importantly, the two generic DEM attacks from Sect. 4 do not apply to ADEMs. In Sect. 5 we further consider *augmented hybrid encryption*, which constructs PKE from a KEM and an ADEM by using the KEM ciphertext as ADEM tag. The corresponding security reduction is tight.

³ Obtaining entropy from a modern operating system kernel involves either file access or system calls; both options are considerably more costly than, say, doing an AES computation. While some modern CPUs have built-in randomness generators, the quality of the latter is difficult to assess and relying exclusively on them thus discouraged (see <https://plus.google.com/+TheodoreTso/posts/SDcoemc9V3J>).

PRACTICAL ADEM CONSTRUCTIONS. Sections 6 and 7 are dedicated to the construction of practical ADEMs. The two constructions in Sect. 6 are based on the well-known counter mode encryption, instantiated with an ideal random function and using the tag as initial counter value. We prove tight, beyond-birthday security bounds of the form $N/|\mathcal{K}|$ for the multi-instance security of our ADEMs. That is, our constructions provably do not fall prey to key collision attacks, in particular not the ones from [8, 22] and Sect. 4. Unfortunately, as they are based on counter mode, the two schemes *per se* are not secure against active adversaries. This is remedied in Sect. 7 where we show that an *augmented message authentication code*⁴ (AMAC) can be used to generically strengthen a passively-secure ADEM to become secure against active adversaries. (We define AMACs and give a tightly secure construction in the same section.)

2 Notation

If S is a finite set, $s \stackrel{\$}{\leftarrow} S$ denotes the operation of picking an element of S uniformly at random and assigning the result to variable s . For a randomized algorithm A we write $y \stackrel{\$}{\leftarrow} A(x_1, x_2, \dots)$ to denote the operation of running A with inputs x_1, x_2, \dots and assigning the output to variable y . Further, we write $[A(x_1, x_2, \dots)]$ for the set of values that A outputs with positive probability. We denote the concatenation of strings with $\|$ and the XOR of same-length strings with \oplus . If $a \leq b$ are natural numbers, we write $[a..b]$ for the range $\{a, \dots, b\}$.

We say a sequence v_1, \dots, v_n has a (two-)collision if there are indices $1 \leq i < j \leq n$ such that $v_i = v_j$. More generally, the sequence has a k -collision if there exist $1 \leq i_1 < \dots < i_k \leq n$ such that $v_{i_1} = \dots = v_{i_k}$. We use predicate $\mathbf{Coll}_k[\cdot]$ to indicate k -collisions. For instance, $\mathbf{Coll}_2[1, 2, 3, 2]$ evaluates to *true* and $\mathbf{Coll}_3[1, 2, 3, 2]$ evaluates to *false*.

Let \mathcal{L} be a finite set of cardinality $L = |\mathcal{L}|$. Sometimes we want to refer to the elements of \mathcal{L} in an arbitrary but circular way, i.e., such that indices x and $x + L$ resolve to the same element. We do this by fixing an arbitrary bijection $[\cdot]_L: \mathbb{Z}/L\mathbb{Z} \rightarrow \mathcal{L}$ and extending the domain of $[\cdot]_L$ to the set \mathbb{Z} in the natural way. This makes expressions like $[[a + b]]_L$, for $a, b \in \mathbb{N}$, well-defined. We use the shortcut notation $[[a \rightarrow l]]_L$ to refer to the span $\{[[a + 1]]_L, \dots, [[a + l]]_L\}$ of length l . In particular we have $[[a \rightarrow 1]]_L = \{[[a + 1]]_L\}$.

Our security definitions are based on games played between a challenger and an adversary. These games are expressed using program code and terminate when the main code block executes ‘return’; the argument of the latter is the output of the game. We write $\Pr[G \Rightarrow 1]$ or $\Pr[G \Rightarrow \text{true}]$ or just $\Pr[G]$ for the probability that game G terminates by executing a ‘return’ instruction with a value interpreted as true. Further, if E is some game-internal event, we write $\Pr[E]$ for the probability this event occurs. (Note the game is implicit in this notation.)

⁴ The notion of an augmented MAC appeared recently in an unrelated context: An AMAC according to [3] is effectively keyed Merkle–Damgård hashing with an unkeyed output transform applied at the end. Importantly, while the notion of [3] follows the classic MAC syntax, ours does not (for having a separate tag input).

3 Traditional KEM/DEM Composition and Its Weakness

We define PKE, KEMs, and DEMs, and give security definitions that consider multi-user, multi-challenge, and multi-instance attacks. Using the techniques from [4] we show that the multi notions are equivalent to their single counterparts, up to a huge tightness loss. We show that hybrid encryption enjoys tight security also in the multi settings. We finally show how (multi-instance) attacks on the DEM can be leveraged to attacks on the PKE.

3.1 Syntax and Security of PKE, KEMs, and DEMs

PUBLIC-KEY ENCRYPTION. A public-key encryption scheme $\text{PKE} = (\text{P.gen}, \text{P.enc}, \text{P.dec})$ is a triple of algorithms together with a message space \mathcal{M} and a ciphertext space \mathcal{C} . The randomized key-generation algorithm P.gen returns a pair (pk, sk) consisting of a public key and a secret key. The randomized encryption algorithm P.enc takes a public key pk and a message $m \in \mathcal{M}$ to produce a ciphertext $c \in \mathcal{C}$. Finally, the deterministic decryption algorithm P.dec takes a secret key sk and a ciphertext $c \in \mathcal{C}$, and outputs either a message $m \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$ to indicate rejection. The correctness requirement is that for all $(pk, sk) \in [\text{P.gen}]$, $m \in \mathcal{M}$, and $c \in [\text{P.enc}(pk, m)]$, we have $\text{P.dec}(sk, c) = m$.

We adapt results from [4] to our notation, giving a game-based security definition for public-key encryption that formalizes multi-user multi-challenge indistinguishability: For a scheme PKE, to any adversary \mathbf{A} and any number of users n we associate the distinguishing advantage $\text{Adv}_{\text{PKE}, \mathbf{A}, n}^{\text{muc-ind}} := |\Pr[\text{MUC-IND}_{\mathbf{A}, n}^0] - \Pr[\text{MUC-IND}_{\mathbf{A}, n}^1]|$, where the two games are specified in Fig. 1. Note that if q_e resp. q_d specify upper bounds on the number of Oenc and Odec queries per user, then the single-user configurations $(n, q_e, q_d) = (1, 1, 0)$ and $(n, q_e, q_d) = (1, 1, \infty)$ correspond to standard definitions of IND-CPA and IND-CCA security for PKE.

Game	MUC-IND $_{\mathbf{A}, n}^b$	Oracle Oenc(j, m_0, m_1)	Oracle Odec(j, c)
00	for all $j \in [1 \dots n]$:	05 $c \xleftarrow{\$} \text{P.enc}(pk_j, m_b)$	08 if $c \in \mathcal{C}_j$: return \perp
01	$(pk_j, sk_j) \xleftarrow{\$} \text{P.gen}$	06 $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{c\}$	09 $m \leftarrow \text{P.dec}(sk_j, c)$
02	$\mathcal{C}_j \leftarrow \emptyset$	07 return c	10 return m
03	$b' \xleftarrow{\$} \mathbf{A}(pk_1, \dots, pk_n)$		
04	return b'		

Fig. 1. PKE security games MUC-IND $_{\mathbf{A}, n}^b$, $b \in \{0, 1\}$, modeling multi-user multi-challenge indistinguishability for n users.

The following states that the multi-user multi-challenge notion is equivalent to the traditional single-user single-challenge case—up to a tightness loss linear in both the number of users and the number of challenges. The proof is in [4].

Lemma 1 [4]. *For any public-key encryption scheme PKE, any number of users n , and any adversary \mathbf{A} that poses at most q_e -many Oenc and q_d -many*

Odec queries per user, there exists an adversary B such that $\mathbf{Adv}_{\text{PKE},A,n}^{\text{muc-ind}} \leq n \cdot q_e \cdot \mathbf{Adv}_{\text{PKE},B,1}^{\text{muc-ind}}$, where B poses at most one Oenc and q_d -many Odec queries. Further, the running time of B is at most that of A plus the time needed to perform nq_e -many P.enc operations and nq_d -many P.dec operations.

KEY ENCAPSULATION. A key-encapsulation mechanism $\text{KEM} = (\text{K.gen}, \text{K.enc}, \text{K.dec})$ for a finite session-key space \mathcal{K} is a triple of algorithms together with a ciphertext space \mathcal{C} . The randomized key-generation algorithm K.gen returns a pair (pk, sk) consisting of a public key and a secret key. The randomized encapsulation algorithm K.enc takes a public key pk to produce a session key $K \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$. Finally, the deterministic decapsulation algorithm K.dec takes a secret key sk and a ciphertext $c \in \mathcal{C}$, and outputs either a session key $K \in \mathcal{K}$ or the special symbol $\perp \notin \mathcal{K}$ to indicate rejection. The correctness requirement is that for all $(pk, sk) \in [\text{K.gen}]$ and $(K, c) \in [\text{K.enc}(pk)]$ we have $\text{K.dec}(sk, c) = K$.

Like for PKE schemes we give a security definition for KEMs that formalizes multi-user multi-challenge indistinguishability: For a scheme KEM , to any adversary A and any number of users n we associate the distinguishing advantage $\mathbf{Adv}_{\text{KEM},A,n}^{\text{muc-ind}} := |\Pr[\text{MUC-IND}_{A,n}^0] - \Pr[\text{MUC-IND}_{A,n}^1]|$, where the two games are specified in Fig. 2. Note that if q_e resp. q_d specify upper bounds on the number of Oenc and Odec queries per user, then the single-user configurations $(n, q_e, q_d) = (1, 1, 0)$ and $(n, q_e, q_d) = (1, 1, \infty)$ correspond precisely to standard definitions of IND-CPA and IND-CCA security for KEMs.

Game	MUC-IND $^b_{A,n}$	Oracle Oenc(j)	Oracle Odec(j, c)
00	for all $j \in [1..n]$:	05 $(K^0, c) \xleftarrow{\$} \text{K.enc}(pk_j)$	09 if $c \in C_j$: return \perp
01	$(pk_j, sk_j) \xleftarrow{\$} \text{K.gen}$	06 $K^1 \xleftarrow{\$} \mathcal{K}$	10 $K \leftarrow \text{K.dec}(sk_j, c)$
02	$C_j \leftarrow \emptyset$	07 $C_j \leftarrow C_j \cup \{c\}$	11 return K
03	$b' \xleftarrow{\$} A(pk_1, \dots, pk_n)$	08 return (K^b, c)	
04	return b'		

Fig. 2. KEM security games $\text{MUC-IND}_{A,n}^b$, $b \in \{0, 1\}$, modeling multi-user multi-challenge indistinguishability for n users.

Akin to the PKE case, our KEM multi-user multi-challenge notion is equivalent to its single-user single-challenge relative—again up to a tightness loss linear in the number of users and challenges. The proof can be found in the full version [14].

Lemma 2. For any key-encapsulation mechanism KEM , any number of users n , and any adversary A that poses at most q_e -many Oenc and q_d -many Odec queries per user, there exists an adversary B such that $\mathbf{Adv}_{\text{KEM},A,n}^{\text{muc-ind}} \leq n \cdot q_e \cdot \mathbf{Adv}_{\text{KEM},B,1}^{\text{muc-ind}}$, where B poses at most one Oenc and q_d -many Odec queries. Further, the running time of B is at most that of A plus the time needed to perform nq_e -many K.enc operations and nq_d -many K.dec operations.

DATA ENCAPSULATION. A data-encapsulation mechanism $\text{DEM} = (\text{D.enc}, \text{D.dec})$ for a message space \mathcal{M} is a pair of deterministic algorithms associated with a finite key space \mathcal{K} and a ciphertext space \mathcal{C} . The encapsulation algorithm D.enc takes a key $K \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$. The decapsulation algorithm D.dec takes a key $K \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$, and outputs either a message $m \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$ to indicate rejection. The correctness requirement is that for all $K \in \mathcal{K}$ and $m \in \mathcal{M}$ we have $\text{D.dec}(K, \text{D.enc}(K, m)) = m$.

As a security requirement for DEMs we formalize a multi-instance variant of the standard one-time indistinguishability notion: In our model the adversary can request one challenge encapsulation for each of a total of N independent keys; decapsulation queries are not restricted and can be asked multiple times for the same key. The corresponding games are in Fig. 3. Note that lines 05 and 09 ensure that the adversary cannot ask for decapsulations with respect to a key before having a challenge message encapsulated with it. (This matches the typical situation as it emerges in a KEM/DEM hybrid.) For a scheme DEM, to any adversary A and any number of instances N we associate the distinguishing advantage $\text{Adv}_{\text{DEM}, A, N}^{\text{miot-ind}} := |\Pr[\text{MIOT-IND}_{A, N}^0] - \Pr[\text{MIOT-IND}_{A, N}^1]|$. Note that if Q_d specifies a global upper bound on the number of Odec queries, then the single-instance configurations $(N, Q_d) = (1, 0)$ and $(N, Q_d) = (1, \infty)$ correspond to standard definitions of OT-IND-CPA and OT-IND-CCA security for DEMs.

Game $\text{MIOT-IND}_{A, N}^b$	Oracle $\text{Oenc}(j, m_0, m_1)$	Oracle $\text{Odec}(j, c)$
00 for all $j \in [1..N]$:	05 if $C_j \neq \emptyset$: return \perp	09 if $C_j = \emptyset$: return \perp
01 $K_j \xleftarrow{\$} \mathcal{K}$	06 $c \leftarrow \text{D.enc}(K_j, m_b)$	10 if $c \in C_j$: return \perp
02 $C_j \leftarrow \emptyset$	07 $C_j \leftarrow C_j \cup \{c\}$	11 $m \leftarrow \text{D.dec}(K_j, c)$
03 $b' \xleftarrow{\$} A$	08 return c	12 return m
04 return b'		

Fig. 3. DEM security games $\text{MIOT-IND}_{A, N}^b$, $b \in \{0, 1\}$, modeling multi-instance one-time indistinguishability for N instances.

Similarly to the cases of PKE and KEMs, our multi-instance notion for DEMs is equivalent to its single-instance counterpart, with a tightness loss of N . The proof can be found in the full version [14].

Lemma 3. *For any data-encapsulation mechanism DEM, any number of instances N , and any adversary A that poses at most Q_d -many Odec queries in total, there exists an adversary B such that $\text{Adv}_{\text{DEM}, A, N}^{\text{miot-ind}} \leq N \cdot \text{Adv}_{\text{DEM}, B, 1}^{\text{miot-ind}}$, where B poses at most one Oenc and Q_d -many Odec queries. Further, the running time of B is at most that of A plus the time needed to perform N -many D.enc operations and Q_d -many D.dec operations.*

3.2 Hybrid Encryption

The main application of KEMs and DEMs is the construction of public key encryption: To obtain a (hybrid) PKE scheme, a KEM is used to establish a

session key and a DEM is used with this key to protect the confidentiality of the message [11]. The details of this construction are in Fig. 4. It requires that the session key space of the KEM and the key space of the DEM coincide.

Proc P.gen	Proc P.enc(pk, m)	Proc P.dec(sk, ⟨c₁, c₂
00 (pk, sk) $\xleftarrow{\$}$ K.gen	02 (K, c ₁) $\xleftarrow{\$}$ K.enc(pk)	05 K ← K.dec(sk, c ₁)
01 return (pk, sk)	03 c ₂ ← D.enc(K, m)	06 if K = ⊥: return ⊥
	04 return ⟨c ₁ , c ₂ ⟩	07 m ← D.dec(K, c ₂)
		08 return m

Fig. 4. Hybrid construction of scheme PKE from schemes KEM and DEM. We write ⟨c₁, c₂⟩ for the encoding of two ciphertext components into one.

The central composability result for hybrid encryption [11] says that if the KEM and DEM components are strong enough then also their combination is secure, with tight reduction. In Theorem 1 we give a generalized version of this claim: it considers multiple users and challenges, and implies the result from [11] as a corollary. Note that also our generalization allows for a tight reduction. The proof can be found in the full version [14].

Theorem 1. *Let PKE be the hybrid public-key encryption scheme constructed from a key-encapsulation mechanism KEM and a data-encapsulation mechanism DEM as in Fig. 4. Then for any number of users n and any PKE adversary A that poses at most q_e -many Oenc and q_d -many Odec queries per user, there exist a KEM adversary B and a DEM adversary C such that*

$$\mathbf{Adv}_{\text{PKE}, A, n}^{\text{muc-ind}} \leq 2\mathbf{Adv}_{\text{KEM}, B, n}^{\text{muc-ind}} + \mathbf{Adv}_{\text{DEM}, C, nq_e}^{\text{miot-ind}}.$$

The running time of B is at most that of A plus the time required to run nq_e DEM encapsulations and nq_e DEM decapsulations. The running time of C is similar to the running time of A plus the time required to run nq_e KEM encapsulations, nq_e KEM decapsulations, and nq_e DEM decapsulations. B poses at most q_e -many Oenc and q_d -many Odec queries per user, and C poses at most nq_e -many Oenc and nq_d -many Odec queries in total.

Theorem 1 bounds the distinguishing advantage of adversaries against hybrid PKE conditioned on its KEM and DEM components being secure. Note that from this result it cannot be deduced that deploying an insecure DEM (potentially in combination with a secure KEM) necessarily leads to insecure PKE. We show in Theorem 2 that also the latter implication holds. To ease the analysis, instead of requiring MUC-IND-like properties of the KEM, we rather assume that it has uniformly distributed session keys. Formally this means that for all public keys pk the distribution of $[(K, c) \xleftarrow{\$} \text{K.enc}(pk); \text{output } K]$ is identical with the uniform distribution on key space \mathcal{K} . The proof can be found in the full version [14].

Theorem 2. *For a key-encapsulation mechanism KEM and a data-encapsulation mechanism DEM let PKE be the corresponding hybrid encryption scheme. If KEM has uniform keys in \mathcal{K} , any attack on DEM can be converted to an attack on PKE. More precisely, for any n, q_e and any DEM adversary A that poses in total at most nq_e -many Odec queries, there exists an adversary B such that*

$$\mathbf{Adv}_{\text{DEM}, A, nq_e}^{\text{miot-ind}} \leq \mathbf{Adv}_{\text{PKE}, B, n}^{\text{muc-ind}} + \frac{nq_e^2}{2|\mathcal{K}|}.$$

The running time of B is about that of A , and B poses at most q_e -many Oenc queries per user and Q_d -many Odec queries in total.

4 Deterministic DEMs and Their Multi-instance Security

We give two generic key-collision attacks on the multi-instance security of (deterministic) DEMs. They have different attack goals (indistinguishability vs. key recovery) and succeed with slightly different probabilities. More precisely, in both cases the leading term of the success probability comes from the birthday bound and evaluates to roughly $N^2/|\mathcal{K}|$, and is thus much larger than the $N/|\mathcal{K}|$ that intuition might expect. By Theorem 2 the attacks can directly be lifted to ones targeting the multi-user multi-challenge security of a corresponding hybrid encryption scheme, achieving the same advantage.

4.1 A Passive Multi-instance Distinguishing Attack on DEMs

We describe an attack against multi-instance indistinguishability that applies generically to all DEMs. Notably, the attack is fully passive, i.e., the adversary does not pose any query to its Odec oracle. As technical requirements we assume a finite message space and a number of instances such that the inequalities $N^2 \leq 2|\mathcal{K}|$ and $|\mathcal{M}| \geq 3|\mathcal{K}| + N - 1$ are fulfilled. We consider these conditions extremely mild, since in practice \mathcal{M} is very large and the value N can be chosen arbitrarily low by simply discarding some inputs.

For any value $N \in \mathbb{N}$ the details of our adversary $A = A_N$ are in Fig. 5a. It works as follows: It starts by picking uniformly at random messages $m_0, m_1^1, \dots, m_1^N \in \mathcal{M}$ such that m_1^1, \dots, m_1^N are pairwise distinct. (Note the corresponding requirement $N \leq |\mathcal{M}|$ follows from above condition.) The adversary then asks for encapsulations of these messages in a way such that it obtains either N encapsulations of m_0 (if executed in game MIOT-IND⁰), or one encapsulation of each message m_1^j (if executed in game MIOT-IND¹). If any two of the received ciphertexts collide, the adversary outputs 1; otherwise it outputs 0. The following theorem makes statements about advantage and running time of this adversary.

Theorem 3. For a finite message space \mathcal{M} , let DEM be a DEM with key space \mathcal{K} . Suppose that $N^2 \leq 2|\mathcal{K}|$ and $|\mathcal{M}| \geq 3|\mathcal{K}| + N - 1$. Then adversary A from Fig. 5a breaks the N -instance indistinguishability of DEM, achieving the advantage

$$\mathbf{Adv}_{\text{DEM}, A, N}^{\text{miot-ind}} \geq \frac{N(N-1)}{12|\mathcal{K}|}.$$

Its running time is $\mathcal{O}(N \log N)$, and it poses N -many Oenc and no Odec queries.

We remark that, more generally, the bound on $|\mathcal{M}|$ can be relaxed to $|\mathcal{M}| \geq 2|\mathcal{K}|(1 + \delta) + N - 1$ for some $\delta \geq 0$ to obtain $\mathbf{Adv}_{\text{DEM}, A, N}^{\text{miot-ind}} \geq \frac{\delta}{\delta+1} \cdot \frac{N(N-1)}{4|\mathcal{K}|}$.

```

Adversary AN
00  $m_0 \xleftarrow{\$} \mathcal{M}$ 
01 for all  $j \in [1..N]$ :
02  $m_1^j \xleftarrow{\$} \mathcal{M} \setminus \{m_1^1, \dots, m_1^{j-1}\}$ 
03  $c^j \leftarrow \text{Oenc}(j, m_0, m_1^j)$ 
04 return 1 iff  $\text{Coll}_2[c^1, \dots, c^N]$ 
    
```

(a) MIOT-IND adversary, Theorem 3.

```

Adversary AN
00 for all  $i \in [1..N]$ :
01  $K_i \xleftarrow{\$} \mathcal{K} \setminus \{K_1, \dots, K_{i-1}\}$ 
02  $c_i \leftarrow \text{D.enc}(K_i, m_0)$ 
03 for all  $j \in [1..N]$ :
04  $c'_j \leftarrow \text{Oenc}(j, m_0)$ 
05 if  $\exists(i, j) \in [1..N]^2$  s.t.  $c_i = c'_j$ :
06 return  $(K_i, j)$ 
07 return  $\perp$ 
    
```

(b) MIOT-KR adversary, Theorem 4.

Fig. 5. Adversaries against: (a) multi-instance indistinguishability and (b) multi-instance key recovery. Both ask for N encapsulations (resp. lines 03 and line 04) but do not use their decapsulation oracle.

Proof. The task of collecting N ciphertexts and checking for the occurrence of a collision can be completed in $\mathcal{O}(N \log N)$ operations. In the following we first assess the performance of the adversary when executed in games MIOT-IND⁰ and MIOT-IND¹; then we combine the results.

CASE MIOT-IND⁰. Adversary A receives N encapsulations of the same message m_0 , created with N independent keys K_1, \dots, K_N . If two of these keys collide then the corresponding (deterministic) encapsulations collide as well and A returns 1. Since $N(N-1) < N^2 \leq 2|\mathcal{K}|$ by the birthday bound we obtain

$$\Pr[\text{MIOT-IND}_{A, N}^0] \geq \frac{N(N-1)}{4|\mathcal{K}|}.$$

CASE MIOT-IND¹. Adversary A receives encapsulations c^1, \dots, c^N of uniformly distributed (but distinct) messages m_1^1, \dots, m_1^N . Denote with K_j the key used to compute c^j , let $\mathcal{M}_j := \mathcal{M} \setminus \{m_1^1, \dots, m_1^{j-1}\}$, and let further $\mathcal{C}_j := \text{D.enc}(K_j, \mathcal{M}_j)$ denote the image of \mathcal{M}_j under (injective) function $\text{D.enc}(K_j, \cdot)$. Observe this setup implies $|\mathcal{C}_j| = |\mathcal{M}_j|$ and $|\mathcal{C}_1| > \dots > |\mathcal{C}_N|$. It further follows that each ciphertext c^j is uniformly distributed in set \mathcal{C}_j .

We aim at establishing an upper-bound on the collision probability of ciphertexts c^1, \dots, c^N . The maximum collision probability is attained in the worst-case $\mathcal{C}_1 \supset \dots \supset \mathcal{C}_N$, in which it is bounded by the collision probability of choosing N values uniformly from a set of cardinality $|\mathcal{C}_N| = |\mathcal{M}| - N + 1$. Using again the birthday bound and $|\mathcal{M}| \geq 3|\mathcal{K}| + N - 1$ we obtain

$$\Pr[\text{MIOT-IND}_{\mathcal{A},N}^1] \leq \frac{1}{2} \cdot \frac{N(N-1)}{|\mathcal{M}| - N + 1} \leq \frac{N(N-1)}{6|\mathcal{K}|}.$$

Combining the two bounds yields the equation in our statement.

4.2 A Passive Multi-instance Key-Recovery Attack on DEMs

We give a generic attack on DEMs that aims at recovering keys rather than distinguishing encapsulations. Like in Sect. 4.1 the attack is passive. It is inspired by work of Zaverucha [22] and Chatterjee et al. [8]. However, our results are more general than theirs for not restricted to one specific DEM.

To formalize the notion of resilience against key recovery we correspondingly adapt the MIOT-IND game from Fig. 3 and obtain the MIOT-KR game specified in Fig. 6. The N -instance advantage of an adversary \mathcal{A} is then defined as $\text{Adv}_{\text{DEM},\mathcal{A},N}^{\text{miot-kr}} := \Pr[\text{MIOT-KR}_{\mathcal{A},N}]$. The following theorem shows that for virtually all practical DEMs (including those based on CBC mode, CTR mode, OCB, etc., and even one-time pad encryption) there exist adversaries achieving a considerable key recovery advantage, conditioned on the DEM key space being small enough. Concretely, the adversaries we propose encapsulate $2N$ times the same message (N times with random but known keys, and N times with random but unknown keys) and detect collisions of ciphertexts.⁵ As any ciphertext collision stems (in practice) from a collision of keys, this method allows for key recovery.⁶

Theorem 4. *Fix a DEM and denote its key space with \mathcal{K} and its message space with \mathcal{M} . Let $m_0 \in \mathcal{M}$ be any fixed message. Fixing $N \in \mathbb{N}$ as a parameter, consider the adversary $\mathcal{A} = \mathcal{A}_N$ specified in Fig. 5b. We then have*

$$\text{Adv}_{\text{DEM},\mathcal{A},N}^{\text{miot-kr}} \geq p(m_0) \cdot \min \left\{ \frac{1}{2}, \frac{N^2}{2|\mathcal{K}|} \right\},$$

where $p(m_0)$ denotes the collision probability

$$p(m_0) := \Pr_{K_1, K_2 \leftarrow \mathcal{K}} [K_1 = K_2 \mid \text{D.enc}(K_1, m_0) = \text{D.enc}(K_2, m_0)].$$

⁵ While our setup is formally meaningful, in practice it would correspond to N parties, for a huge number N , encapsulating the same message m_0 . This might feel rather unrealistic. However, we argue that a close variant of the attack might very well have the potential for practicality: All widely deployed DEMs are *online*, i.e., compute ciphertexts ‘left-to-right’. For such DEMs, for our attack to be successful, it suffices that the N parties encapsulate (different) messages that have a common prefix, for instance a standard protocol header.

⁶ The efficiency of this attack can likely be improved, on a heuristic basis, by deploying dedicated data structures like rainbow tables.

Game MIOT-KR _{A,N}	Oracle Oenc(j, m)	Oracle Odec(j, c)
00 for all $j \in [1..N]$:	05 if $C_j \neq \emptyset$: return \perp	09 if $C_j = \emptyset$: return \perp
01 $K_j \xleftarrow{\$} \mathcal{K}$	06 $c \leftarrow \text{D.enc}(K_j, m)$	10 if $c \in C_j$: return \perp
02 $C_j \leftarrow \emptyset$	07 $C_j \leftarrow C_j \cup \{c\}$	11 $m \leftarrow \text{D.dec}(K_j, c)$
03 $(K, i) \xleftarrow{\$} \mathcal{A}$	08 return c	12 return m
04 return 1 iff $K = K_i$		

Fig. 6. DEM security game MIOT-KR_{A,N} modeling resilience against key recovery, for N instances.

Its running time is $\mathcal{O}(N \log N)$, and it poses N -many Oenc and no Odec queries.

We further prove that in the case of DEMs based on one-time pad encryption we have $p(m_0) = 1$ for any m_0 . Further, in the case of CBC-based encapsulation there exists a message m_0 such that $p(m_0) = |\mathcal{B}| / (|\mathcal{B}| + |\mathcal{K}| - 1)$, where \mathcal{B} is the block space of the blockcipher and the latter is modeled as an ideal cipher.

Note that the performance of our attack crucially depends on the choice of message m_0 , and that there does not seem to be a general technique for identifying good candidates. In particular, (artificial) DEMs can be constructed where $p(m_0)$ is small for some m_0 but large for others, or where $p(m_0)$ is small even for very long messages m_0 . However, in many practical schemes the choice of m_0 is not determinant. After the proof we consider two concrete examples.

Proof. The running time of \mathcal{A} is upper bounded by the search for collisions in line 05, since all other operations require at most linear time in N . We estimate the time bound: The list c_1, \dots, c_N is sorted, requiring time $\mathcal{O}(N \log N)$. Searching an element in the ordered list requires $\mathcal{O}(\log N)$ time. Repeating for all N searches requires $\mathcal{O}(N \log N)$. Combining these observations yields our statement.

We claim that the probability that the adversary does not output \perp (in symbols, $\mathcal{A}_N \not\Rightarrow \perp$) is lower bounded by:

$$\Pr[\mathcal{A}_N \not\Rightarrow \perp] \geq 1 - \left(1 - \frac{N}{|\mathcal{K}|}\right)^N. \quad (1)$$

Since the DEM is deterministic, the probability to find any collision in line 05 is larger than the probability that any of the distinct N keys generated in lines 00–02 collides with one of the N keys $\tilde{K}_1, \dots, \tilde{K}_N$ used by the MIOT-KR game to encapsulate. We compute the latter probability. Let $K \in \{\tilde{K}_1, \dots, \tilde{K}_N\}$. We know that K is uniform in \mathcal{K} . Since K_1, \dots, K_N are distinct and independently chosen we can write: $\Pr[K \in \{K_1, \dots, K_N\}] = N/|\mathcal{K}|$. Moreover, since the keys $\tilde{K}_1, \dots, \tilde{K}_N$ are generated independently of each other, Eq. (1) follows.

Let now (i, j) be the indices for which the condition in line 05 is triggered, i.e., $c_i = c'_j$ and \mathcal{A}_N outputs K_i . We can write:

$$\begin{aligned} \text{Adv}_{\text{DEM,A},N}^{\text{miot-kr}} &= \Pr[\mathbf{A}_N \not\Rightarrow \perp] \cdot \Pr[K_i = \tilde{K}_j \mid \mathbf{A}_N \not\Rightarrow \perp] \\ &\geq \left(1 - \left(1 - \frac{N}{|\mathcal{K}|}\right)^N\right) \cdot p(m_0). \end{aligned}$$

Applying known inequalities to the previous formula we obtain:

$$\text{Adv}_{\text{DEM,A},N}^{\text{miot-kr}} \geq p(m_0) \cdot \left(1 - \left(1 - \frac{N}{|\mathcal{K}|}\right)^N\right) \geq p(m_0) \cdot \min\left\{\frac{1}{2}, \frac{N^2}{2|\mathcal{K}|}\right\}.$$

We compute $p(m_0)$ for two specific DEMs (one-time pad and CBC mode) and choices of m_0 . We formalize the argument for CBC by considering single-block messages. We note that one can apply the same argument to other modes of operation, e.g., CTR. For notational simplicity we omit the description of the probability space, that is, uniform choice of $K_1, K_2 \in \mathcal{K}$.

One-time pad. The one-time pad DEM encapsulation is given by combining a key $K \in \mathcal{K} = \{0, 1\}^k$ with a message $m \in \mathcal{M} = \{0, 1\}^k$ using the XOR operation. In this case, if two ciphertexts for the same message collide, the same key must have been used to encapsulate. Thus $p(m_0) = 1$ for all m_0 .

CBC with an ideal cipher. CBC-based DEM encapsulation consists of encrypting the message using a blockcipher in CBC mode with the zero initialization vector (IV). In the following analysis we assume an idealized blockcipher (ideal cipher model) represented by \mathbf{E} . Note that since the IV is zero, encapsulating a single-block message m_0 under the key K is equivalent to enciphering m_0 with \mathbf{E}_K . Let \mathcal{B} be the block space. First we observe that for any single-block message m_0 we have

$$\begin{aligned} &\Pr[\mathbf{E}_{K_1}(m_0) = \mathbf{E}_{K_2}(m_0)] \\ &= \Pr[K_1 = K_2] + \Pr[K_1 \neq K_2] \Pr[\mathbf{E}_{K_1}(m_0) = \mathbf{E}_{K_2}(m_0) \mid K_1 \neq K_2] \\ &= |\mathcal{K}|^{-1} + (1 - |\mathcal{K}|^{-1}) |\mathcal{B}|^{-1}. \end{aligned}$$

We then use the previous equality to compute $p(m_0)$ from its definition:

$$\begin{aligned} p(m_0) &= \frac{\Pr[K_1 = K_2]}{\Pr[\mathbf{E}_{K_1}(m_0) = \mathbf{E}_{K_2}(m_0)]} \\ &= \frac{|\mathcal{K}|^{-1}}{|\mathcal{K}|^{-1} + (1 - |\mathcal{K}|^{-1}) |\mathcal{B}|^{-1}} = \frac{|\mathcal{B}|}{|\mathcal{B}| + |\mathcal{K}| - 1}. \end{aligned}$$

As an example, if $|\mathcal{B}| \geq |\mathcal{K}|$ then $p(m_0) > 1/2$ for any single-block message m_0 .

5 Augmented Data Encapsulation

In the previous sections we showed that all deterministic DEMs, including those that are widely used in practice, might be less secure than expected in the face of

multi-instance attacks. We further showed that, in the setting of hybrid encryption, attacks on DEMs can be leveraged to attacks on the overall PKE. Given that the KEM+DEM paradigm is so important in practice, we next address the question of how this situation can be remedied. One option would of course be to increase the DEM key size (recall that good success probabilities in Theorems 3 and 4 are achieved only for not too large key spaces); however, increasing key sizes might not be a viable option in practical systems. (Potential reasons for this include that blockciphers like AES are slower with long keys than with short keys, and that ciphers like 3DES do not support key lengths that have a comfortable ‘multi-instance security margin’ in the first place.) A second option would be to augment the input given to the DEM encapsulation routine by an additional value. This idea was already considered in [22, p. 16] where, with the intuition of increasing the ‘entropy’ available to the DEM, it was proposed to use a KEM ciphertext as an initialization vector (IV) of a symmetric encryption mode. However, [22] does not contain any formalization or security analysis of this idea, and so it cannot be taken as granted that this strategy actually works. (And indeed, we show in Sect. 6.3 that deriving the starting value of blockcipher-based counter mode encryption from a KEM ciphertext is not ameliorating the situation for attacks based on indistinguishability.)

We formally explore the additional-input proposal for the DEM in this section. More precisely, we study two approaches of defining an *augmented data encapsulation mechanism* (ADEM), where we call the additional input the *tag*. The syntax is the same in both cases, but the security properties differ: either (a) the DEM encapsulator receives as the tag an auxiliary random (but public) string, or (b) the encapsulator receives as additional input a nonce (a ‘number used once’). In both cases the decapsulation oracle operates with respect to the tag also used for encapsulation. After formalizing this we prove the following results: First, if the tag space is large enough, ADEMs that expect a nonce can safely replace ADEMs that expect a uniform tag. Second, ADEMs that expect a uniform tag can be constructed from ADEMs that expect a nonce by applying a random oracle to the latter. Our third result is that the augmented variant of hybrid encryption remains (tightly) secure.

AUGMENTED DATA ENCAPSULATION. An augmented data encapsulation mechanism $\text{ADEM} = (\text{A.enc}, \text{A.dec})$ for a message space \mathcal{M} is a pair of deterministic algorithms associated with a finite key space \mathcal{K} , a tag space \mathcal{T} , and a ciphertext space \mathcal{C} . The encapsulation algorithm A.enc takes a key $K \in \mathcal{K}$, a tag $t \in \mathcal{T}$, and a message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$. The decapsulation algorithm A.dec takes a key $K \in \mathcal{K}$, a tag $t \in \mathcal{T}$, and a ciphertext $c \in \mathcal{C}$, and outputs either a message $m \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$ to indicate rejection. The correctness requirement is that for all $K \in \mathcal{K}$ and $t \in \mathcal{T}$ and $m \in \mathcal{M}$ we have $\text{A.dec}(K, t, \text{A.enc}(K, t, m)) = m$.

AUGMENTED DATA ENCAPSULATION WITH UNIFORM TAGS. The first security notion we formalize assumes that each encapsulation operation uses a fresh and uniformly picked tag (note this imposes the technical requirement that the tag space be finite). More precisely, while the tag may become public

after the encapsulation operation has completed, it may not be disclosed to the adversary before fixing the message to be encapsulated. We formalize this notion of uniform-tag multi-instance one-time indistinguishability for ADEMs via the games specified in Fig. 7. For a scheme ADEM, to any adversary A and any number of instances N we associate the distinguishing advantage $\text{Adv}_{\text{ADEM},A,N}^{\text{u-miot-ind}} := |\Pr[\text{U-MIOT-IND}_{A,N}^0] - \Pr[\text{U-MIOT-IND}_{A,N}^1]|$.

Game U-MIOT-IND $_{A,N}^b$	Oracle Oenc(j, m_0, m_1)	Oracle Odec(j, c)
00 for all $j \in [1 \dots N]$:	05 if $C_j \neq \emptyset$: return \perp	09 if $C_j = \emptyset$: return \perp
01 $(K_j, t_j) \xleftarrow{\$} \mathcal{K} \times \mathcal{T}$	06 $c \leftarrow \text{A.enc}(K_j, t_j, m_b)$	10 if $c \in C_j$: return \perp
02 $C_j \leftarrow \emptyset$	07 $C_j \leftarrow C_j \cup \{c\}$	11 $m \leftarrow \text{A.dec}(K_j, t_j, c)$
03 $b' \xleftarrow{\$} A$	08 return (t_j, c)	12 return m
04 return b'		

Fig. 7. ADEM security games U-MIOT-IND $_{A,N}^b$, $b \in \{0, 1\}$, for N instances. The tags in line 11 are the same as the ones in line 06.

AUGMENTED DATA ENCAPSULATION WITH NONCES. Our second security notion for ADEMs requires the tag provided to each encapsulation operation to be unique (across all instances). The tag can be generated using any possible method (e.g., using some global type of counter). We formalize the corresponding security notion of nonce-based multi-instance one-time indistinguishability for ADEMs via the games specified in Fig. 8. For a scheme ADEM, to any adversary A and any number of instances N we associate the distinguishing advantage $\text{Adv}_{\text{ADEM},A,N}^{\text{n-miot-ind}} := |\Pr[\text{N-MIOT-IND}_{A,N}^0] - \Pr[\text{N-MIOT-IND}_{A,N}^1]|$.

Game N-MIOT-IND $_{A,N}^b$	Oracle Oenc(j, t, m_0, m_1)	Oracle Odec(j, c)
00 $T \leftarrow \emptyset$	06 if $C_j \neq \emptyset$: return \perp	12 if $C_j = \emptyset$: return \perp
01 for all $j \in [1 \dots N]$:	07 if $t \in T$: return \perp	13 if $c \in C_j$: return \perp
02 $K_j \xleftarrow{\$} \mathcal{K}$	08 $T \leftarrow T \cup \{t\}$; $t_j \leftarrow t$	14 $m \leftarrow \text{A.dec}(K_j, t_j, c)$
03 $C_j \leftarrow \emptyset$	09 $c \leftarrow \text{A.enc}(K_j, t_j, m_b)$	15 return m
04 $b' \xleftarrow{\$} A$	10 $C_j \leftarrow C_j \cup \{c\}$	
05 return b'	11 return c	

Fig. 8. ADEM security games N-MIOT-IND $_{A,N}^b$, $b \in \{0, 1\}$, for N instances. The tags in line 14 are the same as the ones in line 09.

5.1 Relations Between ADEMs with Uniform and Nonce Tags

The two types of ADEMs we consider here can be constructed from each other. More concretely, the following lemma shows that if the tag space is large enough, ADEMs that expect a nonce can safely replace ADEMs that expect a uniform tag. The proof can be found in the full version [14].

Lemma 4. *Let ADEM be an augmented data encapsulation mechanism. If the cardinality of its tag space \mathcal{T} is large enough and ADEM is secure with non-repeating tags, then it is also secure with random tags. More precisely, for any number of instances N and any adversary \mathbf{A} there exist an adversary \mathbf{B} that makes the same amount of queries such that $\text{Adv}_{\text{ADEM},\mathbf{A},N}^{\text{u-miot-ind}} \leq \text{Adv}_{\text{ADEM},\mathbf{B},N}^{\text{n-miot-ind}} + N^2/(2|\mathcal{T}|)$. The running time of the two adversaries is similar.*

The following simple lemma shows that ADEMs that expect a nonce can be constructed from ADEMs that expect a uniform tag by using each nonce to obtain a uniform, independent value from a random oracle. The proof is immediate since all queries to the random oracle have different input, thus the corresponding output is uniformly random and independently generated.

Lemma 5. *Let $\text{ADEM} = (\text{A.enc}, \text{A.dec})$ be an augmented data encapsulation mechanism with tag space \mathcal{T} . Let $H: \mathcal{T}' \rightarrow \mathcal{T}$ denote a hash function, where \mathcal{T}' is another tag space. Define $\text{ADEM}' = (\text{A.enc}', \text{A.dec}')$ such that $\text{A.enc}'(K, t, m) := \text{A.enc}(K, H(t), m)$ and $\text{A.dec}'(K, t, c) := \text{A.dec}(K, H(t), c)$. Then if H is modeled as a random oracle and if ADEM is secure with random tags in \mathcal{T} , then ADEM' is secure with non-repeating tags in \mathcal{T}' . Formally, for any number of instances N and any adversary \mathbf{A} there exists an adversary \mathbf{B} with $\text{Adv}_{\text{ADEM},\mathbf{A},N}^{\text{u-miot-ind}} = \text{Adv}_{\text{ADEM}',\mathbf{B},N}^{\text{n-miot-ind}}$.*

5.2 Augmented Hybrid Encryption

A KEM and an ADEM can be combined to obtain a PKE scheme: the KEM establishes a session key and a first ciphertext component, and the ADEM is used on input the session key and the first ciphertext component (as tag) to protect the confidentiality of the message, creating a second ciphertext component. Figure 9 details this *augmented hybrid encryption*. It requires that the session key space of the KEM and the key space of the ADEM coincide. Further, the ciphertext space of the KEM needs to be a subset of the tag space of the ADEM.

Proc P.gen	Proc P.enc(pk, m)	Proc P.dec($sk, \langle c_1, c_2 \rangle$)
00 $(pk, sk) \xleftarrow{\$} \text{K.gen}$	02 $(K, c_1) \xleftarrow{\$} \text{K.enc}(pk)$	05 $K \leftarrow \text{K.dec}(sk, c_1)$
01 return (pk, sk)	03 $c_2 \leftarrow \text{A.enc}(K, c_1, m)$	06 if $K = \perp$: return \perp
	04 return $\langle c_1, c_2 \rangle$	07 $m \leftarrow \text{A.dec}(K, c_1, c_2)$
		08 return m

Fig. 9. Augmented hybrid construction of scheme PKE from schemes KEM and ADEM. We write $\langle c_1, c_2 \rangle$ for the encoding of two ciphertext components into one.

The claim is that augmented hybrid encryption is more robust against attacks involving multiple users and challenges than standard hybrid encryption (see Fig. 4). The security condition posed on the ADEM requires that it be secure when operated with nonces, and the security property posed on the KEM

requires that it be both indistinguishable and have non-repeating ciphertexts (i.e., invoking the encapsulation twice on any public keys does virtually never result in colliding ciphertexts). Technically, the latter property is implied by indistinguishability. However, to obtain better bounds, we formalize it as a statistical condition: To any scheme KEM we assign the maximum ciphertext-collision probability

$$p := \max_{pk_1, pk_2} \Pr[(K_1, c_1) \xleftarrow{\$} \text{K.enc}(pk_1); (K_2, c_2) \xleftarrow{\$} \text{K.enc}(pk_2) : c_1 = c_2],$$

where the maximum is over all pairs pk_1, pk_2 of (potentially coinciding) public keys. Note that practical KEMs (ElGamal, RSA-based, Cramer–Shoup, ...) have much larger ciphertexts than session keys⁷, so that the ciphertext-collision probability will always be negligible in practice. We proceed with a security claim for augmented hybrid encryption. The proof can be found in the full version [14].

Lemma 6. *Let PKE be the hybrid public-key encryption scheme constructed from a key-encapsulation mechanism KEM and an augmented data-encapsulation mechanism ADEM as in Fig. 9. Let p be the maximum ciphertext-collision probability of KEM over all possible public keys. Then for any n and any PKE adversary A that poses at most q_e -many Oenc and q_d -many Odec queries per user, there exist a KEM adversary B and an ADEM adversary C such that*

$$\text{Adv}_{\text{PKE}, A, n}^{\text{muc-ind}} \leq 2\text{Adv}_{\text{KEM}, B, n}^{\text{muc-ind}} + \text{Adv}_{\text{ADEM}, C, N}^{\text{n-miot-ind}} + 2\binom{N}{2}p,$$

where $N = nq_e$. The running time of B is at most that of A plus the time required to run nq_e ADEM encapsulations and nq_e ADEM decapsulations. The running time of C is similar to that of A plus the time required to run nq_e KEM encapsulations, nq_e KEM decapsulations, and nq_e ADEM decapsulations. B poses at most q_e -many Oenc and q_d -many Odec queries per user, and C poses at most nq_e -many Oenc and nq_d -many Odec queries in total.

6 Constructions of Augmented Data Encapsulation

We construct two augmented data-encapsulation mechanisms and analyze their security. The schemes are based on operating a function in counter mode. If the function is instantiated with an ideal random function then the ADEMs are secure beyond the birthday bound. (We also show that if the function is instead instantiated with an idealized blockcipher, i.e., a random permutation, the schemes' security may degrade.) Practical candidates for instantiating the ideal random function are for instance the compression functions of standardized Merkle–Damgård hash functions, e.g., of SHA2.^{8,9} Another possibility is deriving the random function from an ideal cipher as in [21].

⁷ This is no coincidence but caused by generic attacks against cyclic groups, RSA, etc.

⁸ These compression functions are regularly modeled as having random behavior [2, 13].

⁹ The idea to construct a DEM from a hash function's compression function already appeared in the OMD schemes from [9].

6.1 Counter-Mode Encryption

Many practical DEMs are based on operating a blockcipher E in counter mode (CTR). Here, in brief, the encapsulation key is used as the blockcipher key, a sequence of message-independent input blocks is enciphered under that key, and the output blocks are XOR-ed into the message. More concretely, if under some key K a message m shall be encapsulated that, without requiring padding, evenly splits into blocks $v_1 \parallel \dots \parallel v_l$, then the DEM ciphertext is the concatenation $w_1 \parallel \dots \parallel w_l$ where $w_i = v_i \oplus E_K(i)$.

In the context of this paper, three properties of this construction are worth pointing out: (a) the ‘counting’ component of CTR mode serves a single purpose: preventing that two inputs to the blockcipher coincide; (b) any ‘starting value’ for the counter can be used; (c) security analyses of CTR mode typically model E as a pseudorandom function (as opposed to a pseudorandom permutation)¹⁰.

In Fig. 10 we detail three ways of turning the principles of CTR mode into a DEM encapsulation routine. In all cases the underlying primitive is, syntactically, a function $F: \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{D}$ that takes a key $K \in \mathcal{K}$ and maps some finite input space \mathcal{B} into some finite group (\mathcal{D}, \oplus) . (Intuitively, \mathcal{B} serves as a space of input blocks derived from a counter, and \mathcal{D} as a space of pads that can be XORed into message blocks; note that if F is instantiated with a blockcipher we have $\mathcal{B} = \mathcal{D}$, but we explicitly allow other instantiations.) The most basic encapsulation routine based on CTR mode that we consider, and the one closest to our sketch above, is CTR0enc. Note that this DEM further assumes a bijection $\llbracket \cdot \rrbracket_L: \mathbb{Z}/L\mathbb{Z} \rightarrow \mathcal{L}$ with $\mathcal{L} = \mathcal{B}$. (Intuitively, this bijection turns a counter that is cyclic with period length L into input blocks for F ; see Sect. 2 for notation.) We finally point out that all three variants of CTR mode that we formalize exclusively work with fixed-length multi-block messages (i.e., $\mathcal{M} = \mathcal{D}^l$). This choice, that we made for simplicity of exposition, is not really a restriction as ‘any-length’ CTR mode encryption can be simulated from ‘block-wise’ CTR mode encryption.

Proc CTR0enc(K, m)	Proc CTR+enc(K, t, m)	Proc CTR enc(K, t, m)
00 $(v_1, \dots, v_l) \leftarrow m$	05 $(v_1, \dots, v_l) \leftarrow m$	10 $(v_1, \dots, v_l) \leftarrow m$
01 for all $i \in [1..l]$:	06 for all $i \in [1..l]$:	11 for all $i \in [1..l]$:
02 $w_i \leftarrow v_i \oplus F(K, \llbracket i \rrbracket_L)$	07 $w_i \leftarrow v_i \oplus F(K, \llbracket t + i \rrbracket_L)$	12 $w_i \leftarrow v_i \oplus F(K, t \parallel \llbracket i \rrbracket_L)$
03 $c \leftarrow (w_1, \dots, w_l)$	08 $c \leftarrow (w_1, \dots, w_l)$	13 $c \leftarrow (w_1, \dots, w_l)$
04 return c	09 return c	14 return c

Fig. 10. Encapsulation algorithms of the CTR0 DEM, the CTR+ ADEM, and the CTR|| ADEM, for multi-block messages. In CTR0enc and CTR+enc we assume $\llbracket \cdot \rrbracket_L: \mathbb{Z}/L\mathbb{Z} \rightarrow \mathcal{L}$ with $\mathcal{L} = \mathcal{B}$, and in CTR||enc we assume $\llbracket \cdot \rrbracket_L: \mathbb{Z}/L\mathbb{Z} \rightarrow \mathcal{L}$ and \mathcal{T} such that $\mathcal{B} = \mathcal{T} \times \mathcal{L}$. The corresponding decapsulation routines is immediate.

¹⁰ Technically, the PRP/PRF switching lemma [5] measures the price one has to pay for pursuing this modeling approach.

The two remaining procedures in Fig. 10 are ADEM encapsulation routines. The first one, CTR+enc, is the natural variant of CTR0enc where the tag space is $\mathcal{T} = [1..L]$ and the tag specifies the starting value of the counter. The second, CTR||enc, concatenates tag and counter. Here, the tag space \mathcal{T} and parameter space \mathcal{L} have to be arranged such that $\mathcal{B} = \mathcal{T} \times \mathcal{L}$.

We analyze the security of CTR+ and CTR|| in the upcoming sections. Scheme CTR0 is not an ADEM and falls prey to our earlier attacks.

6.2 Security of Function-Based Counter Mode

We establish upper bounds on the advantage of U-MIOT-IND adversaries against the CTR+ and CTR|| ADEMs.

Counter Mode with Tag-Controlled Starting Value. We limit the maximum amount of blocks in an encapsulation query to a fixed value ℓ . Prerequisites to our statement on CTR+ are two conditions on the number of instances relative to \mathcal{K} and $\mathcal{T} = [1..L]$. The bound is namely $N \leq \min \{ |\mathcal{K}|^{1/2}, (|\mathcal{T}|/(2\ell))^{1/(1+\delta)} \}$, for some arbitrary constant δ such that $1/N \leq \delta \leq 1$. Despite this restriction we consider our statement to be reflecting real-world applications: As an extreme example we see that the values $|\mathcal{K}| = |\mathcal{T}| = 2^{128}$, $N = 2^{56}$, $\ell = 2^{56}$, $q = 2^{64}$ and $\delta = 2/7$ fit above condition, yielding a maximum advantage of around 2^{-61} .

Theorem 5. *Suppose $N \leq \min \{ |\mathcal{K}|^{1/2}, (|\mathcal{T}|/(2\ell))^{1/(1+\delta)} \}$, for some $1/N \leq \delta \leq 1$, and suppose that F is modeled as a random oracle (using oracle F). Then for any adversary A against N -instance uniform-tag indistinguishability of CTR+ that poses at most q queries to F , no decapsulation queries, and encapsulates messages of length at most ℓ blocks we have:*

$$\text{Adv}_{\text{CTR}^+, A, N}^{\text{u-miot-ind}} \leq \frac{1}{3} \frac{N}{|\mathcal{K}|} + \frac{4\ell - 2}{|\mathcal{T}|} + \frac{2q}{|\mathcal{K}|} \left(1 + \frac{1}{\delta} \right).$$

The core of the proof exploits that the outputs of (random oracle) F that are used to encapsulate are uniformly distributed in \mathcal{D} and independent of each other. This requires forcing the inputs to be distinct in \mathcal{L} . We give further insight on some non-standard techniques the we use in the analysis in the proof.

Proof (of Theorem 5). The definition of the games $G_{A,N}^{0,b}$, $G_{A,N}^{1,b}$, $G_{A,N}^{2,b}$ and $G_{A,N}^{3,b}$ are found in Fig. 11. Except for some bookkeeping, game $G_{A,N}^{0,b}$ is equivalent to game $\text{U-MIOT-IND}_{A,N}^b$, where $b \in \{0, 1\}$. For $j \in [1..N]$ we define $T_j = \llbracket t_j \rightarrow \ell \rrbracket_L$.

Game G^1 . In game $G_{A,N}^{1,b}$ we implicitly generate pairs of colliding keys. We loop over all pairs (j_1, j_2) such that $1 \leq j_1 < j_2 \leq N$. If both indices were not previously paired ($\text{matched}[j_1] = \text{matched}[j_2] = \text{false}$) and the corresponding keys collide ($K_{j_1} = K_{j_2}$) then the two indices are marked as paired. Moreover,

if the corresponding tag ranges collide ($T_{j_1} \cap T_{j_2} \neq \emptyset$) the flag bad_1 in line 10 is raised and the game aborts. We claim that

$$|\Pr[\mathsf{G}_{\mathcal{A},N}^{0,b}] - \Pr[\mathsf{G}_{\mathcal{A},N}^{1,b}]| \leq \Pr[\text{bad}_1] \leq \frac{2\ell - 1}{|\mathcal{T}|}. \quad (2)$$

To prove (2), we want to compute the probability $\Pr[\text{bad}_1]$. Let m_{pairs} be the number of colliding key pairs in game $\mathsf{G}_{\mathcal{A},N}^{1,b}$, i.e., $2m_{\text{pairs}}$ entries of flag `matched` are set to 1 at the end of the game. Then, for every $0 \leq i \leq \lfloor N/2 \rfloor$, $\Pr[\text{bad}_1 \mid m_{\text{pairs}} = i] \leq (2\ell - 1)i/|\mathcal{T}|$. This follows from the independent choices of the values K_j, t_j for each instance $j \in [1..N]$, and because for each pair of indices $j_1, j_2 \in [1..N], j_1 \neq j_2$, and for any choice of t_{j_1} there are exactly $2\ell - 1$ possible values of t_{j_2} such that $T_{j_1} \cap T_{j_2} \neq \emptyset$. The sets $\{m_{\text{pairs}} = i\}, i \in 0, \dots, \lfloor N/2 \rfloor$, partition the probability space, thus:

$$\begin{aligned} \Pr[\text{bad}_1] &= \sum_{i=0}^{\lfloor N/2 \rfloor} \Pr[\text{bad}_1 \mid m_{\text{pairs}} = i] \Pr[m_{\text{pairs}} = i] \\ &\leq \frac{2\ell - 1}{|\mathcal{T}|} \sum_{i=0}^{\lfloor N/2 \rfloor} i \Pr[m_{\text{pairs}} = i] = \frac{2\ell - 1}{|\mathcal{T}|} \sum_{i=1}^{\lfloor N/2 \rfloor} \Pr[m_{\text{pairs}} \geq i]. \end{aligned} \quad (3)$$

The last equality follows since the expected value of any random variable m with values in \mathbb{N} can be written as $\sum_{i=0}^{\infty} i \Pr[m = i] = \sum_{i=1}^{\infty} \Pr[m \geq i]$. We show by induction that the terms of the sum are:

$$p_i := \Pr[m_{\text{pairs}} \geq i] \leq \left(\frac{N^2}{2|\mathcal{K}|} \right)^i. \quad (4)$$

To prove (4), we consider a slightly different event. We say that *key K_i is bad* if $K_j = K_i$ for some $1 \leq i < j$. Let m_{badkeys} be the random variable counting the number of bad keys. Since every colliding key pair implies at least one bad key, then it can be shown that $\Pr[m_{\text{pairs}} \geq i] \leq \Pr[m_{\text{badkeys}} \geq i] \leq (N^2/2|\mathcal{K}|)^i$. For more details we refer to the full version [14].

Finally we prove (2) by combining (3) and (4), and by observing that from our hypothesis $N^2/|\mathcal{K}| \leq 1$:

$$\Pr[\text{bad}_1] \leq \frac{2\ell - 1}{|\mathcal{T}|} \sum_{i=1}^{\lfloor N/2 \rfloor} \left(\frac{N^2}{2|\mathcal{K}|} \right)^i \leq \frac{2\ell - 1}{|\mathcal{T}|} \sum_{i=1}^{\infty} \frac{1}{2^i} = \frac{2\ell - 1}{|\mathcal{T}|}. \quad (5)$$

Game G^2 . Game $\mathsf{G}_{\mathcal{A},N}^{2,b}$ is equivalent to $\mathsf{G}_{\mathcal{A},N}^{1,b}$, with the exception that it raises flag bad_2 in line 12 and aborts if any three keys collide. By the generalized birthday bound, and since $N^2/|\mathcal{K}| \leq 1$, we obtain

$$|\Pr[\mathsf{G}_{\mathcal{A},N}^{1,b}] - \Pr[\mathsf{G}_{\mathcal{A},N}^{2,b}]| \leq \Pr[\text{bad}_2] \leq \frac{1}{6} \frac{N^3}{|\mathcal{K}|^2} \leq \frac{1}{6} \frac{N}{|\mathcal{K}|}. \quad (6)$$

Game G^3 . Game $G_{A,N}^{3,b}$ is equivalent to $G_{A,N}^{2,b}$, with the exception that the game raises flag bad_3 in line 23 and aborts if A makes a query (K, v) to F for which there exists an index $j \in [1..N]$ such that $K = K_j$ and $v \in T_j$. In the following we fix m_{inters} to be the random variable that counts the maximum number of sets T_1, \dots, T_N whose intersection is non-empty.

Fix a query (K, v) to F . For each $i \in [1..N]$ we have $\Pr[\exists j \in [1..N] : v \in T_j \wedge K = K_j \mid m_{\text{inters}} = i] \leq i/|\mathcal{K}|$, because in the worst case v belongs to exactly m_{inters} of the sets T_1, \dots, T_N . This bound yields

$$\begin{aligned} & \Pr[\exists j \in [1..N] : v \in T_j \wedge K = K_j] \\ &= \sum_{i=1}^N \Pr[\exists j \in [1..N] : v \in T_j \wedge K = K_j \mid m_{\text{inters}} = i] \cdot \Pr[m_{\text{inters}} = i] \\ &\leq \sum_{i=1}^N \frac{i}{|\mathcal{K}|} \cdot \Pr[m_{\text{inters}} = i] = \frac{1}{|\mathcal{K}|} \cdot \sum_{i=1}^N \Pr[m_{\text{inters}} \geq i]. \end{aligned} \quad (7)$$

Some probabilistic considerations allow us to write $\Pr[m_{\text{inters}} \geq i + 1] \leq N^{i+1} \ell^i / |\mathcal{T}|^i$ (details in the full version [14]). For all $i \geq 1/\delta$ we can write $\frac{N^{i+1} \ell^i}{|\mathcal{T}|^i} \leq \left(\frac{N^{1+\delta} \ell}{|\mathcal{T}|}\right)^i \leq \frac{1}{2^i}$. Thus we can split the sum (7) into

$$\begin{aligned} \frac{1}{|\mathcal{K}|} \cdot \sum_{i=1}^N \Pr[m_{\text{inters}} \geq i] &\leq \frac{1}{|\mathcal{K}|} \left(\sum_{i=1}^{\lfloor 1/\delta \rfloor} \Pr[m_{\text{inters}} \geq i] + \sum_{i=\lfloor 1/\delta \rfloor + 1}^{\infty} \frac{1}{2^{i-1}} \right) \\ &\leq \frac{1}{|\mathcal{K}|} \left(\frac{1}{\delta} + 1 \right). \end{aligned}$$

Since m_{inters} is constant for all q queries to F , a union bound gives us

$$\left| \Pr[G_{A,N}^{2,b}] - \Pr[G_{A,N}^{3,b}] \right| \leq \Pr[\text{bad}_3] \leq \frac{q}{|\mathcal{K}|} \left(\frac{1}{\delta} + 1 \right). \quad (8)$$

The theorem follows by combining the bounds in (2), (6), (8) for both $b = 0$ and $b = 1$ and the fact that game $G_{A,N}^{3,b}$ is independent of the bit b .

Counter Mode with Tag Prefix. We have the following security statement on $\text{CTR}\|$. Note it is slightly better than the one for $\text{CTR}+$.

Theorem 6. *Suppose $N \leq \min \{ |\mathcal{K}|^{1/2}, (|\mathcal{T}|/2)^{1/(1+\delta)} \}$, for some $1/N \leq \delta \leq 1$, and suppose that F is modeled as a random oracle (using oracle F). Then for any adversary A against N -instance uniform-tag indistinguishability of $\text{CTR}\|$ that poses at most q queries to F and no decapsulation queries we have:*

$$\text{Adv}_{\text{CTR}\|, A, N}^{\text{u-miot-ind}} \leq \frac{1}{3} \frac{N}{|\mathcal{K}|} + \frac{1}{|\mathcal{T}|} + \frac{2q}{|\mathcal{K}|} \left(1 + \frac{1}{\delta} \right).$$

<p>Game $\mathsf{G}_{A,N}^{0,b}$ – Game $\mathsf{G}_{A,N}^{3,b}$</p> <pre> 00 for all $j \in [1..N]$: 01 $\text{matched}[j] \leftarrow \text{false}$ 02 $(K_j, t_j) \xleftarrow{\\$} \mathcal{K} \times \mathcal{T}$ 03 for all $j_1 \in [1..N]$: 04 for all $j_2 \in [j_1 + 1..N]$: 05 if $(K_{j_1} = K_{j_2})$: 06 if $\neg \text{matched}[j_1] \wedge \neg \text{matched}[j_2]$: 07 $\text{matched}[j_1] \leftarrow \text{true}$ 08 $\text{matched}[j_2] \leftarrow \text{true}$ 09 if $\llbracket t_{j_1} \rightarrow \ell \rrbracket_L \cap \llbracket t_{j_2} \rightarrow \ell \rrbracket_L \neq \emptyset$: 10 $\text{bad}_1 \leftarrow \text{true}; \text{abort}$ 11 if $\text{Coll}_3[K_1, \dots, K_N]$: 12 $\text{bad}_2 \leftarrow \text{true}; \text{abort}$ 13 $b' \xleftarrow{\\$} \mathcal{A}$ 14 return b' </pre>	<p>Oracle $\text{Oenc}(j, m_0, m_1)$</p> <pre> 15 $(v_1, \dots, v_l) \leftarrow m_b$ 16 for all $i \in [1..l]$: 17 $w_i \leftarrow v_i \oplus \text{F}(K_j, \llbracket t_j + i \rrbracket_L)$ 18 $c \leftarrow (w_1, \dots, w_l)$ 19 return (t_j, c) </pre> <p>Oracle $\text{F}(K, v)$</p> <pre> 20 for all $j \in [1..N]$: 21 if $(K = K_j) \wedge (v \in \llbracket t_j \rightarrow \ell \rrbracket_L)$: 22 if $\text{T}[K, v] \neq \perp$: 23 $\text{bad}_3 \leftarrow \text{true}; \text{abort}$ 24 if $\text{T}[K, v] = \perp$: 25 $\text{T}[K, v] \xleftarrow{\\$} \mathcal{D}$ 26 return $\text{T}[K, v]$ </pre>
---	---

Fig. 11. The security game $\mathsf{G}_{A,N}^{0,b}$ for CTR+ in the random oracle model, and games $\mathsf{G}_{A,N}^{1,b}$, $\mathsf{G}_{A,N}^{2,b}$, and $\mathsf{G}_{A,N}^{3,b}$. Adversary \mathcal{A} can query the oracle Oenc at most once for the same index j .

Proof. We refer to Fig. 12 for the definition of the games $\mathsf{G}_{A,N}^{0,b}$, $\mathsf{G}_{A,N}^{1,b}$, $\mathsf{G}_{A,N}^{2,b}$ and $\mathsf{G}_{A,N}^{3,b}$. Except for some bookkeeping, game $\mathsf{G}_{A,N}^{0,b}$ is equivalent to the security game $\text{U-MIOT-IND}_{A,N}^b$, with $b \in \{0, 1\}$.

Game G^1 . Game $\mathsf{G}_{A,N}^{1,b}$ is equivalent to $\mathsf{G}_{A,N}^{0,b}$, except when any three keys collide. By the generalized birthday bound, and since $N^2/|\mathcal{K}| \leq 1$, we obtain

$$|\Pr[\mathsf{G}_{A,N}^{0,b}] - \Pr[\mathsf{G}_{A,N}^{1,b}]| \leq \Pr[\text{bad}_1] \leq \frac{1}{6} \frac{N^3}{|\mathcal{K}|^2} \leq \frac{1}{6} \frac{N}{|\mathcal{K}|}. \quad (9)$$

Game G^2 . In game $\mathsf{G}_{A,N}^{2,b}$ we abort when two events occur simultaneously: a key 2-collision and collision of the corresponding tags. The probability to abort is by the generalized birthday bound, the independence of the two events, and the condition $N^2/|\mathcal{K}| \leq 1$:

$$|\Pr[\mathsf{G}_{A,N}^{1,b}] - \Pr[\mathsf{G}_{A,N}^{2,b}]| \leq \Pr[\text{bad}_2] \leq \frac{N^2}{2|\mathcal{K}|} \frac{1}{|\mathcal{T}|} \leq \frac{1}{2} \frac{1}{|\mathcal{T}|}. \quad (10)$$

Game G^3 . Game $\mathsf{G}_{A,N}^{3,b}$ is equivalent to $\mathsf{G}_{A,N}^{2,b}$, with the exception that the game raises flag bad_3 in line 16 if some specific condition is met. To get an upper bound on the probability to distinguish $\mathsf{G}_{A,N}^{2,b}$ and $\mathsf{G}_{A,N}^{3,b}$ we compute the probability that the adversary explicitly queries F for an input $(K, v \parallel \llbracket i \rrbracket_L)$ such that for some $j \in [1..N]$, $K = K_j$ and $v = t_j$. This leads to the equation:

$$|\Pr[\mathsf{G}_{A,N}^{2,b}] - \Pr[\mathsf{G}_{A,N}^{3,b}]| \leq \Pr[\text{bad}_3] \leq \frac{q}{|\mathcal{K}|} \left(\frac{1}{\delta} + 1 \right). \quad (11)$$

Fix a query $(K, v \parallel [i]_L)$ to F . Since the adversary knows all possible values of v used by Oenc after each call, the adversary must only guess the key. Assume that there are at most m_{coll} keys that use the same tag value v . Then the probability that $\text{flag}_{\text{bad}_3}$ is triggered during this query is in the worst case $m_{\text{coll}}/|\mathcal{T}|$. We compute the probability of this event as follows.

$$\begin{aligned} & \Pr[\exists j \in [1..N] : v = t_j \wedge K = K_j] \\ &= \sum_{i=1}^N \Pr[\exists j \in [1..N] : v = t_j \wedge K = K_j \mid m_{\text{coll}} = i] \cdot \Pr[m_{\text{coll}} = i] \\ &\leq \sum_{i=1}^N \frac{i}{|\mathcal{K}|} \cdot \Pr[m_{\text{coll}} = i] = \frac{1}{|\mathcal{K}|} \cdot \sum_{i=1}^N \Pr[m_{\text{coll}} \geq i]. \end{aligned} \quad (12)$$

The last equality follows since the expected value of any random variable m with values in \mathbb{N} can be written as $\sum_{i=0}^{\infty} i \Pr[m = i] = \sum_{i=1}^{\infty} \Pr[m \geq i]$. Now we estimate the probability $\Pr[m_{\text{coll}} \leq i]$. Assume that $i \geq 1/\delta$. Then from the generalized birthday bound and the condition $N \leq (|\mathcal{T}|/2)^{1/(1+\delta)}$ we can write:

$$\Pr[m_{\text{coll}} \geq i + 1] \leq \frac{N^{i+1}}{(i+1)! |\mathcal{T}|^i} \leq \left(\frac{N^{1+\delta}}{|\mathcal{T}|} \right)^i \leq \frac{1}{2^i}.$$

Considering this observation we split the sum in Eq. (12) into

$$\begin{aligned} \frac{1}{|\mathcal{K}|} \cdot \sum_{i=1}^N \Pr[m_{\text{coll}} \geq i] &\leq \frac{1}{|\mathcal{K}|} \left(\sum_{i=1}^{\lfloor 1/\delta \rfloor} \Pr[m_{\text{coll}} \geq i] + \sum_{i=\lfloor 1/\delta \rfloor + 1}^{\infty} \frac{1}{2^{i-1}} \right) \\ &\leq \frac{1}{|\mathcal{K}|} \left(\frac{1}{\delta} + 1 \right). \end{aligned}$$

Since m_{coll} is constant for all queries to F , a union bound yields our claim:

$$\Pr[\text{bad}_3] \leq \frac{q}{|\mathcal{K}|} \left(\frac{1}{\delta} + 1 \right).$$

The theorem follows by combining the bounds in (9), (10), (11) for both $b = 0$ and $b = 1$ and the fact that game $\mathbf{G}_{\mathbf{A}, N}^{3, b}$ is independent of b .

6.3 On the Security of Permutation-Based Counter Mode

In above Theorem 5 we assessed the security of the CTR+ ADEM, defined with respect to a function $F: \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{D}$. The analysis modeled F as an ideal random function and showed that using sets \mathcal{K} and \mathcal{B} of moderate size (e.g., of cardinality 2^{128}) is sufficient to let CTR+ achieve security. We next show that if F is instead instantiated with a blockcipher and modeled as an ideal family of

Game $G_{A,N}^{0,b}$ – Game $G_{A,N}^{3,b}$ 00 for all $j \in [1..N]$: 01 $(K_j, t_j) \xleftarrow{\$} \mathcal{K} \times \mathcal{T}$ 02 if $\text{Coll}_3[K_1, \dots, K_N]$: 03 $\text{bad}_1 \leftarrow \text{true}$; abort G^1 04 if $\exists (j_1, j_2) \in [1..N]^2$ s.t. $(K_{j_1} = K_{j_2}) \wedge (t_{j_1} = t_{j_2})$: 05 $\text{bad}_2 \leftarrow \text{true}$; abort G^2 06 $b' \xleftarrow{\$} \mathcal{A}$ 07 return b'	Oracle $\text{Oenc}(j, m_0, m_1)$ 08 $(v_1, \dots, v_\ell) \leftarrow m_b$ 09 for all $i \in [1..l]$: 10 $w_i \leftarrow v_i \oplus F(K_j, t_j \ [i]_L)$ 11 $c \leftarrow (w_1, \dots, w_\ell)$ 12 return (t_j, c) Oracle $F(K, v \ [i]_L)$ 13 for all $j \in [1..N]$: 14 if $(K = K_j) \wedge (v = t_j)$: 15 if $\top[K, v \ [i]_L] \neq \perp$: 16 $\text{bad}_3 \leftarrow \text{true}$; abort G^3 17 if $\top[K, v \ [i]_L] = \perp$: 18 $\top[K, v \ [i]_L] \xleftarrow{\$} \mathcal{D}$ 19 return $\top[K, v \ [i]_L]$
--	---

Fig. 12. The security game $G_{A,N}^{0,b}$ for $\text{CTR}\|$ in the random oracle model, and games $G_{A,N}^{1,b}$, $G_{A,N}^{2,b}$, and $G_{A,N}^{3,b}$. Adversary \mathcal{A} can query the oracle Oenc at most once for the same index j .

Adversary $A_{N,\ell}$ 00 $v_0 \xleftarrow{\$} \mathcal{B}$ 01 $m_0 \leftarrow v_0 \ \dots \ v_0$ 02 for all $j \in [1..N]$: 03 for all $i \in [1..l]$: 04 $v_i^j \xleftarrow{\$} \mathcal{B}$ 05 $m_1^j \leftarrow v_1^j \ \dots \ v_\ell^j$ 06 $c^j \leftarrow \text{Oenc}(m_0, m_1^j)$ 07 $(w_1^j, \dots, w_\ell^j) \leftarrow c^j$ 08 if $\text{Coll}_2[w_1^j, \dots, w_\ell^j]$: 09 return 1 10 return 0

Fig. 13. Definition of adversary $A_{N,\ell}$ against U-MIOT-IND security of $\text{CTR}+$ instantiated with a permutation $F(K, \cdot)$. In line 01 message m_0 is made of ℓ identical blocks.

permutations, then the minimum cardinality of $\mathcal{B} = \mathcal{D}$ for achieving security is considerably increased (e.g., to values around 2^{256}).

Our argument involves the analysis of a U-MIOT-IND adversary \mathcal{A} that is specified in Fig. 13. Effectively, the idea of the attack is exploiting the tightness gap of the PRP/PRF switching lemma [5] via the multi-instance setting. More concretely, the adversary repeats the following multiple times (once for each instance): It asks either for the encapsulation of a message comprised of identical blocks, or for the encapsulation of a message consisting of uniformly-generated blocks. The adversary outputs 1 if any two blocks that form the ciphertext collide. If the ciphertext is the encapsulation of the identical-block message then the adversary does not find a collision, since $F(K, \cdot)$ is a permutation for each key $K \in \mathcal{K}$ and is evaluated on distinct input values. Otherwise the ciphertext blocks are random, and one can thus find a collision.

The theorem uses the technical condition that $N\ell(\ell - 1)/|\mathcal{T}| \leq 4$, where ℓ is a parameter that determines the length of the encapsulated messages, measured in blocks. Note that adversaries that could process values N, ℓ that are too large to fulfill this bound will reach at least the same advantage as adversaries considered by the theorem, simply by refraining from posing queries. The stated lower-bound is roughly $N\ell^2/|\mathcal{T}|$ and effectively induced by N applications of the PRP/PRF switching lemma. Note that if the above condition is met with equality, the adversary's advantage is at least $1/2$. Further, if $|\mathcal{T}| = |\mathcal{B}| = 2^{128}$, $\ell = 2^{40}$ (this corresponds to a message length of 16 terabytes) and we have $N = 2^{48}$ instances, the success probability of \mathbf{A} is about $1/8$, or larger.

Theorem 7. *Consider CTR+ instantiated with a family of permutations $F(K, \cdot)$ over \mathcal{B} , and let $N \geq 2$. Assume moreover that $N\ell(\ell - 1) \leq 4 \cdot |\mathcal{T}|$. Then for the adversary \mathbf{A} in Fig. 13 it holds:*

$$\mathbf{Adv}_{\text{CTR}^+, \mathbf{A}, N}^{\text{u-miot-ind}} \geq \frac{N\ell(\ell - 1)}{8 \cdot |\mathcal{T}|}.$$

The adversary has a running time of $\mathcal{O}(N\ell \log \ell)$, makes N queries to Oenc for messages of length at most ℓ and makes no Odec queries.

Proof. We start with the analysis of the running time of \mathbf{A} : It is predominantly determined by the search for collisions among ℓ blocks for each of the N iterations of the main loop, hence the bound of $\mathcal{O}(N\ell \log \ell)$ on the time. We now compute the probability that the adversary outputs 1 depending on the game bit b .

CASE U-MIOT-IND⁰. For each instance $j \in [1..N]$ the adversary obtains an encapsulation of a sequence of identical blocks. All blocks composing c^j must be distinct, since for each key K , function $F(K, \cdot)$ is a permutation over \mathcal{B} . Therefore the output of this game is always 0 and we have $\Pr[\text{U-MIOT-IND}_{\mathbf{A}, N}^0] = 0$.

CASE U-MIOT-IND¹. Let p be the probability that there is a collision between ℓ random variables that are uniformly distributed in the set \mathcal{B} . We show that for each $j \in [1..N]$ the probability of \mathbf{A} to output 1 when running the j -th iteration of the loop is p . From the definition of Oenc we can write $w_i^j = v_i^j \oplus F(K_j, \llbracket t_j + i \rrbracket_L)$ for each $i \in [1.. \ell]$, where K_j and t_j are the key-tag pairs generated by the game $\text{U-MIOT-IND}_{\mathbf{A}, N}^1$. The elements v_1^j, \dots, v_ℓ^j are generated uniformly in \mathcal{B} and independently of K_j, t_j , their index, and from each other. Hence the elements w_1^j, \dots, w_ℓ^j are also uniformly distributed in \mathcal{B} and mutually independent, even in the presence of colliding keys among K_1, \dots, K_N . Since all blocks v_i^j with $i \in [1.. \ell]$ and $j \in [1.. N]$ are independently random, the probability that the adversary outputs 1 is:

$$\Pr[\text{U-MIOT-IND}_{\mathbf{A}, N}^1] = 1 - (1 - p)^N. \quad (13)$$

Since $\ell(\ell - 1) \leq 2|\mathcal{B}| = 2|\mathcal{T}|$ by our hypotheses we can use the birthday bound to bound the probability p as $p \geq \ell(\ell - 1)/(4 \cdot |\mathcal{B}|)$. With some simple algebra, and since $N\ell(\ell - 1) \leq 4|\mathcal{T}| = 4|\mathcal{B}|$, we can bound Eq. 13 as:

$$\Pr[\text{U-MIOT-IND}_{\mathcal{A},N}^1] \geq \min \left\{ \frac{1}{2}, \frac{Np}{2} \right\} \geq \frac{N\ell(\ell-1)}{8 \cdot |\mathcal{B}|} = \frac{N\ell(\ell-1)}{8 \cdot |\mathcal{T}|}.$$

7 ADEMs Secure Against Active Adversaries

In the preceding section we proposed two ADEMs and proved them multi-instance secure against passive adversaries. However, the constructions are based on counter mode encryption and obviously vulnerable in settings with active adversaries that manipulate ciphertexts on the wire. In this section we alleviate the situation by constructing ADEMs that remain secure in the presence of active attacks. Concretely, in line with the encrypt-then-MAC approach [6], we show that an ADEM that is secure against active adversaries can be built from one that is secure against passive adversaries by tamper-protecting its ciphertexts using a message authentication code (MAC). More precisely, with the goal of *tightly* achieving multi-instance security, we use an *augmented message authentication code* (see footnote 4) (AMAC) where the generation and verification algorithms depend on an auxiliary input: the tag. In the combined construction, the same tag is used for both ADEM and AMAC. As before, using KEM ciphertexts as tags is a reasonable choice. We conclude the section by constructing a (tightly) secure AMAC based on a hash function.

7.1 Augmented Message Authentication

AUGMENTED MESSAGE AUTHENTICATION. An augmented message authentication code $\text{AMAC} = (\text{M.mac}, \text{M.vrf})$ for a message space \mathcal{M} is a pair of deterministic algorithms associated with a finite key space \mathcal{K} , a tag space \mathcal{T} , and a code space \mathcal{C} . The algorithm M.mac takes a key $K \in \mathcal{K}$, a tag $t \in \mathcal{T}$, and a message $m \in \mathcal{M}$, and outputs a code $c \in \mathcal{C}$. The verification algorithm M.vrf takes a key $K \in \mathcal{K}$, a tag $t \in \mathcal{T}$, a message $m \in \mathcal{M}$, and a code $c \in \mathcal{C}$, and outputs either *true* or *false*. The correctness requirement is that for all $K \in \mathcal{K}$, $t \in \mathcal{T}$, $m \in \mathcal{M}$ and $c \in [\text{M.mac}(K, t, m)]$ we have $\text{M.vrf}(K, t, m, c) = \text{true}$.

AUGMENTED MESSAGE AUTHENTICATION WITH NONCES. We give a game-based authenticity model for AMACs.¹¹ In our model, for each of a total of N independent keys the adversary can request one MAC code computation but many verifications. The restriction is that for each key the MAC query has to precede all verification queries, and that always the same tag is used. Further, in

¹¹ In principle we could give two security definitions: one using uniform tags and one using nonce tags. In this paper we formalize only the latter, not the former, for mainly two reasons: (a) the nonce-based notion is not required for our results; (b) in the nonce setting it is not clear how to prove a result similar to the one of Theorem 8. The reason for (b) is that to simulate an encapsulation query for a U-MIOT-IND adversary using an AMAC oracle one must specify the tag that is also used to generate the DEM ciphertext, but this is only given as an output of the AMAC oracle.

line with the definition of nonce-based security for ADEMs, we require the tag provided in each MAC computation request to be unique (across all instances). We formalize the corresponding security notion of (strong) nonce-based multi-instance one-time unforgeability for AMACs via the game specified in Fig. 14. For a scheme AMAC, to any adversary A and any number of instances N we associate the advantage $\mathbf{Adv}_{\text{AMAC},A,N}^{\text{n-miot-uf}} := \Pr[\mathbf{N-MIOT-UF}_{A,N}]$.

Game $\text{N-MIOT-UF}_{A,N}$	Oracle $\text{Omac}(j, t, m)$	Oracle $\text{Ovrf}(j, m, c)$
00 $\text{forged} \leftarrow 0$	07 if $C_j \neq \emptyset$: return \perp	13 if $C_j = \emptyset$: return \perp
01 $T \leftarrow \emptyset$	08 if $t \in T$: return \perp	14 if $(m, c) \in C_j$: return \perp
02 for all $j \in [1..N]$:	09 $T \leftarrow T \cup \{t\}; t_j \leftarrow t$	15 if $\text{M.vrf}(K_j, t_j, m, c)$:
03 $K_j \xleftarrow{\$} \mathcal{K}$	10 $c \leftarrow \text{M.mac}(K_j, t_j, m)$	16 $\text{forged} \leftarrow 1$
04 $C_j \leftarrow \emptyset$	11 $C_j \leftarrow C_j \cup \{(m, c)\}$	17 return <i>true</i>
05 run A	12 return c	18 return <i>false</i>
06 return <i>forged</i>		

Fig. 14. AMAC security game $\text{N-MIOT-UF}_{A,N}$, modeling nonce-based multi-instance one-time unforgeability for N instances. The tags in line 15 are the same as the ones in line 10.

7.2 The ADEM-Then-AMAC Construction

Let ADEM and AMAC be an ADEM and an AMAC, respectively. Following the generic encrypt-then-MAC [6] composition technique, and assuming ADEM is secure against passive adversaries, we combine the two schemes to obtain the augmented data-encapsulation mechanism ADEM' , which we prove secure against active adversaries. More formally, if $\text{ADEM} = (\text{A.enc}, \text{A.dec})$ and $\text{AMAC} = (\text{M.mac}, \text{M.vrf})$ have key spaces \mathcal{K}_{dem} and \mathcal{K}_{mac} , respectively, then the key space of ADEM' is $\mathcal{K}_{\text{dem}} \times \mathcal{K}_{\text{mac}}$, and its algorithms are as in Fig. 15. Note that the tag space is the same for all three schemes (and that the message spaces have to be sufficiently compatible to each other).

Proc $\text{A.enc}'(K, t, m)$	Proc $\text{A.dec}'(K, t, c)$
00 $(K_{\text{dem}}, K_{\text{mac}}) \leftarrow K$	05 $(K_{\text{dem}}, K_{\text{mac}}) \leftarrow K$
01 $c_{\text{dem}} \leftarrow \text{A.enc}(K_{\text{dem}}, t, m)$	06 $(c_{\text{dem}}, c_{\text{mac}}) \leftarrow c$
02 $c_{\text{mac}} \leftarrow \text{M.mac}(K_{\text{mac}}, t, c_{\text{dem}})$	07 if $\text{M.vrf}(K_{\text{mac}}, t, c_{\text{dem}}, c_{\text{mac}})$:
03 $c \leftarrow (c_{\text{dem}}, c_{\text{mac}})$	08 $m \leftarrow \text{A.dec}(K_{\text{dem}}, t, c_{\text{dem}})$
04 return c	09 return m
	10 return \perp

Fig. 15. Construction of ADEM' from ADEM and AMAC.

The proof of the following theorem can be found in the full version [14].

Theorem 8. *Let ADEM' be constructed from ADEM and AMAC as described. Then for any number of instances N and any ADEM adversary A that poses at most Q_d -many Odec queries, there exist an AMAC adversary B and an ADEM adversary C such that*

$$\text{Adv}_{\text{ADEM}', A, N}^{\text{n-miot-ind}} \leq 2\text{Adv}_{\text{AMAC}, B, N}^{\text{n-miot-uf}} + \text{Adv}_{\text{ADEM}, C, N}^{\text{n-miot-ind}}.$$

The running time of B is at most that of A plus the time required to run N -many ADEM encapsulations and Q_d -many ADEM decapsulations. The running time of C is the same as the running time of A . Moreover, B poses at most Q_d -many Ovrf queries, and C poses no Odec query.

7.3 A Multi-instance Secure AMAC

A random oracle directly implies a multi-instance secure AMAC , with a straightforward construction: the MAC code of a message is computed by concatenating key, tag, and message, and hashing the result. We formalize this as follows. Let \mathcal{T} be a tag space and \mathcal{M} a message space. Let \mathcal{K} and \mathcal{C} be arbitrary finite sets. Let $H: \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{C}$ be a hash function. Define function M.mac and a predicate M.vrf such that for all K, t, m, c we have $\text{M.mac}(K, t, m) = H(K, t, m)$, and $\text{M.vrf}(K, t, m, c) = \text{true}$ iff $H(K, t, m) = c$. Let finally $\text{AMAC} = (\text{M.mac}, \text{M.vrf})$.

Note that hash functions based on the Merkle–Damgård design, like SHA256 , do not serve directly as random oracles due to generic length-extension attacks [10], and indeed the ADEM' scheme from Fig. 15 is not secure if its AMAC is derived from such a function. Fortunately, Merkle–Damgård hashing can be modified to achieve indistinguishability from a random oracle [10]. Further, more recent hash functions like SHA3 are naturally resilient against length-extension attacks.

The proof of the following theorem can be found in the full version [14].

Theorem 9. *Let $\mathcal{K}, \mathcal{T}, \mathcal{M}, \mathcal{C}$ and $\text{AMAC} = (\text{M.mac}, \text{M.vrf})$ be as above. If H behaves like a (non-programmable) random oracle, for any number of instances N and any adversary A we obtain*

$$\text{Adv}_{\text{AMAC}, A, N}^{\text{n-miot-uf}} \leq \frac{q}{|\mathcal{K}|} + \left(\frac{1}{|\mathcal{K}|} + \frac{1}{|\mathcal{C}|} \right) Q_v,$$

where q is the number of direct calls to the random oracle by the adversary, and Q_v is the number of calls to the oracle Ovrf . Note that the bound does not depend on the number of Omac queries.

Acknowledgments. We are grateful to Krzysztof Pietrzak and the anonymous reviewers for their valuable comments. The authors were partially supported by ERC Project ERCC (FP7/615074) and by DFG SPP 1736 Big Data.

References

1. Attrapadung, N., Hanaoka, G., Yamada, S.: A framework for identity-based encryption with almost tight security. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015 Part I. LNCS, vol. 9452, pp. 521–549. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_22
2. Bellare, M.: New proofs for NMAC and HMAC: security without collision resistance. *J. Cryptol.* **28**(4), 844–878 (2015)
3. Bellare, M., Bernstein, D.J., Tessaro, S.: Hash-function based PRFs: AMAC and its multi-user security. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016 Part I. LNCS, vol. 9665, pp. 566–595. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_22
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
5. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_32
6. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_41
7. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016 Part I. LNCS, vol. 9814, pp. 247–276. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_10
8. Chatterjee, S., Kobitz, N., Menezes, A., Sarkar, P.: Another look at tightness II: practical issues in cryptography. *Cryptology ePrint Archive*, Report 2016/360 (2016)
9. Cogliani, S., Maimuř, D.ř., Naccache, D., do Canto, R.P., Reyhanitabar, R., Vaudenay, S., Vizár, D.: OMD: a compression function mode of operation for authenticated encryption. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 112–128. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13051-4_7
10. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: how to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_26
11. 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)
12. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016 Part I. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_1
13. Gaži, P., Pietrzak, K., Tessaro, S.: Generic security of NMAC and HMAC with input whitening. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015 Part II. LNCS, vol. 9453, pp. 85–109. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_4
14. Giacon, F., Kiltz, E., Poettering, B.: Hybrid encryption in a multi-user setting, revisited. *Cryptology ePrint Archive*, Report 2017/843 (2017)

15. Gong, J., Chen, J., Dong, X., Cao, Z., Tang, S.: Extended nested dual system groups, revisited. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016 Part I. LNCS, vol. 9614, pp. 133–163. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_6
16. Herranz, J., Hofheinz, D., Kiltz, E.: Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.* **208**(11), 1243–1257 (2010)
17. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_35
18. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_31
19. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014 Part II. LNCS, vol. 8874, pp. 1–21. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_1
20. Libert, B., Peters, T., Joye, M., Yung, M.: Compactly hiding linear spans. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015 Part I. LNCS, vol. 9452, pp. 681–707. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_28
21. Patarin, J.: Security in $O(2^n)$ for the xor of two random permutations—proof with the standard H technique. *Cryptology ePrint Archive*, Report 2013/368 (2013)
22. Zaverucha, G.: Hybrid encryption in the multi-user setting. *Cryptology ePrint Archive*, Report 2012/159 (2012)



KEM Combiners

Federico Giacon^{1(✉)}, Felix Heuer¹, and Bertram Poettering²

¹ Ruhr University Bochum, Bochum, Germany

{federico.giacon,felix.heuer}@rub.de

² Royal Holloway, University of London, London, UK

bertram.poettering@rhul.ac.uk

Abstract. Key-encapsulation mechanisms (KEMs) are a common stepping stone for constructing public-key encryption. Secure KEMs can be built from diverse assumptions, including ones related to integer factorization, discrete logarithms, error correcting codes, or lattices. In light of the recent NIST call for post-quantum secure PKE, the zoo of KEMs that are believed to be secure continues to grow. Yet, on the question of which is the *most* secure KEM opinions are divided. While using the best candidate might actually not seem necessary to survive everyday life situations, placing a wrong bet can actually be devastating, should the employed KEM eventually turn out to be vulnerable.

We introduce KEM combiners as a way to garner trust from different KEM constructions, rather than relying on a single one: We present efficient black-box constructions that, given any set of ‘ingredient’ KEMs, yield a new KEM that is (CCA) secure as long as at least one of the ingredient KEMs is.

As building blocks our constructions use cryptographic hash functions and blockciphers. Some corresponding security proofs require idealized models for these primitives, others get along on standard assumptions.

Keywords: Secure combiners · CCA security · Practical constructions

1 Introduction

Motivation for PKE combiners. Out of the public-key encryption schemes RSA-OAEP, Cramer–Shoup, ECIES, and a scheme based on the LWE hardness assumption, which one is, security-wise, the best? This question has no clear answer, as all schemes have advantages and disadvantages. For instance, RSA-OAEP is based on the arguably best studied hardness assumption but requires a random oracle. Cramer–Shoup encryption does not require a random oracle but its security reduces ‘only’ to a decisional assumption (DDH). While one can give a security reduction for ECIES to a computational assumption (CDH), this reduction comes with a tightness gap much bigger than that of RSA-OAEP. On the other hand, the ‘security-per-bit ratio’ for elliptic curve groups is assumed

The full version of this article can be found in the IACR eprint archive as article 2018/024 (<https://eprint.iacr.org/2018/024>).

to be much better than for RSA based schemes. Finally, the LWE scheme is the only quantum-resistant candidate, although the assumption is relatively new and arguably not yet well understood. All in all, the challenge of picking the most secure PKE scheme is arguably impossible to solve. Fortunately, the challenge can be side-stepped by using a ‘PKE combiner’: Instead of using only one scheme to encrypt a message, one uses all four of them, combining them in a way such that security of any implies security of their combination. Thus, when using a combiner, placing wrong bets is impossible. PKE combiners have been studied in [6, 22] and we give some details on encryption combiners below.

Combiners for other cryptographic primitives. In principle, secure combiners can be studied for any cryptographic primitive. For some primitives they are easily constructed and known for quite some time. For instance, sequentially composing multiple independently keyed blockciphers to a single keyed permutation can be seen as implementing a (S)PRP combiner. PRFs can be combined by XORing their outputs into a single value. More intriguing is studying hash function combiners: Parallely composing hash functions is a good approach if the goal is collision resistance, but pre-image resistance suffers from this. A sequential composition would be better with respect to the latter, but this again harms collision resistance. Hash function combiners that preserve both properties simultaneously exist and can be based on Feistel structures [9]. If indifferentiability from a random oracle is an additional goal, pure Feistel systems become insecure and more involved combiners are required [10, 11]. Recently, also combiners for indistinguishability obfuscation have been proposed [1, 8]. For an overview of combiners in cryptography we refer to [14, 15].

Our target: KEM combiners. Following the contemporary KEM/DEM design principle of public-key encryption [4], in this work we study combiners for key-encapsulation mechanisms (KEMs). That is, given a set of KEMs, an unknown subset of which might be arbitrarily insecure, we investigate how they can be combined to form a single KEM that is secure if at least one ingredient KEM is. How such a combiner is constructed certainly depends on the specifics of the security goal. For instance, if CPA security shall be reached then it can be expected that combining a set of KEMs by running the encapsulation algorithms in parallel and XORing the established session keys together is sufficient. However, if CCA security is intended this construction is obviously weak.

The focus of this paper is on constructing combiners for CCA security. We propose several candidates and analyze them.¹ We stress that our focus is on practicality, i.e., the combiners we propose do not introduce much overhead and are designed such that system engineers can easily adopt them. Besides the ingredient KEMs, our combiners also mix in further cryptographic primitives like blockciphers, PRFs, or hash functions. We consider this an acceptable compromise, since they make secure constructions very efficient and arguably are not

¹ Obviously, showing *feasibility* is not a concern for KEM combiners as combiners for PKE have already been studied (see Sect. 1.2) and the step from PKE to KEMs is minimal.

exposed to the threats we want to hedge against. For instance, the damage that quantum computers do on AES and SHA256 are generally assumed to be limited and controllable, tightness gaps can effectively and cheaply be closed by increasing key lengths and block sizes, and their security is often easier to assume than that of number-theoretic assumptions. While, admittedly, for some of our combiners we do require strong properties of the symmetric building blocks (random oracle model, ideal cipher model, etc.), we also construct a KEM combiner that is, at a higher computational cost, secure in the standard model. In the end we offer a selection of combiners, all with specific security and efficiency features, so that for every need there is a suitable one.

1.1 Our Results

The KEM combiners treated in this paper have a parallel structure: If the number of KEMs to be combined is n , a public key of the resulting KEM consists of a vector of n public keys, one for each ingredient; likewise for secret keys. The encapsulation procedure performs n independent encapsulations, one for each combined KEM. The ciphertext of the resulting KEM is simply the concatenation of all generated ciphertexts. The session key is obtained as a function W of keys and ciphertexts (which is arguably the *core function* of the KEM combiner). A first proposal for a KEM combiner would be to use as session key the value

$$K = H(k_1, \dots, k_n, c_1, \dots, c_n),$$

where H is a hash function modeled as a random oracle and the pair (k_i, c_i) is the result of encapsulation under the i th ingredient KEM. A slightly more efficient combiner would be

$$K = H(k_1 \oplus \dots \oplus k_n, c_1, \dots, c_n),$$

where the input session keys are XOR-combined before being fed into the random oracle. On the one hand these constructions are secure, as we prove, but somewhat unfortunate is that they depend so strongly on H behaving like a random oracle: Indeed, if the second construction were to be reinterpreted as

$$K = F(k_1 \oplus \dots \oplus k_n, c_1 \parallel \dots \parallel c_n),$$

where now F is a (standard model) PRF, then the construction would be insecure (more precisely, we prove that there exists a PRF such that when it is used in the construction the resulting KEM is insecure). The reason for the last construction not working is that the linearity of the XOR operation allows for conducting related-key attacks on the PRF, and PRFs in general are not immune against such attacks.

Our next proposal towards a KEM combiner that is provably secure in the standard model involves thus a stronger “key-mixing component”, i.e., one that is stronger than XOR. Concretely, we study the design that derives the PRF key

from a chain of blockcipher invocations, each with individual key, on input the fixed value 0. We obtain

$$K = F(\pi_{k_n} \circ \dots \circ \pi_{k_1}(0), c_1 \parallel \dots \parallel c_n),$$

where π_k represents a blockcipher π keyed with key k . Unfortunately, also this construction is generally not secure in the standard model. Yet it is—overall—our favorite construction, for the following reason: In practice, one could instantiate F with a construction based on SHA256 (prepend the key to the message before hashing it, or use NMAC or HMAC), and π with AES. Arguably, SHA256 and AES likely behave well as PRFs and PRPs, respectively; further, in principle, SHA256 is a good candidate for a random oracle and AES is a good candidate for an ideal cipher. Our results on above combiner are as follows: While the combiner is not secure if F and π are a standard model PRF and PRP, respectively, two sufficient conditions for the KEM combiner being secure are that F is a random oracle and π a PRP *or* F is a PRF and π an ideal cipher. That is, who uses the named combiner can afford that one of the two primitives, SHA256 or AES, fails to behave like an ideal primitive. Observe that this is a clear advantage over our first two (random oracle based) combiners for which security is likely gone in the moment hash function H fails to be a random oracle.

The attentive reader might have noticed that, so far, we did not propose a KEM combiner secure in the standard model. As our final contribution we remedy this absence. In fact, by following a new approach we propose a standard-model secure KEM combiner. Concretely, if below we write $c = c_1 \parallel \dots \parallel c_n$ for the ciphertext vector, our first standard model KEM combiner is

$$K = F(k_1, c) \oplus \dots \oplus F(k_n, c).$$

While being provably secure if F is a (standard model) PRF, the disadvantage over the earlier designs that are secure in idealized models is that this construction is less efficient, requiring n full passes over the ciphertext vector. Whether this is affordable or not depends on the particular application and the size of the KEM ciphertexts (which might be large for post-quantum KEMs).

In the full version of this paper (see [13]) we give an optimized variant of above combiner where the amount of PRF-processed data is slightly smaller. Exploiting that the ciphertexts of CCA secure KEMs are non-malleable (in the sense of: If a single ciphertext bit flips the session key to which this ciphertext decapsulates is independent of the original one) we observe that the PRF invocation associated with the i th session key actually does not need to process the i th ciphertext component. More precisely, if for all i we write $c^i = c_1 \parallel \dots \parallel c_{i-1} \parallel c_{i+1} \parallel \dots \parallel c_n$, then also

$$K = F(k_1, c^1) \oplus \dots \oplus F(k_n, c^n)$$

is a secure KEM combiner.

SPLIT-KEY PSEUDORANDOM FUNCTIONS. Note that in all our constructions the session keys output by the KEM combiner are derived via a function of the form

$$K = W(k_1, \dots, k_n, c),$$

where k_i denotes the key output by the encapsulation algorithm of KEM K_i and $c = c_1 \parallel \dots \parallel c_n$. We refer to W as *core function*. We can pinpoint a sufficient condition of the core function such that the respective KEM combiner retains CCA security of any of its ingredient KEMs: Intuitively, *split-key pseudorandomness* captures pseudorandom behavior of W as long as any of the keys k_1, \dots, k_n is uniformly distributed (and the other keys known to or controlled by the adversary).

All KEM combiners studied in this work that retain CCA security may be found in Fig. 1.

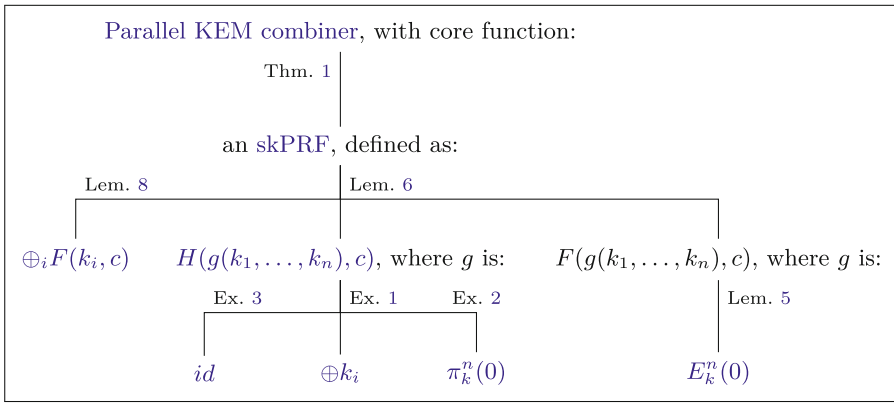


Fig. 1. Overview of our CCA-preserving KEM combiners for n KEMs. F denotes a PRF, H a random oracle, π a keyed permutation, and E an ideal cipher. Moreover, we assume $c = c_1 .. c_n$, $k = k_1 .. k_n$ and write \oplus_i for $\oplus_{i=1}^n$. For $x \in \{\pi, E\}$ we write $x_k^n(\cdot)$ to denote $x_{k_n}(\dots x_{k_1}(\cdot) \dots)$. The left-most construction, $\oplus_i F(k_i, c)$, is secure in the standard model, while the remaining constructions require idealized primitives to be proven secure.

1.2 Related Work

To the best of our knowledge KEM combiners have not been studied in the literature before. However, closely related, encryption combiners were considered. The idea of encrypting multiple times to strengthen security guarantees dates back to the seminal work of Shannon [21].

An immediate and well-studied solution (e.g. [5, 19]) to combine various symmetric encryption schemes is to apply them in a cascade fashion where the message is encrypted using the first scheme, the resulting ciphertext then being encrypted with the second scheme, and so on. Even and Goldreich [7] showed that such a chain is at least as secure as its weakest link.

Focusing on combining PKE schemes and improving on prior work (see [23]) Dodis and Katz [6] gave means to employ various PKE schemes that retain CCA security of any ‘ingredient’ scheme.

More recently, the work of [22] gave another way to combine PKE schemes ensuring that CCA security of any ingredient PKE is passed on to the combined PKE scheme. As a first step, their approach constructs a combiner achieving merely *detectable* CCA (DCCA) security² if any ingredient PKE scheme is CCA secure. Secondly, a transformation from DCCA to CCA security (see [17]) is applied to strengthen the PKE combiner.

Conceptually interesting in the context of this paper is the work of [2] where the authors propose an LWE-based key exchange and integrate it into the TLS protocol suite. The goal is to make TLS future proof (against quantum computers). Thereby, they define not only two LWE-based cipher suites, but also two hybrid ones that, conservatively with respect to the security assumptions, combine the LWE techniques with better-studied cyclic group based Diffie–Hellman key exchange.

2 Preliminaries

Notation. We use the following operators for assigning values to variables: The symbol ‘ \leftarrow ’ is used to assign to a variable (on the left-hand side) a constant value (on the right-hand side), for example the output of a deterministic algorithm. Similarly, we use ‘ $\leftarrow_{\mathfrak{s}}$ ’ to assign to a variable either a uniformly sampled value from a set or the output of a randomized algorithm. If $f: A \rightarrow B$ is a function or a deterministic algorithm we let $[f] := f(A) \subseteq B$ denote the image of A under f ; if $f: A \rightarrow B$ is a randomized algorithm with randomness space R we correspondingly let $[f] := f(A \times R) \subseteq B$ denote the set of all its possible outputs.

Let T be an associative array (also called array, or table), and b any element. Writing ‘ $T[\cdot] \leftarrow b$ ’ we set $T[a]$ to b for all a . We let $[T]$ denote the space of all elements the form $T[a]$ for some a , excluding the rejection symbol \perp . Moreover, $[T[a, \cdot]]$ is the set of all the elements assigned to $T[a, a']$ for any value a' .

Games. Our security definitions are given in terms of *games* written in pseudocode. Within a game a (possibly) stateful *adversary* is explicitly invoked. Depending on the game, the adversary may have oracle access to specific procedures. We write $\mathcal{A}^{\mathcal{O}}$, to indicate that algorithm \mathcal{A} has oracle access to \mathcal{O} . Within an oracle, command ‘Return X ’ returns X to the algorithm that called the oracle.

A game terminates when a ‘Stop with X ’ command is executed; X then serves as the output of the game. We write ‘Abort’ as an abbreviation for ‘Stop with 0’. With ‘ $G \Rightarrow 1$ ’ we denote the random variable (with randomness space

² A confidentiality notion that interpolates between CPA and CCA security. Here, an adversary is given a crippled decryption oracle that refuses to decrypt a specified set of efficiently recognizable ciphertexts. See [17].

specified by the specifics of the game G) that returns true if the output of the game is 1 and false otherwise.

In proofs that employ game hopping, lines of code that end with a comment of the form ‘ $|G_i - G_j$ ’ (resp. ‘ $|G_i, G_j$ ’, ‘ $|G_i$ ’) are only executed when a game in $G_i - G_j$ (resp. G_i and G_j , G_i) is run.

Key encapsulation. A key-encapsulation mechanism (KEM) $K = (K.gen, K.enc, K.dec)$ for a finite session-key space \mathcal{K} is a triple of algorithms together with a public-key space \mathcal{PK} , a secret-key space \mathcal{SK} , and a ciphertext space \mathcal{C} . The randomized key-generation algorithm $K.gen$ returns a public key $pk \in \mathcal{PK}$ and a secret key $sk \in \mathcal{SK}$. The randomized encapsulation algorithm $K.enc$ takes a public key $pk \in \mathcal{PK}$ and produces a session key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$. Finally, the deterministic decapsulation algorithm $K.dec$ takes a secret key $sk \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$, and outputs either a session key $k \in \mathcal{K}$ or the special symbol $\perp \notin \mathcal{K}$ to indicate rejection. For correctness we require that for all $(pk, sk) \in [K.gen]$ and $(k, c) \in [K.enc(pk)]$ we have $K.dec(sk, c) = k$.

We now give a security definition for KEMs that formalizes session-key indistinguishability. For a KEM K , associate with any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ its advantage $Adv_K^{kind}(\mathcal{A})$ defined as $|\Pr[KIND^0(\mathcal{A}) \Rightarrow 1] - \Pr[KIND^1(\mathcal{A}) \Rightarrow 1]|$, where the games are in Fig. 2. We sometimes refer to adversaries that refrain from posing queries to the Dec oracle as *passive* or *CPA*, while we refer to adversaries that pose such queries as *active* or *CCA*. Intuitively, a KEM is *CPA secure* (respectively, *CCA secure*) if all practical CPA (resp., CCA) adversaries achieve a negligible distinguishing advantage.

Game $KIND^b(\mathcal{A})$	Oracle $Dec(c)$
00 $C^* \leftarrow \emptyset$	08 If $c \in C^*$: Abort
01 $(pk, sk) \leftarrow_{\$} K.gen$	09 $k \leftarrow K.dec(sk, c)$
02 $st \leftarrow_{\$} \mathcal{A}_1^{Dec}(pk)$	10 Return k
03 $(k^*, c^*) \leftarrow_{\$} K.enc(pk)$	
04 $k^0 \leftarrow k^*$; $k^1 \leftarrow_{\$} \mathcal{K}$	
05 $C^* \leftarrow C^* \cup \{c^*\}$	
06 $b' \leftarrow_{\$} \mathcal{A}_2^{Dec}(st, c^*, k^b)$	
07 Stop with b'	

Fig. 2. Security experiments $KIND^b$, $b \in \{0, 1\}$, modeling the session-key indistinguishability of KEM K . With st we denote internal state information of the adversary.

Pseudorandom functions. Fix a finite key space \mathcal{K} , an input space \mathcal{X} , a finite output space \mathcal{Y} , and a function $F: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$. Towards defining what it means for F to behave pseudorandomly, associate with any adversary \mathcal{A} its advantage $Adv_F^{pr}(\mathcal{A}) := |\Pr[PR^0(\mathcal{A}) \Rightarrow 1] - \Pr[PR^1(\mathcal{A}) \Rightarrow 1]|$, where the games are in Fig. 3. Intuitively, F is a *pseudorandom function* (PRF) if all practical adversaries achieve a negligible advantage.

Game $\text{PR}^b(\mathcal{A})$	Oracle $\text{Eval}(x)$
00 $X \leftarrow \emptyset$	04 If $x \in X$: Abort
01 $k \leftarrow_{\text{s}} \mathcal{K}$	05 $X \leftarrow X \cup \{x\}$
02 $b' \leftarrow_{\text{s}} \mathcal{A}^{\text{Eval}}$	06 $y \leftarrow F(k, x)$
03 Stop with b'	07 $y^0 \leftarrow y; y^1 \leftarrow_{\text{s}} \mathcal{Y}$
	08 Return y^b

Fig. 3. Security experiments PR^b , $b \in \{0, 1\}$, modeling the pseudorandomness of function F . Line 04 and 05 implement the requirement that Eval not be queried on the same input twice.

Pseudorandom permutations. Intuitively, a pseudorandom permutation (PRP) is a bijective PRF. More precisely, if \mathcal{K} is a finite key space and \mathcal{D} a finite domain, then function $\pi: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ is a PRP if for all $k \in \mathcal{K}$ the partial function $\pi(k, \cdot): \mathcal{D} \rightarrow \mathcal{D}$ is bijective and if $\pi(k, \cdot)$ behaves like a random permutation $\mathcal{D} \rightarrow \mathcal{D}$ once $k \in \mathcal{K}$ is assigned uniformly at random. A formalization of this concept would be in the spirit of Fig. 3. In practice, PRPs are often implemented with blockciphers.

Random oracle model, ideal cipher model. We consider a cryptographic scheme defined with respect to a hash function $H: \mathcal{X} \rightarrow \mathcal{Y}$ in the *random oracle model* for H by replacing the scheme's internal invocations of H by calls to an oracle H that implements a uniform mapping $\mathcal{X} \rightarrow \mathcal{Y}$. In security analyses of the scheme, also the adversary gets access to this oracle. Similarly, a scheme defined with respect to a keyed permutation $\pi: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ is considered in the *ideal cipher model* for π if all computations of $\pi(\cdot, \cdot)$ in the scheme algorithms are replaced by calls to an oracle $E(\cdot, \cdot)$ that implements a uniform mapping $\mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ such that $E(k, \cdot)$ is a bijection for all k , and all computations of $\pi^{-1}(\cdot, \cdot)$ are replaced by calls to an oracle $D(\cdot, \cdot)$ that implements a uniform mapping $\mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ such that $D(k, \cdot)$ is a bijection for all k , and the partial oracles $E(k, \cdot)$ and $D(k, \cdot)$ are inverses of each other (again for all k). In corresponding security analyses the adversary gets access to both E and D . We write E (resp. D) to denote π (resp. π^{-1}) every time that we want to remark that π will be considered in the ideal cipher model.

3 KEM Combiners

A KEM combiner is a template that specifies how a set of existing KEMs can be joined together, possibly with the aid of further cryptographic primitives, to obtain a new KEM. In this paper we are exclusively interested in combiners that are security preserving: The resulting KEM shall be at least as secure as any of its ingredient KEMs (assuming all further primitives introduced by the combiner are secure). While for public-key encryption a serial combination process is possible and plausible (encrypt the message with the first PKE scheme, the

resulting ciphertext with the second PKE scheme, and so on, for KEMs a parallel approach, where the ciphertext consists of a set of independently generated ciphertext components (one component per ingredient KEM), seems more natural. We formalize a general family of parallel combiners that are parameterized by a *core function* that derives a combined session key from a vector of session keys and a vector of ciphertexts.

Parallel KEM combiner. Let K_1, \dots, K_n be (ingredient) KEMs such that each $K_i = (K.\text{gen}_i, K.\text{enc}_i, K.\text{dec}_i)$ has session-key space \mathcal{K}_i , public-key space \mathcal{PK}_i , secret-key space \mathcal{SK}_i , and ciphertext space \mathcal{C}_i . Let $\mathcal{K}_* = \mathcal{K}_1 \times \dots \times \mathcal{K}_n$ and $\mathcal{PK} = \mathcal{PK}_1 \times \dots \times \mathcal{PK}_n$ and $\mathcal{SK} = \mathcal{SK}_1 \times \dots \times \mathcal{SK}_n$ and $\mathcal{C} = \mathcal{C}_1 \times \dots \times \mathcal{C}_n$. Let further \mathcal{K} be an auxiliary finite session-key space. For any *core function* $W: \mathcal{K}_* \times \mathcal{C} \rightarrow \mathcal{K}$, the *parallel combination* $K := K_1 \parallel \dots \parallel K_n$ with respect to W is a KEM with session-key space \mathcal{K} that consists of the algorithms $K.\text{gen}, K.\text{enc}, K.\text{dec}$ specified in Fig. 4. The combined KEM K has public-key space \mathcal{PK} , secret-key space \mathcal{SK} , and ciphertext space \mathcal{C} . A quick inspection of the algorithms shows that if all ingredient KEMs K_i are correct, then so is K .

Algo $K.\text{gen}$	Algo $K.\text{enc}(pk)$	Algo $K.\text{dec}(sk, c)$
00 For $i \leftarrow 1$ to n :	05 $(pk_1, \dots, pk_n) \leftarrow pk$	11 $(sk_1, \dots, sk_n) \leftarrow sk$
01 $(pk_i, sk_i) \leftarrow_{\S} K.\text{gen}_i$	06 For $i \leftarrow 1$ to n :	12 $c_1 .. c_n \leftarrow c$
02 $pk \leftarrow (pk_1, \dots, pk_n)$	07 $(k_i, c_i) \leftarrow_{\S} K.\text{enc}_i(pk_i)$	13 For $i \leftarrow 1$ to n :
03 $sk \leftarrow (sk_1, \dots, sk_n)$	08 $c \leftarrow c_1 .. c_n$	14 $k_i \leftarrow K.\text{dec}_i(sk_i, c_i)$
04 Return (pk, sk)	09 $k \leftarrow W(k_1, \dots, k_n, c)$	15 If $k_i = \perp$: Return \perp
	10 Return (k, c)	16 $k \leftarrow W(k_1, \dots, k_n, c)$
		17 Return k

Fig. 4. Parallel KEM combiner, defined with respect to some core function W .

The security properties of the parallel combiner depend crucially on the choice of the core function W . For instance, if W maps all inputs to one fixed session key $\bar{k} \in \mathcal{K}$, the obtained KEM does not inherit any security from the ingredient KEMs. We are thus left with finding good core functions W .

3.1 The XOR Combiner

Assume ingredient KEMs that share a common binary-string session-key space: $\mathcal{K}_1 = \dots = \mathcal{K}_n = \{0, 1\}^k$ for some k . Consider the XOR core function that, disregarding its ciphertext inputs, outputs the binary sum of the key inputs. Formally, after letting $\mathcal{K} = \{0, 1\}^k$ this means $\mathcal{K}_* = \mathcal{K}^n$ and

$$W: \mathcal{K}_* \times \mathcal{C} \rightarrow \mathcal{K}; \quad (k_1, \dots, k_n, c_1 .. c_n) \mapsto k_1 \oplus \dots \oplus k_n.$$

On W we prove two statements: If the overall goal is to obtain a CPA-secure KEM, then W is useful, in the sense that the parallel combination of KEMs with

respect to W is CPA secure if at least one of the ingredient KEMs is. However, if the overall goal is CCA security, then one weak ingredient KEM is sufficient to break any parallel combination with respect to W .

Lemma 1 (XOR combiner retains CPA security). *Let K_1, \dots, K_n be KEMs and let W be the XOR core function. Consider the parallel combination $K = K_1 \parallel \dots \parallel K_n$ with respect to W . If at least one K_i is CPA secure, then also K is CPA secure. Formally, for all indices $i \in [1..n]$ and every adversary \mathcal{A} that poses no queries to the decapsulation oracle there exists an adversary \mathcal{B} such that*

$$\text{Adv}_K^{\text{kind}}(\mathcal{A}) = \text{Adv}_{K_i}^{\text{kind}}(\mathcal{B}),$$

where also \mathcal{B} poses no decapsulation query and its running time is about that of \mathcal{A} .

Proof. From any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against K we construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against K_i as follows. Algorithm \mathcal{B}_1 , on input $pk_i \in \mathcal{PK}_i$, first generates the $n - 1$ public keys $pk_1, \dots, pk_{i-1}, pk_{i+1}, \dots, pk_n$ by means of $(pk_j, sk_j) \leftarrow_{\S} K.\text{gen}_j$. Then it sets $pk \leftarrow (pk_1, \dots, pk_n)$, invokes $st \leftarrow_{\S} \mathcal{A}_1(pk)$, and outputs $st' \leftarrow (st, pk_1, \dots, pk_{i-1}, pk_{i+1}, \dots, pk_n)$. Algorithm \mathcal{B}_2 , on input (st', c_i^*, k_i^*) , first invokes $(k_j^*, c_j^*) \leftarrow_{\S} K.\text{enc}_j(pk_j)$ for all $j \neq i$, and then sets $c^* \leftarrow c_1^* \dots c_i^* \dots c_n^*$ and $k^* \leftarrow k_1^* \oplus \dots \oplus k_i^* \oplus \dots \oplus k_n^*$. Finally it then invokes $b' \leftarrow_{\S} \mathcal{A}_2(st, c^*, k^*)$ and outputs b' . It is easy to see that the advantages of \mathcal{A} and \mathcal{B} coincide. \square

Remark. Consider a CCA secure KEM (for instance from the many submissions to NIST's recent Post-Quantum initiative [20]) that is constructed by, first, taking a CPA secure KEM and then applying a Fujisaki–Okamoto-like transformation [12, 16, 18] to it in order to obtain a CCA secure KEM.

To combine multiple KEMs that follow the above design principle, Lemma 1 already provides a highly efficient solution that retains CCA security: To this end, one would strip away the FO-like transformation of the KEMs to be combined and apply the XOR-combiner to the various CPA secure KEMs. Eventually one would apply an FO-like transformation to the XOR-combiner.

However, besides results shedding doubts on the instantiability of FO in the presence of indistinguishability obfuscation [3], we pursue generic KEM combiners that retain CCA security independently of how the ingredient KEMs achieve their security.

While it is rather obvious that the XOR-combiner is incapable of retaining CCA security of an ingredient KEM, we formally state and prove it next.

Lemma 2 (XOR combiner does not retain CCA security). *In general, the result of parallelly combining a CCA-secure KEM with other KEMs using the XOR core function is not CCA secure.*

Formally, if $n \in \mathbb{N}$ and W is the XOR core function, then for all $1 \leq i \leq n$ there exists a KEM K_i such that for any set of $n - 1$ KEMs K_1, \dots, K_{i-1} ,

K_{i+1}, \dots, K_n (e.g., all of them CCA secure) there exists an efficient adversary \mathcal{A} that poses a single decapsulation query and achieves an advantage of $\text{Adv}_K^{\text{kind}}(\mathcal{A}) = 1 - 1/|\mathcal{K}|$, where $K = K_1 \parallel \dots \parallel K_n$ is the parallel combination of K_1, \dots, K_n with respect to W .

Proof. We construct KEM K_i such that public and secret keys play no role, it has only two ciphertexts, and it establishes always the same session key: Fix any $\bar{k} \in \mathcal{K}$, let $\mathcal{C}_i = \{0, 1\}$, and let $K.\text{enc}_i$ and $K.\text{dec}_i$ always output $(\bar{k}, 0) \in \mathcal{K} \times \mathcal{C}_i$ and $\bar{k} \in \mathcal{K}$, respectively. Define adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that \mathcal{A}_1 does nothing and \mathcal{A}_2 , on input of c^* and k^* , parses c^* as $c_1^* \dots c_i^* \dots c_n^*$ (where $c_i^* = 0$), poses a decapsulation query $k^{**} \leftarrow \text{Dec}(c^{**})$ on ciphertext $c^{**} = c_1^* \dots c_{i-1}^* 1 c_{i+1}^* \dots c_n^*$, and outputs 1 iff $k^* = k^{**}$. It is easy to see that \mathcal{A} achieves the claimed advantage. \square

3.2 The XOR-Then-PRF Combiner

We saw that the KEM combiner that uses the core function that simply outputs the XOR sum of the session keys fails miserably to provide security against active adversaries. The main reason is that it completely ignores the ciphertext inputs, so that the latter can be altered by an adversary without affecting the corresponding session key. As an attempt to remedy this, we next consider a core function that, using a PRF, mixes all ciphertext bits into the session key that it outputs. The PRF is keyed with the XOR sum of the input session keys and shall serve as an integrity protection on the ciphertexts.

Formally, under the same constraints on $\mathcal{K}, \mathcal{K}_1, \dots, \mathcal{K}_n, \mathcal{K}_*$ as in Sect. 3.1, and assuming a (pseudorandom) function $F: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{K}$, the XOR-then-PRF core function W_F is defined as per

$$W_F: \mathcal{K}_* \times \mathcal{C} \rightarrow \mathcal{K}; \quad (k_1, \dots, k_n, c_1 \dots c_n) \mapsto F(k_1 \oplus \dots \oplus k_n, c_1 \dots c_n).$$

Of course, to leverage on the pseudorandomness of the function F its key has to be uniform. The hope, based on the intuition that at least one of the ingredient KEMs is assumed secure and thus the corresponding session key uniform, is that the XOR sum of all session keys works fine as a PRF key. Unfortunately, as we prove next, this is not the case in general. The key insight is that the pseudorandomness definition does not capture robustness against related-key attacks: We present a KEM/PRF combination where manipulating KEM ciphertexts allows to exploit a particular structure of the PRF.³

Lemma 3 (XOR-then-PRF combiner does not retain CCA security).

There exist KEM/PRF configurations such that if the KEM is parallelly combined with other KEMs using the XOR-then-PRF core function, then the resulting KEM is weak against active attacks. More precisely, for all $n \in \mathbb{N}$ and

³ Note that in Lemma 6 we prove that if F behaves like a random oracle and is thus free of related-key conditions, the XOR-then-PRF core function actually does yield a secure CCA combiner.

$i \in [1..n]$ there exists a KEM K_i and a (pseudorandom) function F such that for any set of $n-1$ (arbitrarily secure) KEMs $K_1, \dots, K_{i-1}, K_{i+1}, \dots, K_n$ there exists an efficient adversary \mathcal{A} that poses a single decapsulation query and achieves advantage $\text{Adv}_{\mathcal{K}}^{\text{kind}}(\mathcal{A}) = 1 - 1/|\mathcal{K}|$, where $\mathcal{K} = K_1 \parallel \dots \parallel K_n$ is the parallel combination of K_1, \dots, K_n with respect to the XOR-then-PRF core function W_F . Function F is constructed from a function F' such that if F' is pseudorandom then so is F .

Proof. In the following we write $\mathcal{K} = \{0, 1\} \times \mathcal{K}'$ (where $\mathcal{K}' = \{0, 1\}^{k-1}$). We construct K_i such that public and secret keys play no role, there are only two ciphertexts, and the two ciphertexts decapsulate to different session keys: Fix any $\bar{k} \in \mathcal{K}'$, let $C_i = \{0, 1\}$, let $K.\text{enc}_i$ always output $((0, \bar{k}), 0) \in \mathcal{K} \times C_i$, and let $K.\text{dec}_i$, on input ciphertext $B \in C_i$, output session key $(B, \bar{k}) \in \mathcal{K}$.

We next construct a specific function F and argue that it is pseudorandom. Consider the involution $\pi: \mathcal{C} \rightarrow \mathcal{C}$ that flips the bit value of the i th ciphertext component, i.e.,

$$\pi(c_1 \dots c_{i-1} B c_{i+1} \dots c_n) = c_1 \dots c_{i-1} (1 - B) c_{i+1} \dots c_n,$$

and let $F': \mathcal{K}' \times \mathcal{C} \rightarrow \mathcal{K}$ be a (pseudorandom) function. Construct $F: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{K}$ from π and F' as per

$$F((D, k'), c) = \begin{cases} F'(k', c) & \text{if } D = 0 \\ F'(k', \pi(c)) & \text{if } D = 1. \end{cases} \tag{1}$$

It is not difficult to see that if F' is pseudorandom then so is F . For completeness, we give a formal statement and proof immediately after this proof.

Consider now the following adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$: Let algorithm \mathcal{A}_1 do nothing, and let algorithm \mathcal{A}_2 , on input of c^* and k^* , parse the ciphertext as $c^* = c_1^* \dots c_i^* \dots c_n^*$ (where $c_i^* = 0$), pose a decapsulation query $k^{**} \leftarrow \text{Dec}(c^{**})$ on ciphertext $c^{**} = c_1^* \dots c_{i-1}^* 1 c_{i+1}^* \dots c_n^*$, and output 1 iff $k^* = k^{**}$.

Let us analyze the advantage of \mathcal{A} . For all $1 \leq j \leq n$, let $(d_j, k'_j) \in \mathcal{K}$ be the session keys to which ciphertext components c_j^* decapsulate. That is, the session key k to which c^* decapsulates can be computed as $k = F((d_1, k'_1) \oplus \dots \oplus (d_n, k'_n), c^*)$, by specification of W_F . By setting $D = d_1 \oplus \dots \oplus d_n$ and expanding F into F' and π we obtain

$$k = F'(k'_1 \oplus \dots \oplus k'_n, c_1^* \dots c_{i-1}^* D c_{i+1}^* \dots c_n^*).$$

Consider next the key k^{**} that is returned by the Dec oracle. Internally, the oracle recovers the same keys $(d_1, k'_1), \dots, (d_n, k'_n)$ as above, with exception of d_i which is inverted. Let $D^{**} = d_1 \oplus \dots \oplus d_n$ be the corresponding (updated) sum. We obtain

$$k^{**} = F'(k'_1 \oplus \dots \oplus k'_n, c_1^* \dots c_{i-1}^* (1 - D^{**}) c_{i+1}^* \dots c_n^*).$$

Thus, as D^{**} is the inverse of D , we have $k = k^{**}$ and adversary \mathcal{A} achieves the claimed advantage. \square

We now give the formal statement that, if F' is a PRF then the same holds for F as defined in (1).

Lemma 4. *Let $\mathcal{K}', \mathcal{X}, \mathcal{Y}$ be sets such that $\mathcal{K}', \mathcal{Y}$ are finite. Let $F': \mathcal{K}' \times \mathcal{X} \rightarrow \mathcal{Y}$ be a function, and let $\pi: \mathcal{X} \rightarrow \mathcal{X}$ be any (efficient) bijection.⁴ Let $\mathcal{K} = \{0, 1\} \times \mathcal{K}'$ and define function $F: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ such that*

$$F((D, k'), x) = \begin{cases} F'(k', x) & \text{if } D = 0 \\ F'(k', \pi(x)) & \text{if } D = 1. \end{cases} \quad (2)$$

Then if F' is a PRF, the same holds for F . More precisely, for every adversary \mathcal{A} there is an adversary \mathcal{B} such that

$$\text{Adv}_F^{\text{PR}}(\mathcal{A}) = \text{Adv}_{F'}^{\text{PR}}(\mathcal{B}),$$

the running times of \mathcal{A} and \mathcal{B} are roughly the same, and if \mathcal{A} queries its evaluation oracle q_e times then \mathcal{B} queries its own evaluation oracle q_e times.

Proof. Let \mathcal{A} be an adversary against the pseudorandomness of F . We build an adversary \mathcal{B} against the pseudorandomness of F' as follows. \mathcal{B} generates a bit D and runs \mathcal{A} . For every Eval query of \mathcal{A} on input x , adversary \mathcal{B} queries its own evaluation oracle on input x if $D = 0$, or $\pi(x)$ if $D = 1$. The output of this query is returned to \mathcal{A} . At the end of \mathcal{A} 's execution its output is returned by \mathcal{B} .

We argue that \mathcal{B} provides a correct simulation of the pseudorandomness games to \mathcal{A} . First we notice that if the input values to Eval by \mathcal{A} are unique, so are the input values to Eval by \mathcal{B} , since π is a bijection and D is constant during each run of the simulation. Conversely, any input repetition by \mathcal{A} leads to an input repetition by \mathcal{B} , thus aborting the pseudorandomness game. If \mathcal{B} is playing against the real game PR^0 for F' then it correctly computes the function F for \mathcal{A} and the distribution of the output to \mathcal{A} is the same as that in game PR^0 for F . Otherwise \mathcal{B} receives uniform independent elements from its oracle Eval, and hence correctly simulates the game PR^1 for F to \mathcal{A} . This proves our statement. \square

3.3 KEM Combiners from Split-Key PRFs

The two core functions for the parallel KEM combiner that we studied so far did not achieve security against active attacks. We next identify a sufficient condition that guarantees satisfactory results: If the core function is *split-key pseudorandom*, and at least one of the ingredient KEMs of the parallel combiner from Fig. 4 is CCA secure, then the resulting KEM is CCA secure as well.

⁴ No cryptographic property is required of π , just that it can be efficiently computed. An easy example is the flip-the-first-bit function.

Split-key pseudorandom functions. We say a symmetric key primitive (syntactically) uses split keys if its key space \mathcal{K} is the Cartesian product of a finite number of (sub)key spaces $\mathcal{K}_1, \dots, \mathcal{K}_n$. In the following we study the corresponding notion of split-key pseudorandom function. In principle, such functions are just a special variant of PRFs, so that the security notion of pseudorandomness (see Fig. 3) remains meaningful. However, we introduce *split-key pseudorandomness* as a dedicated, refined property. In brief, a split-key function has this property if it behaves like a random function if at least one component of its key is picked uniformly at random (while the other components may be known or even chosen by the adversary).

For formalizing this, fix finite key spaces $\mathcal{K}_1, \dots, \mathcal{K}_n$, an input space \mathcal{X} , and a finite output space \mathcal{Y} . Further, let $\mathcal{K} = \mathcal{K}_1 \times \dots \times \mathcal{K}_n$ and consider a function $F: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$. For each index $i \in [1..n]$, associate with an adversary \mathcal{A} its advantage $\text{Adv}_{F,i}^{\text{PR}}(\mathcal{A}) := |\Pr[\text{PR}_i^0(\mathcal{A}) \Rightarrow 1] - \Pr[\text{PR}_i^1(\mathcal{A}) \Rightarrow 1]|$, where the game is given in Fig. 5. Observe that, for any index i , in the game PR_i^b , $b \in \{0, 1\}$, the i th key component of F is assigned at random (in line 01), while the adversary contributes the remaining $n - 1$ components on a per-query basis (see line 06). We say that F is a *split-key pseudorandom function* (skPRF) if the advantages $\text{Adv}_{F,i}^{\text{PR}}$ for all key indices are negligible for all practical adversaries.

Games $\text{PR}_i^b(\mathcal{A})$	Oracle $\text{Eval}(k', x)$
00 $X \leftarrow \emptyset$	04 If $x \in X$: Abort
01 $k_i \leftarrow_{\S} \mathcal{K}_i$	05 $X \leftarrow X \cup \{x\}$
02 $b' \leftarrow_{\S} \mathcal{A}^{\text{Eval}}$	06 $k_1 .. k_{i-1} k_{i+1} .. k_n \leftarrow k'$
03 Stop with b'	07 $y \leftarrow F(k_1 .. k_i .. k_n, x)$
	08 $y^0 \leftarrow y; y^1 \leftarrow_{\S} \mathcal{Y}$
	09 Return y^b

Fig. 5. Security game PR_i^b , $b \in \{0, 1\}$, $1 \leq i \leq n$, modeling the split-key pseudorandomness of function F . Lines 04 and 05 implement the requirement that Eval not be queried on the same input twice.

With lines 04 and 05 we require that the oracle Eval be executed at most once on an input value x , independently on the input value k' . One could imagine a relaxed version of this requirement, where Eval accepts any non-repeating input pair (k', x) , thus permitting repeated values of x in distinct queries to Eval. Most of the following proofs are however not straightforward to be adapted to the relaxed definition, and in many case this would directly lead to an insecure construction. Notice, however, that our current definition of split-key pseudorandomness for a function F still suffices to prove that F is a standard PRF.

Theorem 1. *If the core function used in the parallel composition is split-key pseudorandom, the parallel combiner yields a CCA-secure KEM if at least one of the ingredient KEMs is CCA secure.*

More precisely, for all n, K_1, \dots, K_n , if $K = K_1 \parallel \dots \parallel K_n$ with core function W then for all indices i and all adversaries \mathcal{A} against the key indistinguishability of K there exist adversaries \mathcal{B} against the key indistinguishability of K_i and \mathcal{C} against the split-key pseudorandomness of W such that

$$\text{Adv}_K^{\text{kind}}(\mathcal{A}) \leq 2 \cdot \left(\text{Adv}_{K_i}^{\text{kind}}(\mathcal{B}) + \text{Adv}_{W,i}^{\text{pr}}(\mathcal{C}) \right).$$

Moreover, if adversary \mathcal{A} calls at most q_d times the oracle Dec , then adversary \mathcal{B} makes at most q_d calls to the oracle Dec , and adversary \mathcal{C} makes at most $q_d + 1$ calls to the oracle Eval . The running times of \mathcal{B} and \mathcal{C} are roughly the same as that of \mathcal{A} .

PROOF SKETCH. The proof constitutes of a sequence of games interpolating between games G_0 and G_4 . Noting that the KEMs we consider are perfectly correct, those two games correspond respectively to games KIND^0 and KIND^1 for the KEM $K = K_1 \parallel \dots \parallel K_n$. Code detailing the games involved is in Fig. 7 and the main differences between consecutive games are explained in Fig. 6. In a nutshell, we proceed as follows: In game G_1 we replace the key k_i^* output by $(k_i^*, c_i^*) \leftarrow_{\$} K.\text{enc}_i(pk_i)$ by a uniform key. As K_i is CCA secure this modification is oblivious to \mathcal{A} . As a second step, we replace the real challenge session key k^* as obtained via $k^* \leftarrow W(k_1^*, \dots, k_n^*, c_1^* \dots c_n^*)$ with a uniform session key in game G_2 . Since the core function W is split-key pseudorandom and k_i^* is uniform, this step is oblivious to \mathcal{A} as well. However—for technical reasons within the reduction—replacing the challenge session key will introduce an artifact to the decapsulation procedure: queries of the form $\text{Dec}(\dots, c_i^*, \dots)$ will not be processed using W but answered with uniform session keys. In the transition to game G_3 we employ the split-key pseudorandomness of W again to remove the artifact from the decapsulation oracle. Eventually, in game G_4 we undo our first modification and replace the currently uniform key k_i^* with the actual key obtained by running $K.\text{enc}_i(pk_i)$. Still, the challenge session key k^* remains uniform. Again, the CCA security of K_i ensures that \mathcal{A} will not detect the modification.

We proceed with a detailed proof.

Game	k_i^*	k^*	$\text{Dec}(\dots, c_i^*, \dots)$	Δ
$G_0 (\equiv \text{KIND}^0)$	real	real	real	$\text{Adv}_{K_i}^{\text{kind}}$
G_1	random	real	real	
G_2	random	random	random	$\text{Adv}_{W,i}^{\text{pr}}$
G_3	random	random	real	$\text{Adv}_{W,i}^{\text{pr}}$
$G_4 (\equiv \text{KIND}^1)$	real	random	real	$\text{Adv}_{K_i}^{\text{kind}}$

Fig. 6. Overview of the proof of Theorem 1. We have $(k_i^*, c_i^*) \leftarrow_{\$} K.\text{enc}_i(pk_i)$. Furthermore, $k^* \leftarrow W(k_1^*, \dots, k_n^*, c_1^* \dots c_n^*)$ denotes the challenge session key given to \mathcal{A}_2 along with $c_1^* \dots c_n^*$.

Games G_0 to G_4	Oracle $\text{Dec}(c)$
00 $C^*, C_i^* \leftarrow \emptyset; L[\cdot] \leftarrow \perp$	14 If $c \in C^*$: Abort
01 For $j \leftarrow 1$ to n :	15 If $L[c] \neq \perp$: Return $L[c]$
02 $(pk_j, sk_j) \leftarrow_{\mathcal{S}} \text{K.gen}_j$	16 $c_1 \dots c_n \leftarrow c$
03 $pk \leftarrow (pk_1, \dots, pk_n)$	17 For $j \in [1..n] \setminus \{i\}$:
04 $st \leftarrow_{\mathcal{S}} \mathcal{A}_1^{\text{Dec}}(pk)$	18 $k_j \leftarrow \text{K.dec}_j(sk_j, c_j)$
05 For $j \leftarrow 1$ to n :	19 If $k_j = \perp$: Return \perp
06 $(k_j^*, c_j^*) \leftarrow_{\mathcal{S}} \text{K.enc}_j(pk_j)$	20 If $c_i \in C_i^*$:
07 $k_i^* \leftarrow_{\mathcal{S}} \mathcal{K}_i$ G_1 - G_3	21 $k_i \leftarrow k_i^*$
08 $c^* \leftarrow c_1^* \dots c_n^*$	22 Else:
09 $k^* \leftarrow W(k_1^*, \dots, k_n^*, c^*)$	23 $k_i \leftarrow \text{K.dec}_i(sk_i, c_i)$
10 $k^* \leftarrow_{\mathcal{S}} \mathcal{K}$ G_2 - G_4	24 If $k_i = \perp$: Return \perp
11 $C^* \leftarrow C^* \cup \{c^*\}; C_i^* \leftarrow C_i^* \cup \{c_i^*\}$	25 $L[c] \leftarrow W(k_1, \dots, k_n, c)$
12 $b' \leftarrow_{\mathcal{S}} \mathcal{A}_2^{\text{Dec}}(st, c^*, k^*)$	26 If $c_i \in C_i^*$: $L[c] \leftarrow_{\mathcal{S}} \mathcal{K}$ G_2
13 Stop with b'	27 Return $L[c]$

Fig. 7. Games G_0 – G_4 as used in the proof of Theorem 1. Note that i is implicitly a parameter of all games above.

Proof (Theorem 1). Let \mathcal{A} denote an adversary attacking the CCA security of the KEM K that issues at most q_d queries to the decapsulation oracle. We proceed with detailed descriptions of the games (see Fig. 7) used in our proof.

Game G_0 . The KIND^0 game instantiated with the KEM K as given in Fig. 4. Beyond that we made merely syntactical changes: In line 00 a set C_i^* and an array L are initialized as empty. In line 15 we check if the adversary has already queried the oracle for the same input and we return the same output. Lines 20 and 21 are added such that, instead of using sk_i to decapsulate c_i^* , the key k_i^* is used. Note that if line 21 is executed then key k_i^* is already defined, since $C_i^* \neq \emptyset$.

Claim 1. $\Pr[\text{KIND}^0 \Rightarrow 1] = \Pr[G_0 \Rightarrow 1]$.

This follows immediately from the correctness of K_i and the fact that the decapsulation algorithm is deterministic.

Game G_1 . Line 07 is added to replace the key k_i^* with a uniform key from \mathcal{K}_i .

Claim 2. There is an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against session-key indistinguishability of K_i (see Fig. 8) that issues at most q_d decapsulation queries such that

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_{\text{K}_i}^{\text{kind}}(\mathcal{B}),$$

and the running time of \mathcal{B} is roughly the running time of \mathcal{A} .

Proof. We construct $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ as given in Fig. 8: Adversary \mathcal{B}_1 gets pk_i as input, and runs $(pk_j, sk_j) \leftarrow_{\mathcal{S}} \text{K.gen}_j$ for all $j \in [1..n] \setminus \{i\}$ to instantiate the other KEMs (see lines 01–03). To answer the decapsulation queries of \mathcal{A}_1 ,

\mathcal{B}_1 decapsulates all c_i for $j \neq i$ using sk_j (lines 16–18) and queries its own decapsulation oracle to decapsulate c_i (lines 21–23).

Adversary \mathcal{B}_2 , run on the challenge (c_i^*, k_i^*) , executes $(k_j^*, c_j^*) \leftarrow_{\S} \mathbf{K}.\text{enc}_j$ for $j \neq i$ on its own (lines 07, 08). Then it computes the challenge session key $k^* \leftarrow W(k_1^*, \dots, k_n^*, c_1^*, \dots, c_n^*)$ (line 10) and runs \mathcal{A}_2 on $(c_1^*, \dots, c_n^*, k^*)$ (line 12). Decryption queries are answered as in phase one unless \mathcal{B}_2 has to decapsulate c_i^* where it uses k_i^* instead (lines 19, 20). At the end \mathcal{B}_2 relays \mathcal{A}_2 's output and halts (line 13).

ANALYSIS. Games G_0 and G_1 only differ on the key k_i^* used to compute k^* for \mathcal{A}_2 , and, consequently, when answering \mathcal{A}_2 's decapsulation queries involving c_i^* . If \mathcal{B} is run by the game KIND^0 , that is, key k_i^* is a real key output of $\mathbf{K}.\text{enc}_i$, then \mathcal{B} perfectly emulates game G_0 . Otherwise, if \mathcal{B} is run by the game KIND^1 , and thus the key k_i^* is uniform, then \mathcal{B} emulates G_1 . Hence

$$\Pr[G_0 \Rightarrow 1] = \Pr[\text{KIND}^0 \Rightarrow 1]$$

and

$$\Pr[G_1 \Rightarrow 1] = \Pr[\text{KIND}^1 \Rightarrow 1].$$

Lastly we observe that \mathcal{B} issues at most as many decapsulation queries as \mathcal{A} . Our claim follows. \square

Adversary $\mathcal{B}_1^{\text{Dec}}(pk_i)$	If \mathcal{A} calls $\text{Dec}(c)$:
00 $C^*, C_i^* \leftarrow \emptyset$	14 If $c \in C^*$: Abort
01 For $j \in [1..n] \setminus \{i\}$:	15 $c_1 \dots c_n \leftarrow c$
02 $(pk_j, sk_j) \leftarrow_{\S} \mathbf{K}.\text{gen}_j$	16 For $j \in [1..n] \setminus \{i\}$:
03 $pk \leftarrow (pk_1, \dots, pk_n)$	17 $k_j \leftarrow \mathbf{K}.\text{dec}_j(sk_j, c_j)$
04 $st \leftarrow_{\S} \mathcal{A}_1^{\text{Dec}}(pk)$	18 If $k_j = \perp$: Return \perp
05 $st' \leftarrow (st, pk, sk_1, \dots, sk_{i-1}, sk_{i+1}, \dots, sk_n)$	19 If $c_i \in C_i^*$:
06 Return st'	20 $k_i \leftarrow k_i^*$
Adversary $\mathcal{B}_2^{\text{Dec}}(st', c_i^*, k_i^*)$	21 Else:
07 For $j \in [1..n] \setminus \{i\}$:	22 $k_i \leftarrow \text{Dec}(c_i)$
08 $(k_j^*, c_j^*) \leftarrow_{\S} \mathbf{K}.\text{enc}_j(pk_j)$	23 If $k_i = \perp$: Return \perp
09 $c^* \leftarrow c_1^* \dots c_n^*$	24 $k \leftarrow W(k_1, \dots, k_n, c)$
10 $k^* \leftarrow W(k_1^*, \dots, k_n^*, c^*)$	25 Return k
11 $C^* \leftarrow C^* \cup \{c^*\}$; $C_i^* \leftarrow C_i^* \cup \{c_i^*\}$	
12 $b' \leftarrow_{\S} \mathcal{A}_2^{\text{Dec}}(st, c^*, k^*)$	
13 Stop with b'	

Fig. 8. Adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against session-key indistinguishability of \mathbf{K}_i from adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against session-key indistinguishability of \mathbf{K} .

Game G₂. We add line 10 and line 26. Thus, whenever W is evaluated on a ciphertext whose i th component is c_i^* (that is, either when computing the challenge session key k^* or when answering decapsulation queries involving c_i^* as the i th ciphertext component) the output is overwritten with a uniform value from \mathcal{Y} .

Claim 3. There exists an adversary \mathcal{C} against the split-key pseudorandomness security of W that issues at most $q_d + 1$ evaluation queries such that

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{W,i}^{\text{PR}}(\mathcal{C}),$$

and the running time of \mathcal{C} is roughly the running time of \mathcal{A} .

Proof. We construct an adversary \mathcal{C} that breaks the split-key pseudorandomness of W on the i th key if \mathcal{A} distinguishes between games G_1 and G_2 .

Adversary \mathcal{C} runs K.gen_j for all $j \in [1..n]$ to instantiate all KEMs (see lines 01–03). Then for each KEM K_j it generates a pair key-ciphertext (k_j^*, c_j^*) (lines 05 and 06). All ciphertexts, and all the keys k_j^* for $j \neq i$, are collected and used as input for a call to Eval to generate \mathcal{A}_2 's challenge (lines 07–09). To answer the decapsulation queries of \mathcal{A} on input $c_1..c_n$, the adversary keeps track of previous decapsulation queries and returns the same result for two queries with the same input (line 14). \mathcal{C} uses the secret keys it generated to decapsulate all ciphertext components c_j for $j \neq i$ (lines 16–18). The same procedure is used to decapsulate c_i if $c_i \neq c_i^*$; otherwise it queries its own decapsulation oracle (lines 19–25).

ANALYSIS. First we note that by the conditions in lines 13 and 14 in Fig. 9 all calls to Eval by \mathcal{C}^b have different input and thus we can always use Eval to simulate W .

Adversary $\mathcal{C}^{\text{Eval}}$	If \mathcal{A} calls $\text{Dec}(c)$:
00 $C^*, C_i^* \leftarrow \emptyset; L[\cdot] \leftarrow \perp$	13 If $c \in C^*$: Abort
01 For $j \leftarrow 1$ to n :	14 If $L[c] \neq \perp$: Return $L[c]$
02 $(pk_j, sk_j) \leftarrow_{\S} \text{K.gen}_j$	15 $(c_1, \dots, c_n) \leftarrow c$
03 $pk \leftarrow (pk_1, \dots, pk_n)$	16 For $j \in [1..n] \setminus \{i\}$:
04 $st \leftarrow_{\S} \mathcal{A}_1^{\text{Dec}}(pk)$	17 $k_j \leftarrow \text{K.dec}_j(sk_j, c_j)$
05 For $j \leftarrow 1$ to n :	18 If $k_j = \perp$: Return \perp
06 $(k_j^*, c_j^*) \leftarrow_{\S} \text{K.enc}_j(pk_j)$	19 If $c_i \in C_i^*$:
07 $k'^* \leftarrow k_1^* .. k_{i-1}^* k_{i+1}^* .. k_n^*$	20 $k' \leftarrow k_1 .. k_{i-1} k_{i+1} .. k_n$
08 $c^* \leftarrow (c_1^*, \dots, c_n^*)$	21 $L[c] \leftarrow \text{Eval}(k', c)$
09 $k^* \leftarrow \text{Eval}(k'^*, c^*)$	22 Else:
10 $C^* \leftarrow C^* \cup \{c^*\}; C_i^* \leftarrow C_i^* \cup \{c_i^*\}$	23 $k_i \leftarrow \text{K.dec}_i(sk_i, c_i)$
11 $b' \leftarrow_{\S} \mathcal{A}_2^{\text{Dec}}(st, c^*, k^*)$	24 If $k_i = \perp$: Return \perp
12 Stop with b'	25 $L[c] \leftarrow W(k_1 .. k_i .. k_n, c)$
	26 Return $L[c]$

Fig. 9. Adversary \mathcal{C} against multi-key pseudorandomness of F .

Observe that when \mathcal{C} plays against PR_i^0 we are implicitly setting k_i^* as the key internally generated by PR_i^0 . Hence \mathcal{C} correctly simulates game G_1 to \mathcal{A} . Otherwise when \mathcal{C} plays against PR_i^1 the oracle Eval consistently outputs random elements in \mathcal{K} . Thus \mathcal{C} correctly simulates game G_2 to \mathcal{A} .

Therefore

$$\Pr[G_1 \Rightarrow 1] = \Pr[\text{PR}_i^0 \Rightarrow 1]$$

and

$$\Pr[G_2 \Rightarrow 1] = \Pr[\text{PR}_i^1 \Rightarrow 1].$$

Thus

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{W,i}^{\text{PR}}(\mathcal{C}).$$

We count the number of Eval queries by \mathcal{C} . From the definition of \mathcal{C} we see that the oracle Eval is called once to generate the challenge. Further, for each Dec query by \mathcal{A} , \mathcal{C} queries Eval at most once. \square

Game G_3 . We remove lines 26 to undo the modifications of the Dec oracle introduced in game G_2 . Thus, during decapsulation, whenever W is evaluated on a ciphertext whose i th component is c_i^* the output is computed evaluating the function W on the decapsulated keys instead of returning a uniform input.

Claim 4. There exists an adversary \mathcal{C}' against the split-key pseudorandomness security of W that issues at most q_d evaluation queries such that

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \text{Adv}_{W,i}^{\text{PR}}(\mathcal{C}'),$$

and the running time of \mathcal{C}' is roughly the running time of \mathcal{A} .

Proof. Adversary \mathcal{C}' is essentially the same as adversary \mathcal{C} in Fig. 9, with the exception that we replace line 09 with the generation of a uniform session key ($k^* \leftarrow_{\mathcal{S}} \mathcal{K}$). The proof analysis is the same as in Claim 3. Notice that since this time the challenge session key is uniform, \mathcal{C}' calls Eval just q_d times instead of $q_d + 1$. \square

Note that, currently, the only difference from game G_1 is the addition of line 10, i.e., the challenge session key k^* is uniform.

Game G_4 . Line 07 is removed to undo the modification introduced in game G_1 . That is, we replace the uniform key k_i^* with a real key output by $\text{K.enc}_i(pk_i)$.

Claim 5. There exists an adversary $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ against the session-key indistinguishability of K_i that issues at most q_d decapsulation queries such that

$$|\Pr[G_3 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \text{Adv}_{\text{K}_i}^{\text{kind}}(\mathcal{B}'),$$

and the running time of \mathcal{B}' is roughly the running time of \mathcal{A} .

Proof. Adversary \mathcal{B}' is the same as adversary \mathcal{B} in Fig. 8, with the exception that we replace line 10 with the generation of a uniform session key ($k^* \leftarrow_{\mathfrak{s}} \mathcal{K}$). The proof analysis is the same as in Claim 2. \square

Claim 6. $\Pr[G_4 \Rightarrow 1] = \Pr[\text{KIND}^1 \Rightarrow 1]$.

This follows immediately from the correctness of K_i and the fact that the decapsulation algorithm is deterministic.

The proof of the main statement follows from collecting the statements from Claims 1 to 6. \square

4 Split-Key PRFs in Idealized Models

In the previous section we have shown that if the core function of the parallel combiner is split-key pseudorandom, then said combiner preserves CCA security of any of its ingredient KEMs. It remains to present explicit, practical constructions of skPRFs.

In our first approach we proceed as follows: Given some keys k_1, \dots, k_n and some input x , we mingle together the keys to build a new key k for some (single-key) pseudorandom function F . The output of our candidate skPRF is obtained evaluating $F(k, x)$. In this section we consider variations on how to compute the PRF key k , along with formal proofs for the security of the corresponding candidate skPRFs.

Considering our parallel combiner with such skPRF, evaluating a session key becomes relatively efficient compared to the unavoidable cost of running n distinct encapsulations. Alas, the security of the constructions in this section necessitates some idealized building block, that is, a random oracle or an ideal cipher.

We attempt to abate this drawback by analyzing the following construction from different angles:

$$W(k_1, \dots, k_n, x) := F(\pi(k_n, \pi(\dots \pi(k_1, 0) \dots)), x), \quad (3)$$

where F is a pseudorandom function and π is a pseudorandom permutation. Specifically, we show that W is an skPRF if π is modeled as an ideal cipher (Lemma 5) or F is modeled as a random oracle (Lemma 6 in combination with Example 2).

This statement might be interesting in practice: When implementing such construction the real world, F could reasonably be fixed to SHA-2 (prepending the key), while AES could reasonably be chosen as π . Both primitives are believed to possess good cryptographic properties, arguably so to behave as idealized primitives. Moreover, there is no indication to assume that if one primitive failed to behave ‘ideally’, then the other would be confronted with the same problem.

In Sect. 4.1 we prove that the construction above is secure in the ideal cipher model. In Sect. 4.2 we give some secure constructions in the case that F is modeled as a random oracle.

4.1 Split-Key PRFs in the Ideal Cipher Model

Here we consider constructions of skPRFs where the key-mixing step is conducted in the ideal cipher model followed by a (standard model) PRF evaluation.

Before stating the main result of this section we introduce two additional security notions for keyed functions. The first one is a natural extension of pseudorandomness, whereby an adversary is given access to *multiple instances* of a keyed function (under uniform keys) or truly random functions.

Multi-instance pseudorandomness. See Fig. 10 for the security game that defines the multi-instance pseudorandomness of F . For any adversary \mathcal{A} and number of instances n we define its advantage $\text{Adv}_{F,n}^{\text{mipr}}(\mathcal{A}) := |\Pr[\text{MIPR}^0(\mathcal{A}) \Rightarrow 1] - \Pr[\text{MIPR}^1(\mathcal{A}) \Rightarrow 1]|$. Intuitively, F is *multi-instance pseudorandom* if all practical adversary achieve a negligible advantage.

Game	$\text{MIPR}^b(\mathcal{A})$	Oracle	$\text{Eval}(i, x)$
00	$X_1, \dots, X_n \leftarrow \emptyset$	04	If $x \in X_i$: Abort
01	$k_1, \dots, k_n \leftarrow_{\$} \mathcal{K}$	05	$X_i \leftarrow X_i \cup \{x\}$
02	$b' \leftarrow_{\$} \mathcal{A}^{\text{Eval}}$	06	$y \leftarrow F(k_i, x)$
03	Stop with b'	07	$y^0 \leftarrow y; y^1 \leftarrow_{\$} \mathcal{Y}$
		08	Return y^b

Fig. 10. Security experiments MIPR^b , $b \in \{0, 1\}$, modeling multi-instance pseudorandomness of F for n instances.

While one usually considers indistinguishability between outputs of a pseudorandom functions and uniform elements, key inextractability requires instead that the PRF key be hidden from any efficient adversary. We give a formalization of the latter property in the multi-instance setting next.

Multi-instance key inextractability. Next we introduce multi-instance key inextractability for a keyed function F . To this end, consider the game MIKI given in Fig. 11. To any adversary \mathcal{A} and any number of instances n we associate its advantage $\text{Adv}_{F,n}^{\text{miki}}(\mathcal{A}) := \Pr[\text{MIKI}(\mathcal{A}) \Rightarrow 1]$. Intuitively, F satisfies *multi-instance key inextractability* if all practical adversaries achieve a negligible advantage.

Lemma 5. *Let \mathcal{K} , \mathcal{H} and \mathcal{Y} be finite sets, \mathcal{X} be a set and n a positive integer. Let $F: \mathcal{H} \times \mathcal{X} \rightarrow \mathcal{Y}$, $E: \mathcal{K} \times \mathcal{H} \rightarrow \mathcal{H}$, and $D: \mathcal{K} \times \mathcal{H} \rightarrow \mathcal{H}$ be functions such that for all $k \in \mathcal{K}$ the function $E(k, \cdot)$ is invertible with inverse $D(k, \cdot)$. Consider the function W defined by:*

$$W: \mathcal{K}^n \times \mathcal{X} \rightarrow \mathcal{Y}, \quad W(k_1, \dots, k_n, x) := F(E(k_n, E(\dots E(k_1, 0) \dots)), x).$$

If the function F is pseudorandom then the function W is split-key pseudorandom in the ideal cipher model.

Game	MIKI(\mathcal{A})	Oracle	Eval(i, x)	Oracle	Check(k)
00	$k_1, \dots, k_n \leftarrow_{\$} \mathcal{K}$	03	$y \leftarrow F(k_i, x)$	05	If $k \in \{k_1, \dots, k_n\}$:
01	Run $\mathcal{A}^{\text{Eval}, \text{Check}}$	04	Return y	06	Stop with 1
02	Stop with 0				

Fig. 11. Security experiment MIKI modeling multi-instance key inextractability of F for n instances.

More precisely, suppose that E is modeled as an ideal cipher with inverse D . Then for any $i \in [1..n]$ and for any adversary \mathcal{A} against the split-key pseudorandomness of W there exists an adversary \mathcal{B} against the multi-instance key inextractability of F and an adversary \mathcal{C} against the multi-instance pseudorandomness of F such that:

$$\text{Adv}_{W,i}^{\text{pr}}(\mathcal{A}) \leq \frac{Q + nq_e}{|\mathcal{K}| - n} + 6 \cdot \frac{(Q + 2nq_e)^2}{|\mathcal{H}| - 2Q - 2nq_e} + \text{Adv}_{F,q_e}^{\text{miki}}(\mathcal{B}) + \text{Adv}_{F,q_e}^{\text{mipr}}(\mathcal{C}),$$

where q_e (resp. Q) is the maximum number of calls by \mathcal{A} to the oracle Eval (resp. to the ideal cipher or its inverse). Moreover, \mathcal{B} calls at most q_e (resp. $2Q + nq_e$) times the oracle Eval (resp. Check), and \mathcal{C} calls at most q_e times the oracle Eval. The running times of \mathcal{B} and \mathcal{C} are roughly the same as that of \mathcal{A} .

PROOF SKETCH. The proof consists of a sequence of games interpolating between the games PR_i^0 and PR_i^1 for any $i \in [1..n]$. Our final goal is to make the PRF keys used in Eval as input to F uniform, and then employ the PRF security of F . To achieve this we show that, except with a small probability, the adversary cannot manipulate the game to use anything but independent, uniformly generated values as key input to F .

The PRF keys are sequences of the form $h = E(k_n, E(\dots E(k_1, 0) \dots))$ for some keys $k_1..k_n$. We fix an index i : The key k_i is uniformly generated by the pseudorandomness game, and the remaining keys are chosen by the adversary on each query to Eval. The proof can be conceptually divided into two parts. Initially (games G_0 – G_3) we work on the first part of the sequence, namely $h' = E(k_i, E(\dots E(k_1, 0) \dots))$. Here we build towards a game in which all elements h' that are generated from different key vectors $k_1..k_{i-1}$ are independent uniform values. In the next games (games G_4 – G_9) we work on the second part of the sequence, namely $h = E(k_n, E(\dots E(k_{i+1}, h') \dots))$. Again, we show that all elements h are independent and uniform, assuming independent uniform values h' .

We describe now each single game hop. We start from game G_0 , equivalent to the real game PR_i^0 , and we proceed as follows. Game G_1 aborts if the key k_i is directly used as input by the adversary in one of its oracle queries. In game G_2 the output of E under the uniform key k_i is precomputed and stored in a list R , which is then used by Eval. Game G_3 aborts when, in a query to Eval, the adversary triggers an evaluation of $E(k_{i-1}, E(\dots E(k_1, 0) \dots))$ that gives the same output as one of a previous evaluations using a different key vector. At this point we want

to argue that an adversary sequentially evaluating $n - i$ times the ideal cipher under known keys but uniform initial input still cannot obtain anything but a (n almost) uniform output. This will be achieved by uniformly pre-generating the enciphering output used to evaluate the sequences $E(k_n, E(\dots E(k_{i+1}, h') \dots))$. These elements are precomputed in game G_4 and stored in a list R , but not yet used. In game G_5 the elements stored in R are removed from the range of the ideal cipher. In game G_6 , the oracle Eval uses the values in R to sample the ideal cipher. Since this might not always be possible, the oracle Eval resumes standard sampling if any value to be sampled has already been set in E or D. The next game makes a step forward to guarantee that the previous condition does not occur: If the two oracles E and D have never been queried with input any value that is used as key to the PRF F , then the game aborts if any element stored in R (but not used as a PRF key) is queried to E or D. All previous steps have only involved information-theoretical arguments. In game G_8 we disjoin our simulated ideal cipher from the PRF keys. This requires many small changes to the game structure, but eventually the price paid to switch from game G_7 is the advantage in breaking multi-instance key inextractability of the PRF, i.e., to recover one of the PRF keys from the PRF output. At this point, for any fixed input $k' = k_1 \dots k_{i-1} k_{i+1} \dots k_n$ to Eval we are sampling independent, uniformly generated elements to be used as the PRF keys. Finally endowed with uniform keys, in G_9 the PRF output is replaced with uniform values. If no abort condition is triggered, then the output distributions of G_9 and PR_i^1 are identical.

The complete proof can be found in the full version of the paper [13].

4.2 Split-Key PRFs in the Random Oracle Model

Next, we consider constructions of skPRFs where the key-mixing step employs standard model primitives. However, to achieve security we idealize the PRF that is employed afterwards. Here we identify a sufficient condition on the key-mixing function such that the overall construction achieves split-key pseudorandomness. We begin by giving the aforementioned property for the key-mixing function.

Almost uniformity of a key-mixing function. For all $i \in [1 \dots n]$ let \mathcal{K}_i be a finite key space and \mathcal{K} any key space. Consider a function

$$g: \mathcal{K}_1 \times \dots \times \mathcal{K}_n \rightarrow \mathcal{K}.$$

We say that g is ϵ -almost uniform w.r.t. the i th key if for all $k \in \mathcal{K}$ and all $k_j \in \mathcal{K}_j$ for $j \in [1 \dots n] \setminus \{i\}$ we have:

$$\Pr_{k_i \leftarrow \mathcal{K}_i} [g(k_1 \dots k_n) = k] \leq \epsilon.$$

We say that g is ϵ -almost uniform if it is ϵ -almost uniform w.r.t. the i th key for all $i \in [1 \dots n]$.

We give three standard model instantiations of key-mixing functions that enjoy almost uniformity.

Example 1. Let $\mathcal{K}_1 = \dots = \mathcal{K}_n = \mathcal{K} = \{0, 1\}^k$ for some $k \in \mathbb{N}$ and define

$$g_{\oplus}(k_1 .. k_n) := \bigoplus_{j=1}^n k_j.$$

Then g_{\oplus} is $1/|\mathcal{K}|$ -almost uniform.

The proof follows from observing that for any $i \in [1..n]$ and any fixed $k_1 .. k_{i-1}k_{i+1} .. k_n$, the function $g_{\oplus}(k_1 .. k_{i-1} \cdot k_{i+1} .. k_n)$ is a permutation.

Example 2. Let \mathcal{K}, \mathcal{H} be finite and $\pi: \mathcal{K} \times \mathcal{H} \rightarrow \mathcal{H}$ such that for all $k \in \mathcal{K}$ we have that $\pi(k, \cdot)$ is a permutation on \mathcal{H} . Let

$$g(k_1 .. k_n) := \pi(k_n, \dots \pi(k_1, 0) \dots),$$

for some $0 \in \mathcal{K}$.

If for all $k \in \mathcal{K}$, $\pi(k, \cdot)$ is a pseudorandom permutation (i.e., π is a blockcipher) then for all i and all $k_1 .. k_{i-1}k_{i+1} .. k_n$ there exists an adversary \mathcal{A} against the pseudorandomness of π such that g is $\text{Adv}_{\pi}^{\text{PRP}}(\mathcal{A}) + 1/|\mathcal{K}|$ -almost uniform. Here $\text{Adv}_{\pi}^{\text{PRP}}(\mathcal{A})$ is the advantage of \mathcal{A} in distinguishing π under a uniform key from a uniform permutation.

We sketch a proof of Example 2. First, observe that, since k_j for all $j \neq i$ is known by \mathcal{A} , all permutations $\pi(k_j, \cdot)$ can be disregarded. Secondly, we replace the permutation $\pi(k_i, \cdot)$ with a uniform permutation, losing the term $\text{Adv}_{\pi}^{\text{PRP}}(\mathcal{A})$. The claim follows.

Example 3. Let $\mathcal{K}_1, \dots, \mathcal{K}_n, \mathcal{K}$ be finite. Let

$$g(k_1 .. k_n) := k_1 \parallel \dots \parallel k_n,$$

then g is $1/|\mathcal{K}|$ -almost uniform.

The proof uses the same argument as in Example 1.

We now show that we can generically construct a pseudorandom skPRF from any almost-uniform key-mixing function in the random oracle model.

Lemma 6. *Let $g: \mathcal{K}_* \rightarrow \mathcal{K}'$ be a function. Let $H: \mathcal{K}' \times \mathcal{X} \rightarrow \mathcal{Y}$ be a (hash) function. Let*

$$H \diamond g: \mathcal{K}_* \times \mathcal{X} \rightarrow \mathcal{Y}, (H \diamond g)(k_1, \dots, k_n, x) := H(g(k_1 .. k_n), x).$$

If H is modeled as a random oracle then for any adversary \mathcal{A} such that g is ϵ -almost uniform and \mathcal{A} makes at most q_H H queries and q_e Eval queries and all i we have

$$\text{Adv}_i^{\text{PR}}(\mathcal{A}) \leq q_H \cdot \epsilon.$$

PROOF SKETCH. Note that any adversary against the pseudorandomness of $H \diamond g$ is given access to Eval and H , the latter implementing a random oracle. Now, intuitively, \mathcal{A} is unlikely to predict the output of the g invocation within an Eval

Games PR_i^b	Oracle $\text{Eval}(k', x)$	Oracle $\text{H}(k'', x)$
00 $X \leftarrow \emptyset$	07 If $x \in X$: Abort	17 $S_H \leftarrow S_H \cup \{(k'', x)\}$
01 $S_E, S_H \leftarrow \emptyset$	08 $X \leftarrow X \cup \{x\}$	18 If $H[k'', x] = \perp$:
02 $k_i \leftarrow_{\mathcal{S}} \mathcal{K}_i$	09 $k_1 \dots k_{i-1} k_{i+1} \dots k_n \leftarrow k'$	19 $H[k'', x] \leftarrow_{\mathcal{S}} \mathcal{K}'$
03 $b' \leftarrow_{\mathcal{S}} \mathcal{A}^{\text{Eval}, \text{H}}$	10 $k'' \leftarrow g(k_1 \dots k_i \dots k_n)$	20 Return $H[k'', x]$
04 If $S_H \cap S_E \neq \emptyset$:	11 $S_E \leftarrow S_E \cup \{(k'', x)\}$	
05 bad \leftarrow true	12 If $H[k'', x] = \perp$:	
06 Stop with b'	13 $H[k'', x] \leftarrow_{\mathcal{S}} \mathcal{K}'$	
	14 $y \leftarrow H[k'', x]$	
	15 $y^0 \leftarrow y; y^1 \leftarrow_{\mathcal{S}} \mathcal{Y}$	
	16 Return y^b	

Fig. 12. Game PR_i^b for $i \in [1..n]$ instantiated with $H \diamond g$.

query as g is almost uniform. Hence, \mathcal{A} will not query H on the same input as done within Eval . Thus, even in the real game, the output of Eval is likely to be uniform.

We give a refined analysis next.

Proof (Lemma 6). We bound the distance between the probabilities of \mathcal{A} outputting 1 in game PR_i^0 and PR_i^1 . The PR_i^b game is given in Fig. 12. For game PR_i^b we performed merely syntactical changes: \mathcal{A} is given access to H via oracle H . Two sets S_E, S_H are initialized as empty and updated in lines 01, 11, 17 and used to define an event in line 05.

Observe that for all i the PR_i^0 and PR_i^1 games are identical if bad does not happen: As $S_H \cap S_E$ remains empty, adversary \mathcal{A} did not query H on an input that H was evaluated on during an Eval query (see line 14). Hence, $y \leftarrow \text{H}(k'', x)$ is uniform and thus, $y^0 \leftarrow y$ and $y^1 \leftarrow \mathcal{Y}$ are identically distributed.

We bound $\text{Pr}[\text{bad}]$ in PR_i^1 . To this end, let (k''_j, x_j) for $j \in [1..q_H]$ denote the H queries made by \mathcal{A} . We have

$$\text{Pr}[\text{bad}] = \text{Pr}[S_H \cap S_E \neq \emptyset] \leq \sum_{j=1}^{q_H} \text{Pr}[(k''_j, x_j) \in S_E].$$

Recall from line 07 that for every $x \in \mathcal{X}$ there is at most one query $\text{Eval}(\cdot, x)$ by \mathcal{A} . Hence, for each (k''_j, x_j) in S_H there is at most one element of the form (\cdot, x_j) in S_E . Assume it exists⁵ and let k''_{x_j} be such that $(k''_{x_j}, x_j) \in S_E$ denotes that element. Then

$$\begin{aligned} \sum_{j=1}^{q_H} \text{Pr}[(k''_j, x_j) \in S_E] &\leq \sum_{j=1}^{q_H} \text{Pr}[k''_j = k''_{x_j}] \\ &= \sum_{j=1}^{q_H} \text{Pr}_{k_i, x_j \leftarrow \mathcal{K}_i} [k''_j = g(k'_1 \dots k'_{i-1} k'_{i, x_j} k'_{i+1} \dots k'_n)] \end{aligned}$$

⁵ If such an element does not exist the following bounds would only become tighter.

for $k'_1, \dots, k'_{i-1}, k'_{i+1}, \dots, k'_n$ chosen by \mathcal{A} and uniform k_{i,x_j} such that it satisfies $g(k'_1 \dots k'_{i-1} k_{i,x_j} k'_{i+1} \dots k'_n) = k''_{x_j}$. Eventually, we can employ the ϵ -almost uniformity of g to conclude that

$$\sum_{j=1}^{q_H} \Pr_{k_{i,x_j} \leftarrow \mathcal{K}_i} [k''_j = g(k'_1 \dots k'_{i-1} k_{i,x_j} k'_{i+1} \dots k'_n)] \leq \sum_{j=1}^{q_H} \epsilon \leq q_H \cdot \epsilon.$$

□

Next, we show that, generally, the construction from Lemma 6 does not yield a split-key pseudorandom function in the standard model.

Lemma 7. *Let g be with syntax as in Lemma 6 and let F be with syntax as H in Lemma 6. There exists an instantiation of g and F such that g is almost uniform and F is pseudorandom but*

$$F \diamond g: \mathcal{K}_* \times \mathcal{X} \rightarrow \mathcal{Y}, \quad (F \diamond g)(k_1, \dots, k_n, x) := F(g(k_1 \dots k_n), x)$$

is not a pseudorandom skPRF.

Proof. We saw in Example 1 that g_{\oplus} is almost uniform. Further, we saw in Lemma 3 that, when using $F \diamond g_{\oplus}$ as a core function, there exists a pseudorandom function F such that the combined KEM is not CCA secure. If $F \diamond g_{\oplus}$ (with such F) were split-key pseudorandom, then this would contradict Theorem 1.

5 A KEM Combiner in the Standard Model

Our approach was hitherto to mix the keys k_1, \dots, k_n to obtain a key for a PRF, which was then evaluated on the ciphertext vector. The drawback of this is that to show security we had to turn to idealized primitives. In the following we embark on a different approach, with the goal to obtain a standard model construction.

5.1 The PRF-Then-XOR Split-Key PRF

Here we abstain from mixing the keys together, but use each key k_i in a PRF evaluation. The security of the model is offset by its price in terms of efficiency: When employed in a parallel combiner, the skPRF requires n PRF calls, whereas for our constructions secure in idealized models in Sect. 4.2 a single call to a PRF suffices. We give our construction next.

As before we want to allow possibly different session-key spaces of the ingredient KEMs. Thus, as the keys k_i in Construction 2 come from an encapsulation of \mathcal{K}_i , we allow the construction to use distinct PRFs. Yet, one may choose $F_i = F_j$ for all i, j , if supported by the ingredient KEM's syntax.

Construction 2. For all $i \in [1..n]$ let $F_i: \mathcal{K}_i \times \mathcal{X} \rightarrow \mathcal{Y}$ be a function and let $\mathcal{K} = \mathcal{K}_1 \times \dots \times \mathcal{K}_n$. We define the PRF-then-XOR composition of F_1, \dots, F_n :

$$[F_1 .. F_n]: \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}, \quad [F_1 .. F_n](k_1, \dots, k_n, x) := \bigoplus_{i=1}^n F_i(k_i, x).$$

Lemma 8. For all $i \in [1..n]$ let F_i be as in Construction 2. If all F_i are pseudorandom then $[F_1 .. F_n]$ is split-key pseudorandom.

More precisely, for all n, F_1, \dots, F_n , for all indices i and all adversaries \mathcal{A} there exist an adversary \mathcal{B} such that

$$\text{Adv}_{[F_1 .. F_n], i}^{\text{PR}}(\mathcal{A}) \leq \text{Adv}_{F_i}^{\text{PR}}(\mathcal{B}).$$

Suppose that \mathcal{A} poses at most q queries to its evaluation oracle. Then adversary \mathcal{B} poses at most q queries to its own encapsulation oracle. The running times of \mathcal{B} is roughly the same as of \mathcal{A} .

Proof. We fix an index $i \in [1..n]$ and we build an adversary \mathcal{B} against the PRF F_i from an adversary \mathcal{A} against the skPRF $[F_1 .. F_n]$.

Adversary \mathcal{B} works as follows. It starts by running adversary \mathcal{A} . Each time that \mathcal{A} queries the oracle Eval on input (k', x) it queries its own evaluation oracle on input x , obtaining the output $y \in \mathcal{Y}$. Then it computes the key $k := y \oplus \bigoplus_{j \neq i} F_j(k_j, x)$, and returns the key to \mathcal{A} . Finally, \mathcal{B} returns the output of \mathcal{A} .

We observe that if \mathcal{B} is playing against game PR^0 then it receives a real evaluation of F_i from the oracle Eval. Hence \mathcal{B} returns to \mathcal{A} a real key and \mathcal{A} is playing against game PR_i^0 . If \mathcal{B} is playing against game PR^1 instead, then \mathcal{B} receives independent, uniformly distributed values from the oracle Eval (note that, by the restrictions of game PR_i^1 , adversary \mathcal{A} queries its oracle on distinct input each time). If we add any constant value to $y \leftarrow_{\$} \mathcal{Y}$ the result remains uniformly distributed. Hence, on each query to Eval adversary \mathcal{B} returns to \mathcal{A} independent uniformly distributed keys and \mathcal{A} is playing against game PR_i^1 . \square

Note that Lemma 8 gives raise to a standard model KEM combiner that requires n PRF invocations, each processing the concatenation of n encapsulations $c = c_1 || \dots || c_n$. For a slightly more efficient combiner where each of the n PRF invocations is evaluated on the concatenation of $n - 1$ encapsulations see the full version of this paper [13].

Acknowledgments. We are grateful to the anonymous PKC reviewers for their valuable comments.

Federico Giacon was supported by ERC Project ERCC (FP7/615074). Felix Heuer was supported by Mercator Research Center Ruhr project “LPN-Krypt: Das LPN-Problem in der Kryptographie”. Bertram Poettering conducted part of this work at Ruhr University Bochum, supported by ERC Project ERCC (FP7/615074).

References

1. Ananth, P., Jain, A., Naor, M., Sahai, A., Yorgev, E.: Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 491–520. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_17
2. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015, pp. 553–570. IEEE Computer Society Press (2015)
3. Brzuska, C., Farshim, P., Mittelbach, A.: Random-oracle uninstantiability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 428–455. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_17
4. 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). <https://doi.org/10.1137/S0097539702403773>
5. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977). <https://doi.org/10.1109/C-M.1977.217750>
6. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_11
7. Even, S., Goldreich, O.: On the power of cascade ciphers. ACM Trans. Comput. Syst. **3**(2), 108–116 (1985). <http://doi.acm.org/10.1145/214438.214442>
8. Fischlin, M., Herzberg, A., Bin-Noon, H., Shulman, H.: Obfuscation combiners. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 521–550. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_18
9. Fischlin, M., Lehmann, A.: Security-amplifying combiners for collision-resistant hash functions. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 224–243. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_13
10. Fischlin, M., Lehmann, A.: Multi-property preserving combiners for hash functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 375–392. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_21
11. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust multi-property combiners for hash functions revisited. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 655–666. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_53
12. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptol. **26**(1), 80–101 (2013)
13. Giacon, F., Heuer, F., Poettering, B.: KEM combiners. Cryptology ePrint Archive, Report 2018/024 (2018). <https://eprint.iacr.org/2018/024>
14. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_6
15. Herzberg, A.: On tolerant cryptographic constructions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 172–190. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_13

16. 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
17. Hohenberger, S., Lewko, A., Waters, B.: Detecting dangerous queries: a new approach for chosen ciphertext security. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 663–681. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_39
18. Manulis, M., Poettering, B., Stebila, D.: Plaintext awareness in identity-based key encapsulation. *Int. J. Inf. Secur.* **13**(1), 25–49 (2014). <https://doi.org/10.1007/s10207-013-0218-5>
19. Merkle, R.C., Hellman, M.E.: On the security of multiple encryption. *Commun. ACM* **24**(7), 465–467 (1981). <http://doi.acm.org/10.1145/358699.358718>
20. NIST: Post-Quantum Cryptography Standardization Project (2017). <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography>
21. Shannon, C.: Communication theory of secrecy systems. *Bell Syst. Tech. J.* **28**, 656–715 (1949)
22. Zhang, C., Cash, D., Wang, X., Yu, X., Chow, S.S.M.: Combiners for chosen-ciphertext security. In: Dinh, T.N., Thai, M.T. (eds.) COCOON 2016. LNCS, vol. 9797, pp. 257–268. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42634-1_21
23. Zhang, R., Hanaoka, G., Shikata, J., Imai, H.: On the security of multiple encryption or CCA-security+CCA-security=CCA-security? In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 360–374. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24632-9_26



Revisiting Proxy Re-encryption: Forward Secrecy, Improved Security, and Applications

David Derler¹, Stephan Krenn², Thomas Lorüenser², Sebastian Ramacher^{1(✉)}, Daniel Slamanig², and Christoph Striecks²

¹ IAIK, Graz University of Technology, Graz, Austria
{david.derler,sebastian.ramacher}@tugraz.at

² AIT Austrian Institute of Technology, Vienna, Austria
{stephan.krenn,thomas.loruenser,daniel.slamanig, christoph.striecks}@ait.ac.at

Abstract. We revisit the notion of proxy re-encryption (PRE), an enhanced public-key encryption primitive envisioned by Blaze et al. (EUROCRYPT'98) and formalized by Ateniese et al. (NDSS'05) for delegating decryption rights from a delegator to a delegatee using a semi-trusted proxy. PRE notably allows to craft re-encryption keys in order to equip the proxy with the power of transforming ciphertexts under a delegator's public key to ciphertexts under a delegatee's public key, while not learning anything about the underlying plaintexts.

We study an attractive cryptographic property for PRE, namely that of forward secrecy. In our forward-secret PRE (fs-PRE) definition, the proxy periodically evolves the re-encryption keys and permanently erases old versions while the delegator's public key is kept constant. As a consequence, ciphertexts for old periods are no longer re-encryptable and, in particular, cannot be decrypted anymore at the delegatee's end. Moreover, delegators evolve their secret keys too, and, thus, not even they can decrypt old ciphertexts once their key material from past periods has been deleted. This, as we will discuss, directly has application in short-term data/message-sharing scenarios.

Technically, we formalize fs-PRE. Thereby, we identify a subtle but significant gap in the well-established security model for conventional PRE and close it with our formalization (which we dub fs-PRE⁺). We present the first provably secure and efficient constructions of fs-PRE as well as PRE (implied by the former) satisfying the strong fs-PRE⁺ and PRE⁺ notions, respectively. All our constructions are instantiable in the standard model under standard assumptions and our central building block are hierarchical identity-based encryption (HIBE) schemes that only need to be selectively secure.

Keywords: Forward secrecy · Proxy re-encryption
Improved security model

1 Introduction

The security of cryptosystems essentially relies on the secrecy of the respective secret key. For example, if for an encryption scheme a secret key is (accidentally) leaked, the confidentiality of all the data encrypted with respect to this key so far is immediately destroyed. One simple mitigation strategy for such a secret-key leakage is to frequently change secret keys such that leaking a secret key only affects a small amount of data. Implementing this in a naïve way, for instance in context of public-key encryption, means that one either has to securely and interactively distribute copies of new public keys frequently or to have huge public keys¹, which is rather inconvenient in practice. Consequently, cryptographic research focused on the design of cryptosystems that inherently provide such a property, being denoted as *forward secrecy* (or, *forward security*) [28]. The goal hereby is that key leakage at some point in time does not affect the data which was encrypted before the key leakage, while mitigating the drawbacks of the naïve solution discussed before. That is, one aims at efficient non-interactive solutions that have fixed sublinear-size public keys in the number of key switches/time periods. Those (strong) properties are the minimal requirements in the de-facto standard notion of forward secrecy in the cryptographic literature.

Within the last two decades, forward secrecy has been identified as an important property of various different cryptographic primitives such as digital signatures [6], identification schemes [1], public-key encryption [15], and private-key cryptography [7]. Only recently, another huge step forward has been made by Green and Miers [27] as well as Günther et al. [29] to bring forward secrecy to important practical applications in the context of asynchronous messaging and zero round-trip time (0-RTT) key exchange. Given revelations and leaks about large-scale surveillance activities of security agencies within the last years, it is of utmost importance to further develop and deploy cryptosystems that inherently provide forward secrecy. We aim at advancing the research on forward secrecy with respect to other practically important public-key primitives, ideally, to ones with slightly more functionality.

Proxy re-encryption. Proxy re-encryption (PRE), envisioned by Blaze et al. [9] and formalized by Ateniese et al. [4, 5], is a cryptographic primitive that can be seen as an extension of public-key encryption. A central feature of PRE is that senders can craft so-called re-encryption keys, which are usually created using only public information of the designated delegatee and the delegators' key material. Those re-encryption keys have the power to transform ciphertexts under a delegator's public key to ciphertexts under the delegates' public keys. Within PRE, this transformation is done by a semi-trusted² proxy. The widely accepted model for PRE security (i.e., the conventional or plain PRE model) [4]

¹ With size $O(n)$ for n key switches/time periods.

² A semi-trusted proxy honestly follows the protocols, i.e., stores consistent re-encryption keys and re-encrypts correctly.

requires that the proxy does not learn anything about the plaintexts which underlie the ciphertexts to be transformed.³

Proxy re-encryption is considered very useful in applications such as encrypted e-mail forwarding or access control in secure file systems, which was already discussed heavily in earlier work, e.g., in [4]. Furthermore, PRE has been object of significant research for almost two decades now, be it in a conventional setting [4, 5, 9], PRE with temporary delegation [4, 5, 34], identity-based PRE [26, 37], extensions to the chosen-ciphertext setting [16, 34], type-based/conditional PRE [39, 41], anonymous (or key-private) PRE [3], traceable PRE [32], or PRE from lattice-based assumptions [18, 36]. Generic constructions of PRE schemes from fully-homomorphic encryption [24] and from non-standard building blocks such as resplittable-threshold public key encryption as proposed in [30] are known, where different constructions of secure obfuscators for the re-encryption functionality have been given [18, 19, 31]. Despite PRE being an object of such significant research, forward-secret constructions remain unknown.⁴

On modeling forward-secret proxy re-encryption. Forward secrecy in the context of PRE is more complex than in standard public-key primitives, as PRE involves multiple different parties (i.e., delegator, proxy, and delegates), where delegator and delegates all have their own secret-key material and the proxy additionally holds all the re-encryption keys. One may observe that the proxy needs to be considered as a semi-trusted (central) party being always online, and, thus, it is reasonable to assume that this party is most valuable to attack. Consequently, we model forward secrecy in the sense that the re-encryption-key material can be evolved by the proxy to new periods while past-period re-encryption keys are securely erased. Hence, ciphertexts under the delegator’s public key with respect to past-periods can no longer be re-encrypted. In addition, we model forward secrecy for the delegator’s key material in a way that it is consistent with the evolution of the re-encryption material at the proxy.

For now, we do not consider forward secrecy at the delegatee, who can be seen as a passive party and does not need to take any further interaction with the delegator during the life-time of the system, except providing her public key once after set-up (e.g., via e-mail or public key server). It also does not have to be online when ciphertexts are re-encrypted for her by the proxy. Nevertheless, we leave it as a path for future research to cover the third dimension, i.e., model forward secrecy for the delegator and proxy as well as forward secrecy for the delegatee with efficient non-trivial constructions. However, it seems highly non-trivial to achieve efficient constructions that support forward secrecy for the delegatee additionally. In particular, we believe that the difficulty of achieving such strong type of forward secrecy is due to the circumstance that one has to

³ The well-established security notions for PRE leave a potentially critical gap open. To look ahead, our proposed security model for *forward-secret* PRE closes this gap (implicitly also for plain PRE) and goes even beyond.

⁴ We stress that we only aim at efficient non-trivial (non-interactive) forward-secret PRE constructions that have sublinear-size public and re-encryption keys in the number of time periods.

carefully integrate three dimension of evolving key-material, one at the delegator, one at the proxy, and one at the delegatee. All dimensions seem to interfere with each other.⁵ As it will be confirmed by our application, covering the two dimensions already yields an interesting tool.

Moreover, to achieve forward secrecy for delegator and proxy key material, we face the following obstacles. First, it has to be guaranteed that the honest proxy must not be able to gain any information from the ciphertexts while at the same time being able to transform such ciphertexts *and* to update re-encryption key material consistently to newer time periods *without* any interaction with the delegator. Secondly, any delegatee *must not* be able to decrypt past-period ciphertexts. In this work, we give an affirmative answer to overcome those obstacles.

A practical application of forward-secret PRE. We believe that forward secrecy is an essential topic nowadays for any application. Also PRE is increasingly popular, be it in applied cryptographic literature [10, 14, 35, 36, 42], working groups such as the CFRG of the IRTF⁶, large-scale EU-funded projects⁷, and meanwhile also companies⁸ that foster transition of such technologies into applications.

A practical application for forward-secret PRE is disappearing 1-to- n messaging. Here, a user encrypts a message under his public key and sends it to the proxy server that is responsible for distributing the encrypted messages to all pre-determined n receivers (note that receivers do not have to be online at the time the encrypted message is sent and an initial public-key exchange has to be done only in the beginning, but no more interactivity is needed). During setup time, the user has equipped the server with re-encryption keys (one for each receiver) while new keys can be added any time once a new receiver is present. Furthermore, the user does not need to manage a potentially huge list of public keys for each message to be sent. After a period, the data gets deleted by the proxy server, the re-encryption keys get evolved to a new period (without any interactions), and old-period re-encryption keys get deleted. The security of forward-secret PRE then guarantees that the proxy server does not learn the sensitive messages, neither can the two types of parties access disappeared messages later on. Once period- i re-encryption keys leak from the proxy server, only present and future encrypted messages (from period i onward) are compromised, while period- $(i - 1)$ messages stay confidential. More generally, we believe that forward-secret PRE can be beneficially used in all kinds of settings that require access revocation, e.g., in outsourced encrypted data storage.

We also stress that within our forward-secret PRE instantiations, each user is only required to manage her own public and secret keys on her device and not a

⁵ It is currently unknown to us how to solve the problem with efficient cryptographic tools, e.g., in the bilinear-maps setting. For efficiency reasons, multilinear maps and obfuscation are out of focus.

⁶ <https://www.ietf.org/id/draft-hallambaker-mesh-recrypt-00.txt>.

⁷ <https://credential.eu/>.

⁸ e.g., <http://www.nucypher.com>, <https://besafe.io/>.

list of recipient public keys (or, identities). This deviates significantly from other primitives such as broadcast encryption (BE) [12, 22, 38], which could also be suitable in such scenarios. However, practical BE schemes, e.g., [13], need large public keys and are computationally expensive.

1.1 Contribution

In this paper, we investigate forward secrecy in the field of proxy re-encryption (PRE) and term it fs-PRE. More precisely, our contributions are as follows:

- We first port the security model of PRE to the forward-secret setting (fs-PRE[−]). Thereby, we observe a subtle but significant gap in existing (plain) security models for conventional PRE with regard to the granularity of delegations of decryption rights. In particular, existing models allow that a recipient, who has once decrypted a re-encrypted ciphertext, can potentially decrypt all re-encryptable ciphertexts of the same sender without further involvement of the proxy. In the forward-secret setting, it would essentially require to trust the delegates to delete their re-encrypted ciphertexts whenever the period is switched, which is a problematic trust assumption.⁹
- We close this gap by introducing an additional security notion which inherently requires the involvement of a proxy in every re-encryption and in particular consider this notion in the forward-secret setting (fs-PRE⁺). We also note that, when considering only a single time interval, this implicitly closes the aforementioned gap in the conventional PRE setting.¹⁰ We also provide an explicit separation of the weaker fs-PRE[−] notion (resembling existing PRE models) and our stronger notion fs-PRE⁺.
- We then continue by constructing the first forward-secret PRE schemes (in the weaker as well as our stronger model) that are secure in the standard model under standard assumptions. On a technical side, only few approaches to forward secrecy are known. Exemplary, in the public-key-encryption (PKE) setting, we essentially have two ways to construct forward secrecy, i.e., the Canetti-Halevi-Katz (CHK) framework [15] from selectively secure hierarchical identity-based encryption (HIBE) [25] schemes and the more abstract puncturable-encryption (PE) approaches by [27, 29] (where both works either explicitly or implicitly use the CHK techniques). Particularly, we are not aware of any framework to achieve forward secrecy for PKE schemes based on “less-complex” primitives in comparison to selectively secure HIBE schemes. Consequently, we also base our constructions on selectively secure HIBE schemes [25], which we combine with linearly homomorphic encryption schemes, e.g., (linear) ElGamal.
- As a side result, we generalize the recent work of PE [17, 21, 27, 29] to what we call fully puncturable encryption (FuPE) in the full version of this paper and show how we can use FuPE to construct fs-PRE.

⁹ Clearly, we still have to trust that the proxy deletes past-period re-encryption key material.

¹⁰ In the conventional PRE setting, this gap was very recently independently addressed by Cohen [20].

1.2 Intuition and Construction Overview

To obtain more general results and potentially also more efficient instantiations, we use a relaxation of HIBEs denoted as binary-tree encryption (BTE) which was introduced by Canetti, Halevi, and Katz (CHK) in [15]. As an intermediate step, we introduce the notion of a forward-secret delegatable public-key encryption (fs-DPKE) scheme and present one instantiation which we obtain by combining the results of CHK with a suitable homomorphic public-key encryption (HPKE) scheme. Loosely speaking, a fs-DPKE scheme allows to delegate the decryption functionality of ciphertexts computed with respect to the public key of some user A to the public key of some other user B . Therefore, A provides a *public* delegation key to B . B then uses the delegation key *together* with the secret key corresponding to B 's public key to decrypt any ciphertext that has been produced for A . A fs-DPKE scheme moreover incorporates forward secrecy in a sense that the originator A can evolve its secret key and the scheme additionally allows to *publicly* evolve delegation keys accordingly. Interestingly, such a scheme is already sufficient to construct a fs-PRE⁻-secure PRE scheme. Finally, we demonstrate how to strengthen this construction to a fs-PRE⁺-secure PRE scheme, by solely relying on a certain type of key-homomorphism of the underlying fs-DPKE scheme. The intermediate step of introducing fs-DPKE is straightforward yet interesting, since we believe fs-DPKE is the “next natural step” to lift PKE to a setting which allows for controlled delegation of decryption rights.

Instantiation. In Table 1, we present an instantiation including the resulting key and ciphertext sizes. Thereby, we only look at fs-PRE instantiations that are fs-PRE⁺-secure and note that the asymptotic sizes for fs-PRE⁻-secure fs-PRE schemes are identical. For our instantiation, we use the BTE (or any selectively secure HIBE) from [15] and the linear encryption scheme from [11] as HPKE scheme under the Bilinear Decisional Diffie-Hellman (BDDH) and decision linear (DLIN) assumption respectively.

Table 1. Our fs-PRE⁺-secure instantiation. All parameters additionally scale asymptotically in a security parameter k which is, hence, omitted. Legend: n ... number of periods, $|\text{pk}|$... public key size, $|\text{rk}^{(i)}|$... size of re-encryption key for period i , $|\text{sk}^{(i)}|$... size of secret key for period i , $|C|$... ciphertext size.

Building blocks	$ \text{pk} $	$ \text{rk}^{(i)} $	$ \text{sk}^{(i)} $	$ C $	Assumption
BTE [15], HPKE [11]	$\mathcal{O}(\log n)$	$\mathcal{O}((\log n)^2)$	$\mathcal{O}((\log n)^2)$	$\mathcal{O}(\log n)$	BDDH, DLIN

A note on a side result. Additionally, in the full version, we include the definition and a construction of a so called fully puncturable encryption (FuPE) scheme which is inspired by techniques known from HIBEs and the recent PE schemes in [27, 29]. We then show that FuPE schemes closely capture the essence which is required to construct fs-PRE⁺-secure schemes by presenting a construction of a fs-PRE⁺-secure PRE scheme from FuPE and HPKE.

1.3 Related Work and Outline

Work related to forward-secret PRE. Tang et al. [39,41] introduced type-based/conditional PRE, which allows re-encryption of ciphertexts at the proxy only if a specific condition (e.g., a time period) is satisfied by the ciphertext. Furthermore, PRE with temporary delegations was proposed by Ateniese et al. [4,5] and improved by Libert and Vermaud (LV) [34]. All those approaches yield a weak form of forward secrecy. Notably, the LV schemes provide fixed public parameters and non-interactivity with the delegatee as well. However, in contrast to our approach, LV and Tang et al. require at least to update the re-encryption keys for each time period with the help of the delegator (i.e., one message per time period from the delegator to the proxy) and also do not allow for exponentially many time periods, which do not suit our (stronger) forward-secret scenario.

Concurrent work on PRE. There is a considerable amount of very recent independent and concurrent work on different aspects of PRE and its applications [8,20,23,35]. The works in [8,23,35] are only related in that they also deal with various aspects of PRE, but not fs-PRE. Those aspects are however unrelated to the work presented in this paper. In contrast, the work presented in [20] is related to one aspect of our work. It formalizes a security property for conventional PRE, which can be seen as a special case of our fs-PRE⁺ notion which we introduce in context of fs-PRE. More precisely, our notion generalizes the notion of [20] and implies it if we fix the numbers of time periods to $n = 1$.

Outline. After discussing preliminaries in Section 2, we define fs-PRE in Section 3, discuss the gap in previous models and also briefly discuss its consequences to conventional PRE. We then give the first construction of a fs-PRE scheme from binary tree encryption in Section 4. We also show a separation result for the weaker fs-PRE⁻ (resembling existing PRE models) and our stronger notion fs-PRE⁺.

2 Preliminaries

For $n \in \mathbb{N}$, let $[n] := \{1, \dots, n\}$ and let $k \in \mathbb{N}$ be the security parameter. For an algorithm A , let $y \leftarrow A(1^k, x)$ be the process of running A , on input 1^k and x , with access to uniformly random coins and assigning the result to y . We assume that all algorithms take 1^k as input and we will sometimes not make this explicit in the following. To make the random coins r explicit, we write $A(1^k, x; r)$. An algorithm A is probabilistic polynomial time (PPT) if its running time is polynomially bounded in k . A function f is negligible if $\forall c \exists k_0 \forall k \geq k_0 : |f(k)| \leq 1/k^c$. For binary trees, we denote the root node with ε and all other nodes are encoded as binary strings, i.e., for a node w we denote child nodes as $w0$ and $w1$.

Homomorphic public-key encryption. A \mathcal{F} -homomorphic public key encryption (HPKE) scheme is a public-key encryption (PKE) scheme that is homomorphic with respect to a class of functions \mathcal{F} , i.e., given a sequence of

ciphertexts to messages $(M_i)_{i \in [n]}$ one can evaluate a function $f : \mathcal{M}^n \rightarrow \mathcal{M} \in \mathcal{F}$ on the ciphertexts such that the resulting ciphertext decrypts to $f(M_1, \dots, M_n)$.

Definition 1 ((\mathcal{F} -)HPKE). A \mathcal{F} -homomorphic public key encryption (\mathcal{F} -HPKE or HPKE for short) scheme with message space \mathcal{M} , ciphertext space \mathcal{C} and a function family \mathcal{F} consists of the PPT algorithms $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$:

$\text{Gen}(1^k)$: On input security parameter k , outputs public and secret keys (pk, sk) .

$\text{Enc}(\text{pk}, M)$: On input a public key pk , and a message $M \in \mathcal{M}$, outputs a ciphertext $C \in \mathcal{C}$.

$\text{Dec}(\text{sk}, C)$: On input a secret key sk , and ciphertext C , outputs $M \in \mathcal{M} \cup \{\perp\}$.

$\text{Eval}(f, (C_i)_{i \in [n]})$: On input a function $f : \mathcal{M}^n \rightarrow \mathcal{M} \in \mathcal{F}$, a sequence of ciphertexts $(C_i)_{i \in [n]}$ encrypted under the same public key, outputs C .

In addition to the standard and folklore correctness definition for public-key encryption (PKE), we further require for HPKE that for all security parameters $k \in \mathbb{N}$, all key pairs $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k)$, all functions $f : \mathcal{M}^n \rightarrow \mathcal{M} \in \mathcal{F}$, all message sequences $(M_i)_{i \in [n]}$ it holds that $\text{Dec}(\text{sk}, \text{Eval}(f, (\text{Enc}(\text{pk}, M_i))_{i \in [n]})) = f(M_1, \dots, M_n)$. We are particularly interested in the case where \mathcal{M} is a group and \mathcal{F} is the set of all *linear functions* on products of \mathcal{M} . In that case, we call the HPKE scheme *linearly homomorphic*. For a HPKE, we require conventional IND-CPA security as with PKE schemes and recall an efficient instantiation of a linearly homomorphic scheme, i.e., linear ElGamal [11], in the full version.

Proxy re-encryption. Subsequently, we define proxy re-encryption and defer a formal treatment of security to Sect. 3.

Definition 2 (PRE). A proxy re-encryption (PRE) scheme with message space \mathcal{M} consists of the PPT algorithms $(\text{Setup}, \text{Gen}, \mathbf{Enc}, \mathbf{Dec}, \text{ReGen}, \text{ReEnc})$ where $\mathbf{Enc} = (\text{Enc}^{(j)})_{j \in [2]}$ and $\mathbf{Dec} = (\text{Dec}^{(j)})_{j \in [2]}$. For $j \in [2]$, they are defined as follows.

$\text{Setup}(1^k)$: On input security parameter k , outputs public parameters pp .

$\text{Gen}(\text{pp})$: On input public parameters pp , outputs public and secret keys (pk, sk) .

$\text{Enc}^{(j)}(\text{pk}, M)$: On input a public key pk , and a message $M \in \mathcal{M}$ outputs a level j ciphertext C .

$\text{Dec}^{(j)}(\text{sk}, C)$: On input a secret key sk , and level j ciphertext C , outputs $M \in \mathcal{M} \cup \{\perp\}$.

$\text{ReGen}(\text{sk}_A, \text{pk}_B)$: On input a secret key sk_A and a public key pk_B for B , outputs a re-encryption $\text{rk}_{A \rightarrow B}$.

$\text{ReEnc}(\text{rk}_{A \rightarrow B}, C_A)$: On input a re-encryption key $\text{rk}_{A \rightarrow B}$, and a ciphertext C_A for user A , outputs a ciphertext C_B for user B .

Binary tree encryption. Binary tree encryption (BTE) [15] is a relaxed version of hierarchical identity-based encryption (HIBE) [25]. Similar to a HIBE scheme, a BTE scheme has a (master) public key associated to a binary tree where each node in the tree has a corresponding secret key. To encrypt a message for some node, one uses both the public key and the name of the target node. Using the

node's secret key, the resulting ciphertext can then be decrypted. Additionally, the secret key of a node can be used to derive the secret keys of its child nodes.

In contrast to BTE defined in [15], we make the part of the secret key used to perform the key derivation explicit, i.e., we will have secret keys for the decryption and derivation keys to derive secret keys. In case, an instantiation does not support a clear distinction, it is always possible to assume that the derivation key is empty and everything is contained in the secret key.

Definition 3. A binary tree encryption (BTE) scheme with message space \mathcal{M} consists of the PPT algorithms (Gen, Evo, Enc, Dec) as follows:

$\text{Gen}(1^k, \ell)$: On input security parameter k and depth of the tree ℓ , outputs public, secret, and derivation keys $(\text{pk}, \text{sk}^{(\varepsilon)}, \text{dk}^{(\varepsilon)})$.

$\text{Der}(\text{sk}^{(w)}, \text{dk}^{(w)})$: On input secret key $\text{sk}^{(w)}$ and derivation key $\text{dk}^{(w)}$, for node $w \in \{0, 1\}^{<\ell}$, outputs secret keys $\text{sk}^{(w_0)}, \text{sk}^{(w_1)}$ and derivation keys $\text{dk}^{(w_0)}, \text{dk}^{(w_1)}$ for the two children of w .

$\text{Enc}(\text{pk}, M, w)$: On input a public key pk , a message $M \in \mathcal{M}$, and node $w \in \{0, 1\}^{\leq \ell}$, outputs a ciphertext C .

$\text{Dec}(\text{sk}^{(w)}, C)$: On input a secret key $\text{sk}^{(w)}$, for node $w \in \{0, 1\}^{\leq \ell}$, and ciphertext C , outputs $M \in \mathcal{M} \cup \{\perp\}$.

For correctness, we require that for all security parameters $k \in \mathbb{N}$, all depths $\ell \in \mathbb{N}$, all key pairs $(\text{pk}, (\text{sk}^{(\varepsilon)}, \text{ek}^{(\varepsilon)}))$ generated by $\text{Gen}(1^k, \ell)$, any node $w \in \{0, 1\}^{\leq \ell}$, any derived key $\text{sk}^{(w)}$ derived using Der from $(\text{sk}^{(\varepsilon)}, \text{dk}^{(\varepsilon)})$, and all messages $M \in \mathcal{M}$, it holds that $\text{Dec}(\text{sk}^{(w)}, \text{Enc}(\text{pk}, M, w)) = M$.

The indistinguishability against selective node, chosen plaintext attacks (IND-SN-CPA) is a generalization of the standard IND-CPA security notion of PKE schemes. Essentially, the security notion requires the adversary to commit to the node to be attacked in advance. The adversary gets access to all secret keys except the secret keys for all nodes that are on the path from the root node to the targeted node.

Experiment $\text{Exp}_{\text{BTE}, A}^{\text{ind-sn-cpa}}(1^k, \ell)$

$(\text{pk}, \text{sk}^{(\varepsilon)}, \text{dk}^{(\varepsilon)}) \leftarrow \text{Gen}(1^k, \ell)$

$b \xleftarrow{R} \{0, 1\}$

$(w^*, \text{st}) \leftarrow A(1^k, \ell)$

Let W be the set of all nodes that are siblings to the path from the root node to w^* and (if possible) w^*0 and w^*1 .

Compute $(\text{sk}^{(w)}, \text{dk}^{(w)})$ for all $w \in W$ from $(\text{sk}^{(\varepsilon)}, \text{dk}^{(\varepsilon)})$ using Der .

$(M_0, M_1, \text{st}) \leftarrow A(\text{st}, \text{pk}, (\text{sk}^{(w)}, \text{dk}^{(w)})_{w \in W})$

$b^* \leftarrow A(\text{st}, \text{Enc}(\text{pk}, M_b, w^*))$

if $b = b^*$ return 1, else return 0

Experiment 1. The IND-SN-CPA security experiment for a BTE scheme.

Definition 4 (IND-SN-CPA). For a polynomially bounded function ℓ , a PPT adversary A , we define the advantage function in the sense of IND-SN-CPA as

$$\text{Adv}_{\text{BTE},A}^{\text{ind-sn-cpa}}(1^k, \ell(k)) = \left| \Pr \left[\text{Exp}_{\text{BTE},A}^{\text{ind-sn-cpa}}(1^k, \ell(k)) = 1 \right] - \frac{1}{2} \right|.$$

If for all ℓ , and any A there exists a negligible function ε such that $\text{Adv}_{\text{BTE},A}^{\text{ind-sn-cpa}}(1^k, \ell(k)) < \varepsilon(k)$, then a BTE scheme is IND-SN-CPA secure.

The CHK Compiler. The technique of Canetti et al. [15] can be summarized as follows. To build a forward-secret PKE scheme with n periods, one uses a BTE of depth ℓ such that $n < 2^{\ell+1}$. Associate each period with a node of the tree and write w^i to denote the node for period i . The node for period 0 is the root node, i.e. $w^0 = \varepsilon$. If w^i is an internal node, then set $w^{i+1} = w^i 0$. Otherwise, if w^i is a leaf node and $i < N - 1$, then set $w^{i+1} = w^i 1$ where w^i is the longest string such that $w^i 0$ is a prefix of w^i . The public key is simply the public key of the BTE scheme. The secret key for period i consists of the secret key for node w^i .

3 Security of (Forward-Secret) Proxy Re-encryption

Proxy re-encryption (PRE) schemes can exhibit several important properties. In the following, we focus on the most common PRE properties in the cryptographic literature, i.e., uni-directionality (Alice is able to delegate decryption rights to Bob but not from Bob to Alice), non-interactivity (Alice can generate delegation key material without interacting with Bob), and collusion-safeness (even if Bob and other delegates are colluding with the proxy, they cannot extract Alice' full secret key). Moreover, we consider PRE schemes that only allow a single hop, i.e., a ciphertext can be re-encrypted only a single time in contrast to multiple times in a row (multi-hop). Latter can be problematic due to unwanted transitivity.

In this work, we examine a further property of PRE schemes, namely the property of forward secrecy and propose the first uni-directional, non-interactive, collusion-safe, single hop, and forward-secret PRE scheme (dubbed fs-PRE) in the standard model from generic assumptions. Subsequently, in Sect. 3.1, we present the formal model for fs-PRE, while in Sect. 3.3 we discuss the relation and application of our stronger model to the conventional (i.e., plain) PRE security model.

3.1 Syntax of Forward-Secret Proxy Re-encryption

To realize forward-secure PRE (fs-PRE), we lift the definitions and security models of uni-directional, single-hop, non-interactive, and collusion-safe PRE to a setting where we can have several periods. Thereby, we allow re-encryptions in every period such that re-encryption keys—in the same way as secret keys—are bound to a period. Furthermore, we align our PRE definitions with Ateniese et

al. as well as Libert and Vergnaud [4, 5, 33] such that if we only have a single period, then they are equivalent to the definitions for plain PRE in [5, 33].¹¹

Definition 5 (fs-PRE). *A forward-secure proxy re-encryption (fs-PRE) scheme with message space \mathcal{M} consists of the PPT algorithms (Setup, Gen, Evo, **Enc**, **Dec**, ReGen, ReEvo, ReEnc) where $\mathbf{Enc} = (\text{Enc}^{(j)})_{j \in [2]}$ and $\mathbf{Dec} = (\text{Dec}^{(j)})_{j \in [2]}$ for levels $j \in [2]$. We denote level-2 ciphertext as re-encryptable ciphertexts, whereas level-1 ciphertexts are not re-encryptable.*

$\text{Setup}(1^k)$: On input security parameter k , outputs public parameters pp .

$\text{Gen}(\text{pp}, n)$: On input public parameters pp , and number of periods $n \in \mathbb{N}$, outputs public and secret keys $(\text{pk}, (\text{sk}^{(0)}, \text{ek}^{(0)}))$.

$\text{Evo}(\text{sk}^{(i)}, \text{ek}^{(i)})$: On input secret key $\text{sk}^{(i)}$ and evolution key $\text{ek}^{(i)}$ for period $i \in \{0, \dots, n-2\}$, outputs a secret key $\text{sk}^{(i+1)}$ and evolution key $\text{ek}^{(i+1)}$ for period $i+1$.

$\text{Enc}^{(j)}(\text{pk}, M, i)$: On input a public key pk , a message $M \in \mathcal{M}$, and period $i \in \{0, \dots, n-1\}$, outputs a level- j ciphertext C .

$\text{Dec}^{(j)}(\text{sk}^{(i)}, C)$: On input a secret key $\text{sk}^{(i)}$, for period $i \in \{0, \dots, n-1\}$, and level- j ciphertext C , outputs $M \in \mathcal{M} \cup \{\perp\}$.

$\text{ReGen}(\text{sk}_A^{(i)}, \text{ek}_A^{(i)}, \text{pk}_B)$: On input a secret key $\text{sk}_A^{(i)}$ and a evolution key $\text{ek}_A^{(i)}$ (or \perp) for A and period $i \in \{0, \dots, n-1\}$, and a public key pk_B for B , outputs a re-encryption $\text{rk}_{A \rightarrow B}^{(i)}$ and re-encryption-evolution key $\text{rek}_{A \rightarrow B}^{(i)}$ (or \perp).

$\text{ReEvo}(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)})$: On input a re-encryption key $\text{rk}_{A \rightarrow B}^{(i)}$, and a re-encryption-evolution key $\text{rek}_{A \rightarrow B}^{(i)}$ for period $i \in \{0, \dots, n-2\}$, outputs a re-encryption key $\text{rk}_{A \rightarrow B}^{(i+1)}$ and re-encryption evolution key $\text{rek}_{A \rightarrow B}^{(i+1)}$ for the period $i+1$.

$\text{ReEnc}(\text{rk}_{A \rightarrow B}^{(i)}, C_A)$: On input a re-encryption key $\text{rk}_{A \rightarrow B}^{(i)}$, and a (level-2) ciphertext C_A for user A , outputs a (level-1) ciphertext C_B for user B .

Correctness. For correctness, we basically require on the one hand that every ciphertext encrypted for some period i can be decrypted with the respective secret key from period i . On the other hand—when also considering re-encryptable and re-encrypted ciphertexts—we require that level-2 ciphertexts encrypted for period i can be re-encrypted with a suitable re-encryption key for the same period and then decrypted using the (delegatee’s) respective secret key for period i . More formally, for all security parameters $k \in \mathbb{N}$, all public parameters $\text{pp} \leftarrow \text{Setup}(1^k)$, any number of periods $n \in \mathbb{N}$ and users $U \in \mathbb{N}$, all key tuples $(\text{pk}_u, \text{sk}_u^{(0)}, \text{ek}_u^{(0)})_{u \in [U]}$ generated by $\text{Gen}(1^k, n)$, any period $i \in \{0, \dots, n-1\}$, for any $u \in [U]$, any evolved key $\text{sk}_u^{(i+1)}$ generated by $\text{Evo}(\text{sk}_u^{(i)})$, for all $u' \in [U], u \neq u'$, any (potentially evolved) re-encryption and

¹¹ Observe that for a single period, i.e., $n = 1$, Evo and ReEvo in Definition 5 are not defined. Dropping these algorithms and the corresponding evolution keys ek and rek yields a plain PRE scheme.

re-encryption-evolution keys $\text{rk}_{u \rightarrow u'}^{(i)}$ and $\text{rek}_{u \rightarrow u'}^{(i)}$, respectively, for period i generated using ReGen from (potentially evolved) secret and evolution keys as well as the target public key, and all messages $M \in \mathcal{M}$, it holds that

$$\forall j \in [2] \exists j' \in [2] : \text{Dec}^{(j')}(\text{sk}_u^{(i)}, \text{Enc}^{(j)}(\text{pk}_u, M, i)) = M, \\ \text{Dec}^{(1)}(\text{sk}_{u'}^{(i)}, \text{ReEnc}(\text{rk}_{u \rightarrow u'}^{(i)}, \text{Enc}^{(2)}(\text{pk}_u, M, i))) = M.$$

3.2 Security of Forward-Secret Proxy Re-encryption

The security notions for fs-PRE are heavily inspired by the security notions of (plain) PRE [4, 5, 33] and forward-secret PKE [15]. We will discuss multiple notions, combine them carefully, and introduce forward-secret indistinguishably under chosen-plaintext attacks for level-1 and level-2 ciphertexts (termed fs-IND-CPA-1 and fs-IND-CPA-2, respectively) which we argue to be reasonable notions in our setting. Additionally, we define a new (stronger) variant of indistinguishably-under-chosen-plaintext-attacks security for fs-PRE (dubbed fs-RIND-CPA) that focuses on malicious users in the face of honest proxies. In particular, the latter strengthen the folklore PRE security notion.

For all experiments defined in this section, the environment keeps initially empty lists of dishonest (DU) and honest users (HU). The oracles are defined as follows:

$\text{Gen}^{(h)}(\text{pp}, n)$: Run $(\text{pk}, \text{sk}, \text{ek}) \leftarrow \text{Gen}(\text{pp}, n)$, set $\text{HU} \leftarrow \text{HU} \cup \{(\text{pk}, \text{sk}, \text{ek})\}$, and return pk .

$\text{Gen}^{(d)}(\text{pp}, n)$: Run $(\text{pk}, \text{sk}, \text{ek}) \leftarrow \text{Gen}(\text{pp}, n)$, set $\text{DU} \leftarrow \text{DU} \cup \{(\text{pk}, \text{sk}, \text{ek})\}$, and return $(\text{pk}, \text{sk}, \text{ek})$.

$\text{ReGen}^{(h)}(j, \text{pk}_u, \text{pk})$: On input a period j , a public key pk_u and a public key pk , abort if $(\text{pk}_u, \cdot, \cdot) \notin \text{HU}$. Otherwise, look up $\text{sk}_u^{(0)}$ and $\text{ek}_u^{(0)}$ corresponding to pk_u from HU. If $j > 0$ set $(\text{sk}_u^{(j)}, \text{ek}_u^{(j)}) \leftarrow \text{Evo}(\text{sk}_u^{(j-1)}, \text{ek}_u^{(j-1)})$ for $i \in [j]$. Return $\text{ReGen}(\text{sk}_u^{(j)}, \text{ek}_u^{(j)}, \text{pk})$.

$\text{ReGen}^{(h')}(j, \text{sk}^{(0)}, \text{ek}^{(0)}, \text{pk}_u)$: On input a period j , secret key $\text{sk}^{(0)}$, evolution key $\text{ek}^{(0)}$, and a public key pk_u , abort if $(\text{pk}_u, \cdot, \cdot) \notin \text{HU}$. Otherwise, if $j > 0$ set $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $i \in [j]$. Return $\text{ReGen}(\text{sk}^{(j)}, \text{ek}^{(j)}, \text{pk}_u)$.

$\text{ReGen}^{(d)}(j, \text{sk}^{(0)}, \text{ek}^{(0)}, \text{pk}_d)$: On input a period j , secret key $\text{sk}^{(0)}$, evolution key $\text{ek}^{(0)}$, and a public key pk_d , abort if $(\text{pk}_d, \cdot, \cdot) \notin \text{DU}$. Otherwise, if $j > 0$ set $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $i \in [j]$. Return $\text{ReGen}(\text{sk}^{(j)}, \text{ek}^{(j)}, \text{pk}_d)$.

fs-IND-CPA-i security. We start with the definition of fs-IND-CPA-1 and fs-IND-CPA-2 security for fs-PRE. Inspired by the work on forward secrecy due to Canetti et al. [15], our experiments lift standard PRE security notions as defined in Ateniese et al. [4] (AFGH) to the forward-secrecy setting. More concretely, after the selection of a target period j^* by the adversary A , A gets access to the secret and the evolution key $(\text{sk}^{(j^*)}, \text{ek}^{(j^*)})$ of the target period j^* , while the challenge ciphertext for A -chosen message M_b is generated for period $j^* - 1$, for

uniform $b \leftarrow \{0, 1\}$. Eventually, A outputs a guess on b . We say A is valid if A only outputs equal-length messages $|M_0| = |M_1|$ and $1 \leq j^* \leq n$.

Furthermore, we adapted the AFGH security experiment such that A has access to re-encryption and re-encryption-evolution keys for period $j^* - 1$. Analogously to previous work on PRE, we present two separate notions for level-1 and level-2 ciphertexts. The corresponding security experiments are given in Experiment 2 and Experiment 3. The only difference in Experiment 2 is that for level-1 ciphertexts, i.e., the ones which can no longer be re-encrypted, the adversary gets access to more re-encryption and re-encryption-evolution keys (obviously, the challenge ciphertext in that experiment is a level-1 ciphertext).

Experiment $\text{Exp}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-1}}(1^k, n)$

$\text{pp} \leftarrow \text{Setup}(1^k), (\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n), b \xleftarrow{R} \{0, 1\}$
 $(j^*, \text{st}) \leftarrow A(\text{pp}, n, \text{pk})$
 $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $j \in [j^*]$.
 $\mathcal{O} \leftarrow \{\text{Gen}^{(h)}, \text{ReGen}^{(h)}(j^* - 1, \cdot, \text{pk}), \text{ReGen}^{(h')}(j^* - 1, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot), \text{Gen}^{(d)}, \text{ReGen}^{(d)}(j^* - 1, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot)\}$
 $(M_0, M_1, \text{st}) \leftarrow A^{\mathcal{O}}(\text{st}, \text{sk}^{(j^*)}, \text{ek}^{(j^*)})$
 $b^* \leftarrow A(\text{st}, \text{Enc}^{(1)}(\text{pk}, M_b, j^* - 1))$
 if $b = b^*$ return 1, else return 0

Experiment 2. The fs-IND-CPA-1 security experiment for level-1 ciphertexts of fs-PRE schemes.

Experiment $\text{Exp}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-2}}(1^k, n)$

$\text{pp} \leftarrow \text{Setup}(1^k), (\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n), b \xleftarrow{R} \{0, 1\}$
 $(j^*, \text{st}) \leftarrow A(\text{pp}, n, \text{pk})$
 $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $j \in [j^*]$.
 $\mathcal{O} \leftarrow \{\text{Gen}^{(h)}, \text{ReGen}^{(h)}(j^* - 1, \cdot, \text{pk}), \text{ReGen}^{(h')}(j^* - 1, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot)\}$
 $(M_0, M_1, \text{st}) \leftarrow A^{\mathcal{O}}(\text{st}, \text{sk}^{(j^*)}, \text{ek}^{(j^*)})$
 $b^* \leftarrow A(\text{st}, \text{Enc}^{(2)}(\text{pk}, M_b, j^* - 1))$
 if $b = b^*$ return 1, else return 0

Experiment 3. The fs-IND-CPA-2 security experiment for level-2 ciphertexts of fs-PRE schemes.

Definition 6 (fs-IND-CPA-i). For a polynomially bounded function $n(\cdot) > 1$, a PPT adversary A , we define the advantage function for A in the sense of fs-IND-CPA- i for level- i ciphertexts as

$$\text{Adv}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-}i}(1^k, n(k)) := \left| \Pr \left[\text{Exp}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-}i}(1^k, n(k)) = 1 \right] - \frac{1}{2} \right|.$$

A fs-PRE scheme is fs-IND-CPA- i secure if for all polynomially bounded $n(\cdot) > 1$ and any valid PPT A there exists a negligible function ε such that $\text{Adv}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-}i}(1^k, n(k)) < \varepsilon(k)$, where $\text{Exp}_{\text{fs-PRE}, A}^{\text{fs-ind-cpa-}i}$, for all $i \in [2]$, are defined in Experiment 2 and Experiment 3, respectively.

Master-secret security. As discussed in [33], the security notion for level-1 (i.e., non re-encryptable) ciphertexts already implies classical master-secret security notion for PRE [4].¹² However, this must not be the case in the forward-secret setting. To formally close this gap, we give a trivial lemma (cf. Lemma 1) which states that fs-IND-CPA-1 implies master-secret security in the sense of Experiment 4 in the forward-secrecy setting. Essentially, master-secret security ensures collusion safeness such that re-encryption keys in period j do not leak the secret key corresponding to level-1 ciphertexts which can not be re-encrypted in period $j-1$. In Experiment 4, we lift master-secret security in the classical PRE sense to the forward-secret setting. In the experiment, the adversary A selects a target period j^* and receives the secret and evolution keys $(\text{sk}^{(j^*)}, \text{ek}^{(j^*)})$ for the target period in return. Within the experiment, A has access to several oracles, e.g., to obtain re-encryption and re-encryption-evolution keys for period j^* . Eventually, A outputs secret and evolutions keys $(\text{sk}^*, \text{ek}^*)$ and the experiment returns 1 (i.e., A wins) if $(\text{sk}^*, \text{ek}^*) = (\text{sk}^{(j^*-1)}, \text{ek}^{(j^*-1)})$. We say A is valid if A only outputs $1 \leq j^* \leq n$.

Experiment $\text{Exp}_{fs\text{-PRE},A}^{\text{fs-msk}}(1^k, n)$

$\text{pp} \leftarrow \text{Setup}(1^k), (\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n)$
 $(j^*, \text{st}) \leftarrow A(\text{pp}, n, \text{pk})$
 $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $j \in [j^*]$.
 $\mathcal{O} \leftarrow \{\text{Gen}^{(h)}, \text{ReGen}^{(h)}(j^*, \cdot, \text{pk}), \text{ReGen}^{(h')}(j^*, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot), \text{Gen}^{(d)}, \text{ReGen}^{(d)}(j^*, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot)\}$
 $(\text{sk}^*, \text{ek}^*) \leftarrow A^{\mathcal{O}}(\text{st}, \text{sk}^{(j^*)}, \text{ek}^{(j^*)})$
 if $(\text{sk}^*, \text{ek}^*) = (\text{sk}^{(j^*-1)}, \text{ek}^{(j^*-1)})$ return 1, else return 0

Experiment 4. The forward secure master secret security experiment for fs-PRE schemes.

Definition 7 (fs-master-secret security). For a polynomially bounded function $n(\cdot) > 1$ and a PPT adversary A , we define the advantage function for A in the sense of fs-master-secret security as

$$\text{Adv}_{fs\text{-PRE},A}^{\text{fs-msk}}(1^k, n(k)) := \Pr \left[\text{Exp}_{fs\text{-PRE},A}^{\text{fs-msk}}(1^k, n(k)) = 1 \right].$$

A fs-PRE scheme is fs-master-secret secure if for all polynomially bounded $n(\cdot) > 1$ and any valid PPT A there exists a negligible function ε such that $\text{Adv}_{fs\text{-PRE},A}^{\text{fs-msk}}(1^k, n(k)) < \varepsilon(k)$, where $\text{Exp}_{fs\text{-PRE},A}^{\text{fs-msk}}$ is defined in Experiment 4.

We now show that this notion in the sense of Definition 7 is trivially implied by fs-IND-CPA-1 security for fs-PRE in the sense of Definition 6.

¹² As we will discuss below, this notion seems to suggest false guarantees and leaves a critical gap in the security model open.

Lemma 1. *If a fs-PRE scheme is fs-IND-CPA-1 secure in the sense of Definition 6, then the same fs-PRE scheme is fs-master-secret secure in the sense of Definition 7.*

Proof sketch. It is trivial to see that any successful PPT adversary on the fs-master-secret security of a fs-PRE scheme can be transformed into a PPT adversary on the fs-IND-CPA-1 security of that fs-PRE scheme. (Essentially, any PPT adversary that is able to gain access to the secret key of the prior period can trivially distinguish ciphertexts for the same period.)

The problem with (fs-)PRE security. A problem with the notion of standard (i.e., IND-CPA and master secret) security for (plain) PRE and also our fs-PRE notions so far is that the secret keys used for level-1 (i.e., non re-encryptable) and level-2 (i.e., re-encryptable) ciphertexts can be independent. Consequently, although ciphertexts on both levels can be shown to be indistinguishable, this does not rule out the possibility that ciphertexts on level-2 reveal the respective level-2 secret key of the sender to an legitimate receiver. This is exactly the reason for the gap in the plain PRE model which allows to leak a “level-2 secret key” once a re-encryption has been performed while all security properties are still satisfied (we provide an example for such a scheme in Sect. 4.4). In particular, this allows the receiver to potentially decrypt *any* level-2 ciphertext. We provide a solution in form of a stronger security notion which we term fs-RIND-CPA security in the following.

fs-RIND-CPA security. We observe that existing PRE notions only consider that (1) as long as the users are honest, the proxy learns nothing about any plaintext, and (2) if proxies and users collude they do not learn anything about the ciphertexts which are not intended to be re-encrypted. We go a step further and consider malicious users in the face of an honest proxy in the forward-secret and, hence, also in the plain PRE sense. That is, we want to enforce that a malicious user can only read the ciphertexts which were actually re-encrypted by the proxy and can not tell anything about the ciphertexts which can potentially be re-encrypted. We capture this via the notion of fs-RIND-CPA security. In this scenario, an adversary receives re-encrypted ciphertexts generated by an honest proxy, that it is able to decrypt. Nevertheless, for all other level-2 ciphertexts, the adversary should still be unable to recover the plaintext. In Experiment 5, we model this notion where the adversary gets access to a ReEnc-oracle which is in possession of the re-encryption key from the target user to the adversary. We say A is valid if A only outputs $1 \leq j^* \leq n$ and equal length messages $|M_0| = |M_1|$.

Definition 8 (fs-RIND-CPA). *For a polynomially bounded function $n(\cdot)$ and a PPT adversary A , we define the advantage function for A in the sense of fs-RIND-CPA as*

$$\text{Adv}_{fs\text{-PRE}, A}^{\text{fs-rind-cpa}}(1^k, n(k)) := \left| \Pr \left[\text{Exp}_{fs\text{-PRE}, A}^{\text{fs-rind-cpa}}(1^k, n(k)) = 1 \right] - \frac{1}{2} \right|.$$

Experiment $\text{Exp}_{fs\text{-PRE},A}^{fs\text{-rind-cpa}}(1^k, n)$

$\text{pp} \leftarrow \text{Setup}(1^k), (\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n), b \xleftarrow{R} \{0, 1\}$
 $(j^*, \text{pk}^*, \text{st}) \leftarrow A(\text{pp}, n, \text{pk})$
 $(\text{sk}^{(j)}, \text{ek}^{(j)}) \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $j \in [j^*]$
 $\text{rk} \leftarrow \text{ReGen}(\text{sk}^{(j^*)}, \perp, \text{pk}^*)$
 $(M_0, M_1, \text{st}) \leftarrow A^{\{\text{ReEnc}(\text{rk}, \cdot)\}}(\text{st})$
 $b^* \leftarrow A(\text{st}, \text{Enc}^{(2)}(\text{pk}, M_b, j^*))$
 if $b = b^*$ return 1, else return 0

Experiment 5. The fs-RIND-CPA security experiment for fs-PRE schemes.

A fs-PRE scheme is fs-RIND-CPA if for all polynomially bounded $n(\cdot)$ and any valid PPT A there exists a negligible function ε such that $\text{Adv}_{fs\text{-PRE},A}^{fs\text{-rind-cpa}}(1^k, n(k)) < \varepsilon(k)$, where $\text{Exp}_{fs\text{-PRE},A}^{fs\text{-rind-cpa}}$ is defined in Experiment 5.

We distinguish fs-PRE schemes based on this last notion:

Definition 9 (fs-PRE⁻-security). If a fs-PRE scheme is fs-IND-CPA-1 and fs-IND-CPA-2 secure, then we say this fs-PRE scheme is fs-PRE⁻-secure.

Definition 10 (fs-PRE⁺-security). If a fs-PRE scheme is fs-IND-CPA-1, fs-IND-CPA-2, and fs-RIND-CPA secure, then we say this fs-PRE scheme is fs-PRE⁺-secure.

3.3 Stronger Security for Proxy Re-encryption

To conclude the discussion of the security model of fs-PRE schemes, we first observe that it is interesting to consider the notion of fs-RIND-CPA security in the classical setting for PRE, i.e., Experiment 5 with fixed $n = 1$ and no call to the Evo algorithm. The notion again ensures involvement of the proxy for the re-encryption of every ciphertext, and can, thus, enforce that malicious users cannot learn anything beyond the explicitly re-encrypted ciphertexts. This immediately leads to a stronger security model for classical PRE (given in the full version), which we denote as PRE⁺. In particular, it extends the classical model [4], dubbed PRE⁻, which covers standard (IND-CPA) and master-secret security definitions, by our fs-RIND-CPA security notion ported to the PRE setting. As our fs-IND-CPA-i notions for fs-PRE are generalizations of the established standard security notions of PRE as defined in [4], we consequently obtain a PRE⁺-secure PRE scheme from any fs-PRE⁺-secure fs-PRE scheme. We formalize this observation via Lemma 2.

Lemma 2. Any fs-PRE⁺-secure fs-PRE scheme yields a PRE⁺-secure PRE scheme.

In the full version, we formally prove this lemma. This immediately gives us a construction for a PRE⁺-secure PRE scheme.

Corollary 1. Scheme 3 when limited to a single time period, i.e., setting $n = 1$, represents a PRE⁺-secure PRE scheme.

4 Constructing Fs-PRE from Binary Tree Encryption

In this section we present our construction of fs-PRE which is based on BTEs. Along the way, we introduce the notion of forward-secret delegatable PKE (fs-DPKE) as intermediate step. Such a fs-DPKE scheme then directly gives us a first fs-PRE satisfying fs-PRE⁻ security. To extend our construction to satisfy the stronger fs-PRE⁺ notion generically, we require a relatively mild homomorphic property of the fs-DPKE. This property is in particular satisfied by our fs-DPKE instantiation, which yields the first fs-PRE scheme with strong security.

4.1 Forward-Secret Delegatable Public-Key Encryption

We now formalize fs-DPKE. In such a scheme decryption rights within a public-key encryption scheme can be delegated from a delegator to a delegatee and secret keys of delegators can be evolved so that a secret key for some period e_i is no longer useful to decrypt ciphertexts of prior periods e_j with $j < i$.

Definition 11 (fs-DPKE). *A forward-secret delegatable PKE (fs-DPKE) scheme with message space \mathcal{M} consists of the PPT algorithms (Setup, Gen, Evo, Del, Enc, Dec, DelEvo, DelDec) as follows:*

Setup(1^k): *On input security parameter k , outputs public parameters pp .*

Gen(pp, n): *On input public parameters pp , and maximum number of periods n , outputs public, secret and evolution keys $(\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)})$.*

Evo($\text{sk}^{(i)}, \text{ek}^{(i)}$): *On input secret key $\text{sk}^{(i)}$, and evolution key $\text{ek}^{(i)}$ for period $i \in \{0, \dots, n-2\}$, outputs secret key $\text{sk}^{(i+1)}$ and evolution key $\text{ek}^{(i+1)}$ for period $i+1$.*

Del($\text{sk}_A^{(i)}, \text{ek}_A^{(i)}, \text{pk}_B$): *On input secret key $\text{sk}_A^{(i)}$ and evolution key $\text{ek}_A^{(i)}$ (or \perp) for A and period $i \in \{0, \dots, n-1\}$, and public key pk_B for B , outputs delegated key $\text{dk}^{(i)}$ and delegated evolution key $\text{dek}^{(i)}$ (or \perp) for period i .*

Enc(pk, M, i): *On input a public key pk , a message $M \in \mathcal{M}$, and period $i \in \{0, \dots, n-1\}$, outputs a ciphertext C .*

Dec($\text{sk}^{(i)}, C$): *On input a secret key $\text{sk}^{(i)}$, for period $i \in \{0, \dots, n-1\}$, and ciphertext C , outputs $M \in \mathcal{M} \cup \{\perp\}$.*

DelEvo($\text{dk}^{(i)}, \text{dek}^{(i)}$): *On input a delegation key $\text{dk}^{(i)}$ and delegated evolution key $\text{dek}^{(i)}$ for period $i \in \{0, \dots, n-2\}$, output delegation key $\text{dk}^{(i+1)}$ and delegated evolution key $\text{dek}^{(i+1)}$ for period $i+1$.*

DelDec($\text{sk}_B^{(i)}, \text{dk}_{A \rightarrow B}^{(i)}, C_A$): *On input secret key $\text{sk}_B^{(i)}$ for B and period $i \in \{0, \dots, n-1\}$, delegation key $\text{dk}_{A \rightarrow B}^{(i)}$ from A for B and period i , and ciphertext C_A for A , outputs $M \in \mathcal{M} \cup \{\perp\}$.*

We note that the existence of the DelEvo algorithm is entirely optional. If provided, it allows the user in possession of a delegation key to evolve it for later periods without additional interaction with the delegator.

Correctness. For correctness we require that period i ciphertexts encrypted for user u can be decrypted if one is in possession of the secret key of u evolved

to that period or one possess a delegation key of u to another user u' and the secret key for u' for that period. More formally, we require that for all security parameters $k \in \mathbb{N}$, all public parameters pp generated by $\text{Setup}(1^k)$, all number of periods $n \in \mathbb{N}$, all users $U \in \mathbb{N}$, all key tuples $(\text{pk}_u, \text{sk}_u^{(0)}, \text{ek}_u^{(0)})_{u \in [U]}$ generated by $\text{Gen}(\text{pp}, n)$, any period $i \in \{0, \dots, n-1\}$, for any $u \in [U]$, any evolved keys $(\text{sk}_u^{(i)}, \text{ek}_u^{(i)})$ generated by Evo from $(\text{sk}_u^{(0)}, \text{ek}_u^{(0)})$, for all $u' \in [U], u \neq u'$, any (potentially evolved) delegation key $\text{dk}_{u \rightarrow u'}^{(i)}$ for period i generated using Del from a (potentially evolved) secret key and the target public key, and all messages $M \in \mathcal{M}$ it holds that

$$\text{Dec}(\text{sk}_u^{(i)}, \text{Enc}(\text{pk}_u, M, i)) = \text{DelDec}(\text{sk}_{u'}^{(i)}, \text{dk}_{u \rightarrow u'}^{(i)}, \text{Enc}(\text{pk}_u, M, i)) = M.$$

Security notions. The forward-secret IND-CPA notion is a straight-forward extension of the typical IND-CPA notion: the adversary selects a target period and gets access to secret and evolution keys of the targeted user for the selected period and is able to request delegation keys with honest and dishonest users for that period. The adversary then engages with an IND-CPA style challenge for the previous period. For the experiment, which is depicted in Experiment 6, the environment keeps a list of an initial empty list of honest users HU .

$\text{Gen}^{(h)}(\text{pp}, n)$: Run $(\text{pk}, \text{sk}, \text{ek}) \leftarrow \text{Gen}(\text{pp}, n)$, set $\text{HU} \leftarrow \text{HU} \cup \{(\text{pk}, \text{sk}, \text{ek})\}$, and return pk .

$\text{Del}^{(h)}(j, \text{pk}_u, \text{pk})$: On input a period j , a public key pk_u and a public key pk , abort if $(\text{pk}_u, \cdot) \notin \text{HU}$. Otherwise, look up $\text{sk}_u^{(0)}, \text{ek}_u^{(0)}$ corresponding to pk_u from HU , set $(\text{sk}_u^{(i)}, \text{ek}_u^{(i)}) \leftarrow \text{Evo}(\text{sk}_u^{(i-1)}, \text{ek}_u^{(i-1)})$ for $i \in [j]$ if $j > 0$, and return $\text{Del}(\text{sk}_u^{(j)}, \text{ek}_u^{(j)}, \text{pk})$.

$\text{Del}^{(h')}(j, \text{sk}^{(0)}, \text{ek}^{(0)}, \text{pk}_u)$: On input a period j , a secret key $\text{sk}^{(0)}$, a evolution key $\text{ek}^{(0)}$, and a public key pk_u , abort if $(\text{pk}_u, \cdot) \notin \text{HU}$. Otherwise, set $(\text{sk}^{(i)}, \text{ek}^{(i)}) \leftarrow \text{Evo}(\text{sk}^{(i-1)}, \text{ek}^{(i-1)})$ for $i \in [j]$ if $j > 0$, and return $\text{Del}(\text{sk}^{(j)}, \text{ek}^{(j)}, \text{pk}_u)$.

Experiment $\text{Exp}_{fs\text{-DPKE}, A}^{\text{fs-ind-cpa}}(1^k, n)$

$\text{pp} \leftarrow \text{Setup}(1^k)$, $(\text{pk}, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n)$, $b \xleftarrow{R} \{0, 1\}$
 $(j^*, \text{st}) \leftarrow A(\text{pp}, n, \text{pk})$
 $\text{sk}^{(j)}, \text{ek}^{(j)} \leftarrow \text{Evo}(\text{sk}^{(j-1)}, \text{ek}^{(j-1)})$ for $j \in [j^*]$.
 $\mathcal{O} \leftarrow \{\text{Gen}^{(h)}, \text{Del}^{(h)}(j^* - 1, \cdot, \text{pk}), \text{Del}^{(h')}(j^* - 1, \text{sk}^{(0)}, \text{ek}^{(0)}, \cdot)\}$
 $(M_0, M_1, \text{st}) \leftarrow A^{\mathcal{O}}(\text{st}, \text{sk}^{(j^*)}, \text{ek}^{(j^*)})$
 $b^* \leftarrow A(\text{st}, \text{Enc}(\text{pk}, M_b, j^* - 1))$
 if $b = b^*$ return 1, else return 0

Experiment 6. The fs-IND-CPA security experiment for a fs-DPKE scheme.

Definition 12 (fs-IND-CPA). For a polynomially bounded function $n(\cdot) > 1$, a PPT adversary A , we define the advantage function in the sense of fs-IND-CPA as

$$\text{Adv}_{fs\text{-DPKE},A}^{\text{fs-ind-cpa}}(1^k, n(k)) := \left| \Pr \left[\text{Exp}_{fs\text{-DPKE},A}^{\text{fs-ind-cpa}}(1^k, n(k)) = 1 \right] - \frac{1}{2} \right|.$$

If for all $n(\cdot) > 1$, and any A there exists a negligible function ε such that $\text{Adv}_{fs\text{-DPKE},A}(1^k, n(k)) < \varepsilon(k)$, then a fs-DPKE scheme is fs-IND-CPA secure.

4.2 Constructing fs-DPKE from BTE

Now we construct a fs-DPKE scheme from a BTE scheme by applying the CHK compiler to a BTE and combining it with an \mathcal{F} -HPKE scheme for handling the delegation keys, i.e., the fs-DPKE key contains a BTE and an \mathcal{F} -HPKE key. The evolution key contains the secret and derivation keys for all right siblings on the path from the root node to w^i as well as the evolution key for w^i . The evolution algorithms traverse the tree in a depth-first manner, hence the evolution keys are viewed as stack and when visiting a node, the derived secret and derivation keys are pushed onto the stack. To simplify the presentation of the scheme, we define an algorithm DFEval that performs the stack manipulation on a stack of pairs:

$\text{DFEval}(s_1^{(w^i)}, s, \text{Eval})$: On input the stack s and first element $s_1^{(w^i)}$ of the pair for node w^i , an algorithm Eval , perform the following steps:

- Pop the topmost element, $(\perp, s_2^{(w^i)})$, from the stack s .
- If w^i is an internal node, set $s^{(w^i0)}, s^{(w^i1)} \leftarrow \text{Eval}(s_1^{(w^i)}, s_2^{(w^i)})$ and push $s^{(w^i1)}, s^{(w^i0)}$ onto s .
- Replace the topmost element, $(s_1^{(w^{i+1})}, s_2^{(w^{i+1})})$, with $(\perp, s_2^{(w^{i+1})})$.
- Return $s_1^{(w^{i+1})}$ and the new stack s .

The overall idea is now to encrypt the BTE secret key of the current period using the \mathcal{F} -HPKE scheme's public key of the target user. Using the homomorphic property of the encryption scheme, we are able to evolve the delegation keys in the same way as the secret keys of the nodes. In particular, we will require that the key derivation algorithm of the BTE can be represented by functions in \mathcal{F} , i.e., $\text{Der}_{\text{BTE}} = (f_i)_{i \in [m]}$. For notional simplicity, we will write $\text{Eval}_{\text{HPKE}}(\text{Der}_{\text{BTE}}, \cdot)$ instead of repeating it for each f_i that represents Der_{BTE} .

For our fs-DPKE scheme we need keys of different users to live in compatible key spaces. To that end, we introduce Setup algorithms for both schemes that fix the key spaces and we change the key generation algorithms to take the public parameters instead of the security parameter as argument. Note that when using the BTE from [15], linear ElGamal [11] as \mathcal{F} -HPKE to encrypt the BTE keys suffices for our needs.

Our construction. The fs-DPKE scheme is detailed in Scheme 1. We note that only the definition of DelEvo relies on the homomorphic properties of the HPKE

<p>Let $(\text{Setup}_{\text{BTE}}, \text{Gen}_{\text{BTE}}, \text{Der}_{\text{BTE}}, \text{Enc}_{\text{BTE}}, \text{Dec}_{\text{BTE}})$ be a BTE scheme and $(\text{Setup}_{\text{HPKE}}, \text{Gen}_{\text{HPKE}}, \text{Enc}_{\text{HPKE}}, \text{Dec}_{\text{HPKE}}, \text{Eval}_{\text{HPKE}})$ a compatible \mathcal{F}-HPKE scheme with $\text{Der}_{\text{BTE}} \in \mathcal{F}$.</p>
<p>$\text{Setup}(1^k)$: Set $\text{pp}_{\text{BTE}} \leftarrow \text{Setup}_{\text{BTE}}(1^k)$, $\text{pp}_{\text{HPKE}} \leftarrow \text{Setup}_{\text{HPKE}}(1^k)$, and return $(\text{pp}_{\text{BTE}}, \text{pp}_{\text{HPKE}})$.</p>
<p>$\text{Gen}(\text{pp}, n)$: Parse pp as $(\text{pp}_{\text{BTE}}, \text{pp}_{\text{HPKE}})$. Choose ℓ such that $n < 2^{\ell+1}$, set $(\text{pk}_{\text{BTE}}, \text{sk}_{\text{BTE}}^{(\varepsilon)}, \text{dk}_{\text{BTE}}^{(\varepsilon)}) \leftarrow \text{Gen}_{\text{BTE}}(\text{pp}_{\text{BTE}}, \ell)$ and $(\text{pk}_{\text{HPKE}}, \text{sk}_{\text{HPKE}}) \leftarrow \text{Gen}_{\text{HPKE}}(\text{pp}_{\text{HPKE}})$, and return $((\text{pk}_{\text{BTE}}, \text{pk}_{\text{HPKE}}), (\text{sk}_{\text{BTE}}^{(\varepsilon)}, \text{sk}_{\text{HPKE}}), (\perp, \text{dk}_{\text{BTE}}^{(\varepsilon)}))$.</p>
<p>$\text{Evo}(\text{sk}^{(i)}, \text{ek}^{(i)})$: Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w^i)}, \text{sk}_{\text{HPKE}})$ and view $\text{ek}^{(i)}$ organized as a stack of secret key and evolution keys pairs. Set $\text{sk}_{\text{BTE}}^{(w^{i+1})}, \text{ek}^{(i+1)} \leftarrow \text{DFEval}(\text{sk}_{\text{BTE}}^{(w^i)}, \text{ek}^{(i)}, \text{Der}_{\text{BTE}})$, and $\text{sk}^{(i+1)} \leftarrow (\text{sk}_{\text{BTE}}^{(w^{i+1})}, \text{sk}_{\text{HPKE}})$. Return $\text{sk}^{(i+1)}, \text{ek}^{(i+1)}$.</p>
<p>$\text{Enc}(\text{pk}, M, i)$: Parse pk as $(\text{pk}_{\text{BTE}}, \cdot)$, and return $\text{Enc}_{\text{BTE}}(\text{pk}_{\text{BTE}}, M, w^i)$.</p>
<p>$\text{Dec}(\text{sk}^{(i)}, C)$: Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w^i)}, \cdot)$, and return $\text{Dec}_{\text{BTE}}(\text{sk}_{\text{BTE}}^{(w^i)}, C)$.</p>
<p>$\text{Del}(\text{sk}_A^{(i)}, \text{ek}_A^{(i)}, \text{pk}_B)$: Parse $\text{sk}_A^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w^i)}, \cdot)$ and pk_B as $(\cdot, \text{pk}_{\text{HPKE}})$. If $\text{ek}_A^{(i)} = \perp$, return $\text{Enc}_{\text{HPKE}}(\text{pk}_{\text{HPKE}}, \text{sk}_{\text{BTE}}^{(w^i)})$. Otherwise parse $\text{ek}_A^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w)}, \text{dk}_{\text{BTE}}^{(w)})_{w \in W}, (\cdot, \text{dk}_{\text{BTE}}^{(w^i)})$, and set $\text{dk}^{(w)} \leftarrow \text{Enc}_{\text{HPKE}}(\text{pk}_{\text{HPKE}}, \text{sk}_{\text{BTE}}^{(w)})$ and $\text{dek}^{(w)} \leftarrow \text{Enc}_{\text{HPKE}}(\text{pk}_{\text{HPKE}}, \text{dk}_{\text{BTE}}^{(w)})$ for $w \in W \cup \{w^i\}$. Set $\text{dk}^{(i)} \leftarrow \text{dk}^{(w^i)}$ and $\text{dek}^{(i)} \leftarrow (\text{dk}^{(w)}, \text{dek}^{(w)})_{w \in W}, (\perp, (\text{dek}^{(w^i)}))$ and return $\text{dk}^{(i)}, \text{dek}^{(i)}$.</p>
<p>$\text{DelEvo}(\text{dk}_{A \rightarrow B}^{(i)}, \text{dek}_{A \rightarrow B}^{(i)})$: Parse $\text{dk}_{A \rightarrow B}^{(i)}$ as $\text{dk}_{A \rightarrow B}^{(w^i)}$ and view $\text{dek}_{A \rightarrow B}^{(i)}$ organized as a stack of encrypted evolution keys. Set $\text{dk}_{A \rightarrow B}^{(w^{i+1})}, \text{dek}_{A \rightarrow B}^{(i+1)} \leftarrow \text{DFEval}(\text{dk}_{A \rightarrow B}^{(w^i)}, \text{dek}_{A \rightarrow B}^{(i)}, \text{Eval}_{\text{HPKE}}(\text{Der}_{\text{BTE}}, \cdot))$, and $\text{dk}_{A \rightarrow B}^{(i+1)} \leftarrow \text{dk}_{A \rightarrow B}^{(w^{i+1})}$. Return $\text{dk}_{A \rightarrow B}^{(i+1)}, \text{dek}_{A \rightarrow B}^{(i+1)}$.</p>
<p>$\text{DelDec}(\text{sk}_B^{(i)}, \text{dk}_{A \rightarrow B}^{(i)}, C_A)$: Parse $\text{sk}_B^{(i)}$ as $(\cdot, \text{sk}_{\text{HPKE}})$, set $\text{sk}_{\text{BTE}}^{(w^i)} \leftarrow \text{Dec}_{\text{HPKE}}(\text{sk}_{\text{HPKE}}, \text{dk}_{A \rightarrow B}^{(i)})$, and return $\text{Dec}_{\text{BTE}}(\text{sk}_{\text{BTE}}^{(w^i)}, C_A)$.</p>

Scheme 1. fs-DPKE scheme from BTE scheme and a compatible HPKE scheme.

scheme. So to obtain a fs-DPKE scheme without DelEvo algorithm, a compatible PKE scheme is sufficient. Yet, we will require the homomorphic properties later to achieve a suitable notion of adaptability regardless of the availability of DelEvo.

Similar to Canetti et al.'s construction, our fs-DPKE scheme inherits the fs-IND-CPA security from the BTE's IND-SN-CPA security.

Theorem 1. *If instantiated with an IND-SN-CPA secure BTE scheme and a IND-CPA secure HPKE scheme, then Scheme 1 is a fs-IND-CPA secure fs-DPKE.*

Proof. We prove the theorem using a sequence of games. We denote by W all the relevant nodes in the binary tree for period j . We note that the size of W is bounded by $\log_2(n)$. We index W as w_i for $i \in [|W|]$.

Game 0: The original game.

Game $1_{i,j}$ ($1 \leq i \leq q_{\text{Del}^h}, 1 \leq j \leq 2|W|$): As the previous game, but we replace all HPKE ciphertexts up to the j -th one in the i -th query with ciphertexts encrypting random plaintexts. That is, we modify the Del^h in the i -th query as follows:

$\text{Del}^{h'}(j, \text{sk}^{(0)}, \text{ek}^{(0)}, \text{pk}_i)$: Up to the j -th call to Enc_{HPKE} , encrypt a uniformly random value.

Transition $^{0 \rightarrow 1_{1,1}}$, **Transition** $^{1_{i,j} \rightarrow 1_{i,j+1}}$, **Transition** $^{1_{i,2|W|} \rightarrow 1_{i+1,1}}$: A distinguisher $\mathcal{D}^{0 \rightarrow 1_{1,1}}$ (respectively $\mathcal{D}^{1_{i,j} \rightarrow 1_{i,j+1}}$ or $\mathcal{D}^{1_{i,2|W|} \rightarrow 1_{i+1,1}}$) is an IND-CPA adversary against the HPKE scheme. We construct a reduction where we let \mathcal{C} be a IND-CPA challenger. We modify $\text{Del}^{h'}$ in the i -th query in the following way:

$\text{Del}^{h'}(j, \text{sk}^{(0)}, \text{ek}^{(0)}, \text{pk}_i)$: Simulate everything honestly, but on the j -th query choose r uniformly at random and run

$$c \leftarrow \mathcal{C}(\text{sk}_{\text{BTE}}^{(w_{j/2}-1)}, r) \text{ if } j \text{ is odd and } c \leftarrow \mathcal{C}(\text{ek}_{\text{BTE}}^{(w_{j/2})}, r) \text{ if } j \text{ is even,}$$

where $c \leftarrow \mathcal{C}(m_0, m_b)$ denotes a challenge ciphertext with respect to m_0 and m_1 .

Now, the bit b chosen by \mathcal{C} switches between the distributions of the Games.

In Game $1_{q_{\text{Del}^h}, 2|W|}$ all ciphertexts obtainable from $\text{Del}^{h'}$ are with respect to random values. Now, an adversary B winning Game $1_{q_{\text{Del}^h}, 2|W|}$ can be transformed into a IND-SN-CPA adversary A against the underlying BTE scheme:

1. When A is first started on $1^k, \ell$, choose $i^* \xleftarrow{R} [n]$ and output $w^{(i^*-1)}$.
2. When A is started on $\text{pk}_{\text{BTE}}, (\text{sk}^{(w)}, \text{dk}^{(w)})_{w \in W}$, compute $(\text{pk}_{\text{HPKE}}, \text{sk}_{\text{HPKE}}) \leftarrow \text{Gen}_{\text{HPKE}}(1^k)$. The secret key sk_{HPKE} is stored in the state st and we extend the public key to $\text{pk} \leftarrow (\text{pk}_{\text{BTE}}, \text{pk}_{\text{HPKE}})$. Now start B on the extended public key, i.e. $(j^*, \text{st}) \leftarrow B(1^k, n, \text{pk})$. If $i^* \neq j^*$, output a random bit and halt. Otherwise we have the secret-derivation key pairs of all nodes that are right siblings on the path from the root node to $w^{(j^*-1)}$ and (if they exist) all child nodes of $w^{(j^*-1)}$, hence we are able to simulate all oracle queries from B honestly. Similarly, we can compute $(\text{sk}^{(j^*)}, \text{dk}^{(j^*)})$ from the given keys. Thus we run $B^{\mathcal{O}}(\text{st}, \text{sk}^{(j^*)}, \text{dk}^{(j^*)})$ and forward its result.
3. When A is finally started on the challenge ciphertext, the ciphertext is simply forwarded to B and when B outputs the bit b , A returns b and halts.

When B is running within A and $j^* = i^*$, B has exactly the same view as in Game $1_{q_{\text{Gen}^h}, 2|W|}$. In this case the probability of A to win is exactly the same as the winning probability of B , and Game $1_{q_{\text{Gen}^h}, 2|W|}$ is computationally indistinguishable from the initial game. The random guess of i^* so that $i^* = j^*$ induces a loss of $\frac{1}{n}$, which is however bounded by a polynomial in the security parameter. \square

4.3 Constructing fs-PRE from fs-DPKE

Now we present a construction of a fs-PRE⁺-secure fs-PRE scheme from a fs-DPKE scheme. Therefore, we define additional properties of fs-DPKE and show that a fs-PRE can be directly obtained from a fs-DPKE. For our transformation to

work, we need to define an additional algorithm that allows us to homomorphically shift ciphertexts and delegation keys. That is, ciphertexts and delegation keys are modified in such a way that the delegation keys look like randomly distributed fresh keys, which are only useful to decrypt ciphertexts adapted to this key. Formally, we introduce an algorithm **Adapt** that enables this adaption:

Adapt(dk, C'): On input a delegation key dk , a ciphertext C , outputs an adapted delegation key dk' and ciphertext C' .

Since the delegation keys in our construction are encrypted BTE secret keys, we essentially adapt secret keys and ciphertexts from a BTE. We will see that this adaption is possible as long as the HPKE scheme used to encrypt the BTE keys provides a suitable homomorphism on the message space.

To adapt ciphertexts and delegation keys we extend correctness to additionally require that for any message M encrypted under the public key of A , any delegation key $dk_{A \rightarrow B}^{(i)}$, and any adapted delegation key-ciphertext pairs $(dk', C') \leftarrow \text{Adapt}(dk_{A \rightarrow B}^{(i)}, C_A)$, it holds that $M = \text{DelDec}_{\text{DPKE}}(sk_B^{(i)}, dk', C')$.

As security notion we introduce the fs-ADAP-IND-CPA notion, where the adversary may see multiple adapted delegation keys and ciphertexts, but the adversary should be unable to win an IND-CPA game for non-adapted ciphertexts. We give the formal definition of the security experiment in Experiment 7. This notion gives the delegator more control over the ciphertexts that should be readable for the delegatee. If given the delegation key, the delegatee can always decrypt all ciphertexts, but if just given an adapted delegation key, only a selected subset of ciphertexts is decryptable.

Experiment $\text{Exp}_{fs\text{-DPKE},A}^{\text{fs-adap-ind-cpa}}(1^k, n)$

$pp \leftarrow \text{Setup}(1^k), (pk, sk^{(0)}, ek^{(0)}) \leftarrow \text{Gen}(pp, n), b \xleftarrow{R} \{0, 1\}$
 $(j^*, pk^*, st) \leftarrow A(pp, n, pk)$
 $sk^{(j)}, ek^{(j)} \leftarrow \text{Evo}(sk^{(j-1)}, ek^{(j-1)})$ for $j \in [j^*], dk \leftarrow \text{Del}(sk^{(j^*)}, \perp, pk^*)$
 $(M_0, M_1, st) \leftarrow A^{\{\text{Adapt}(dk, \cdot)\}}(st)$
 $b^* \leftarrow A(st, \text{Enc}(pk, M_b, j^*))$
 if $b = b^*$ return 1, else return 0

Experiment 7. The fs-ADAP-IND-CPA security experiment for a fs-DPKE scheme.

Definition 13 (fs-ADAP-IND-CPA). For a polynomially bounded function $n(\cdot) > 1$, a PPT adversary A , we define the advantage function in the sense of fs-IND-CPA as

$$\text{Adv}_{fs\text{-DPKE},A}^{\text{fs-adap-ind-cpa}}(1^k, n(k)) := \left| \Pr \left[\text{Exp}_{\text{DPKE},A}^{\text{fs-adap-ind-cpa}}(1^k, n(k)) = 1 \right] - \frac{1}{2} \right|.$$

If for all $n(\cdot) > 1$, and any A there exists a negligible function ε such that $\text{Adv}_{fs\text{-DPKE},A}^{\text{fs-adap-ind-cpa}}(1^k, n(k)) < \varepsilon(k)$, then a fs-DPKE scheme is fs-ADAP-IND-CPA secure.

For Scheme 1, this adaption can be achieved solely from key-homomorphic properties of the BTE and homomorphic properties of the HPKE, respectively. Subsequently, we define the required homomorphisms. Our definitions are inspired by [2, 40]. We focus on schemes where the secret/derived key pairs, and public keys live in groups $(\mathbb{G}, +)$, and (\mathbb{H}, \cdot) , respectively. We will require two different properties: first, the public key is the image of the secret key under a group homomorphism, and second, given two secret keys with a known difference, we can map the binary tree of derived keys from one key to the other key. In other words, the difference in the keys propagates to the derived keys.

Definition 14. Let Ω be a BTE scheme with secret/derived key space $(\mathbb{G}, +)$ and public key space (\mathbb{H}, \cdot) .

1. The scheme Ω provides a secret-key-to-public-key homomorphism, if there exists an efficiently computable group homomorphism $\mu : \mathbb{G} \rightarrow \mathbb{H}$ such that for all $(\text{pk}, \text{sk}) \leftarrow \text{Gen}$, it holds that $\text{pk} = \mu(\text{sk})$.
2. The scheme Ω provides a derived-key homomorphism, if there exists a family of efficiently computable group homomorphisms $\nu^{(w)} : \mathbb{G} \rightarrow \mathbb{G}^2$ such that for all $(\text{pk}, \text{sk}^{(\varepsilon)}) \leftarrow \text{Gen}$, all nodes w it holds that $(\text{sk}^{(w0)}, \text{sk}^{(w1)}) = \nu^{(w)}(\text{sk}^{(w)})$ and for all messages M it holds that $\text{Dec}(\text{sk}^{(w)}, \text{Enc}(\text{pk}, M, w)) = M$.

We denote by Φ^+ the set of all possible secret key differences in \mathbb{G} . Alternatively, it is possible to view Φ^+ as set of functions representing all linear shifts in \mathbb{G} and we simply identify each shift by an element $\Delta \in \mathbb{G}$.

Definition 15. A BTE scheme Ω is called Φ^+ -key-homomorphic, if it provides both a secret-key-to-public-key homomorphism and a derived key homomorphism and an additional PPT algorithm Adapt , defined as:

$\text{Adapt}(\text{pk}, C, \Delta)$: On input a delegation key dk , a ciphertext C and a secret key difference Δ , outputs a public key pk' and a ciphertext C' .

such that for all $\Delta \in \Phi^+$, and all $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\dots)$, all message M , and all $C \leftarrow \text{Enc}(\text{pk}, M)$, and $(\text{pk}', C') \leftarrow \text{Adapt}(\text{pk}, C, \Delta)$ it holds that $\text{pk}' = \text{pk} \cdot \mu(\Delta)$ and $\text{Dec}(\text{sk}^{(w)} + \nu^{(w)}(\Delta), C') = M$.

Definition 16 (Adaptability of ciphertexts). A Φ^+ -key-homomorphic BTE scheme provides adaptability of ciphertexts, if for every security parameter $k \in \mathbb{N}$, any public parameters $\text{pp} \leftarrow \text{Setup}(1^k)$, every message M and every period j , it holds that $\text{Adapt}(\text{pk}, \text{Enc}(\text{pk}, M, j), \Delta)$ and $(\text{pk} \cdot \mu(\Delta), \text{Enc}(\text{pk} \cdot \mu(\Delta), M, j))$ as well as (sk, pk) and $(\text{sk}', \mu(\text{sk}'))$ are identically distributed, where $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(\text{pp}, n)$, $\text{sk}' \xleftarrow{R} \mathbb{G}$ and $\Delta \leftarrow \Phi^+$.

Next, we discuss the BTE from [15] with respect to our notion of ciphertext adaptability. We first recall the BTE scheme in Scheme 2 where BGen is a bilinear group generator. By [15, Proposition 1] this scheme is IND-SN-CPA secure if the decisional BDH assumption holds relative to BGen .

Now we show that Scheme 2 also provides adaptability of ciphertexts:

Setup(1^k): Run to $\text{BGGen}_p(1^k)$ to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q and a bilinear map e and select a random generator $P \in \mathbb{G}_1$. Set $\text{pp} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, e, q, P)$ and return pp .

Gen(pp, ℓ): Choose $\alpha \leftarrow \mathbb{Z}_q$ and set $Q \leftarrow \alpha \cdot P$. Set $\text{sk}^{(\varepsilon)} \leftarrow \alpha H(\varepsilon)$ and $\text{pk} \leftarrow (Q, H)$. Return $(\text{pk}, \text{sk}^{(\varepsilon)})$.

Der($\text{sk}^{(i)}$): Parse $\text{sk}^{(w)}$ as $(R_{w|1}, \dots, R_w, S_w)$. Choose $r_0, r_1 \xleftarrow{R} \mathbb{Z}_q$ and set $R_{wi} \leftarrow r_i P$ and $S_{wi} \leftarrow S_w + r_i \cdot H(wi)$ for $i \in [2]$ and return $((R_{w|1}, \dots, R_w, R_{w0}, S_{w0}), (R_{w|1}, \dots, R_w, R_{w1}, S_{w1}))$.

Enc(pk, M, i): Choose $\gamma \leftarrow \mathbb{Z}_q$ and set $C \leftarrow (\gamma \cdot P, \gamma \cdot H(w|1), \dots, \gamma \cdot H(w), M \cdot e(Q, \gamma \cdot H(\varepsilon)))$. Return C .

Dec($\text{sk}^{(w)}, C$): Parse $\text{sk}^{(w)}$ as $(R_{w|1}, \dots, R_w, S_w)$ and C as (U_0, \dots, U_t, V) . Return $M = V/d$ where

$$d = \frac{e(U_0, S_w)}{\prod_{i=1}^t e(R_{w|i}, U_i)}.$$

Scheme 2. BTE scheme from [15]

Lemma 3. *Scheme 2 provides adaptability of ciphertexts under shared H .*

Proof. We show the existence of the homomorphisms and give the **Adapt** algorithm. Note that the master secret key can easily be viewed as containing α , hence, the secret-to-public-key homomorphism is simply $\mu : \alpha \mapsto \alpha P$. As the Der algorithm simply computes sums, the existence of the homomorphism is clear.

We now show the existence of **Adapt**:

Adapt(pk, C, Δ): Parse pk as (Q, ℓ, H) and C as (U_0, \dots, U_t, V) . Let $Q' \leftarrow Q + \Delta \cdot P$ and set $\text{pk}' \leftarrow (Q', \ell, H)$. Let $V' \leftarrow Ve(U_0, \Delta \cdot H(\varepsilon))$ and set $C' \leftarrow (U_0, \dots, U_t, V')$ and return (pk', C') .

The adapted C' ciphertext is an encryption of the original message under the public key $Q' = Q + \Delta \cdot P$. □

Now, given any Φ^+ -key-homomorphic BTE scheme, it can be turned into an adaptable fs-DPKE by defining **Adapt** in a publicly computable way as follows:

Adapt($\text{dk}_{A \rightarrow B}^{(i)}, C$): Sample $\Delta \xleftarrow{R} \Phi^+$ and compute $\text{dk}_\Delta \leftarrow \text{Enc}_{\text{HPKE}}(\text{pk}_B, \nu^{(w^i)}(\Delta))$, and then $\text{dk}' \leftarrow \text{Eval}_{\text{HPKE}}(+, \text{dk}_{A \rightarrow B}^{(i)}, \text{dk}_\Delta)$. Set $(\cdot, C') \leftarrow \text{Adapt}_{\text{BTE}}(\text{pk}_A, C, \Delta)$. Return (dk', C') .

Theorem 2. *If in addition to the premise in Theorem 1 the BTE scheme also provides adaptability of ciphertexts, then Scheme 1 is a fs-ADAP-IND-CPA secure fs-DPKE scheme.*

Proof. We prove this theorem with a sequence of games.

Game 0: The original game.

Game 1: We modify the simulation of the **Adapt** oracle as follows, where we denote the modified oracle by **Adapt'**:

Adapt'($\boxed{\text{sk}^{(i)}, \text{pk}, \text{pk}^*}, C$): Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w^i)}, \cdot)$, pk as $(\text{pk}_{\text{BTE}}, \cdot)$, and pk^* as $(\cdot, \text{pk}_{\text{HPKE}}^*)$. Choose $\Delta \leftarrow \Phi^+$, run

$$\boxed{\text{dk}' \leftarrow \text{Enc}_{\text{HPKE}}(\text{pk}_{\text{HPKE}}^*, \text{sk}_{\text{BTE}}^{(w^i)} + \nu^{(w^i)}(\Delta))}$$
 and

$$\boxed{C' \leftarrow \text{Enc}_{\text{BTE}}(\text{pk} \cdot \mu(\Delta), \text{Dec}_{\text{BTE}}(\text{sk}^{(i)}, C), i)}$$
. Return (dk', C') .

Transition $^{0 \rightarrow 1}$: The distributions of Game 0 and Game 1 are indistinguishable under the BTE's adaptability of ciphertexts.

Game 2: We further modify the simulation of **Adapt'** as follows:

Adapt'($\boxed{\text{sk}^{(i)}, \text{pk}^*}, C$): Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{BTE}}^{(w^i)}, \cdot)$, pk as $(\text{pk}_{\text{BTE}}, \cdot)$, and pk^* as $(\cdot, \text{pk}_{\text{HPKE}}^*)$. Choose $\boxed{\text{pk}'_{\text{BTE}}, \text{sk}'_{\text{BTE}}^{(\varepsilon)}, \text{ek}'_{\text{BTE}}^{(\varepsilon)} \leftarrow \text{Gen}_{\text{BTE}}}$ and evolve the secret key to period i , run

$$\boxed{\text{dk}' \leftarrow \text{Enc}_{\text{HPKE}}(\text{pk}_{\text{HPKE}}^*, \text{sk}'_{\text{BTE}}^{(w^i)})}$$
 and

$$\boxed{C' \leftarrow \text{Enc}_{\text{BTE}}(\text{pk}'_{\text{BTE}}, \text{Dec}_{\text{BTE}}(\text{sk}^{(i)}, C), i)}$$
. Return (dk', C') .

Transition $^{1 \rightarrow 2}$: The change is conceptual.

In Game 2 all the secret BTE keys the adversary gets are chosen independently from the challenge key. Hence, Game 2 is a standard IND-CPA game and thus the success probability of Game 2 is negligible by Theorem 1. \square

Now, given an adaptable fs-DPKE scheme, we use the **Adapt** algorithm to obtain a fs-PRE $^+$ secure fs-PRE scheme. While the algorithms **Setup**, **Gen**, **Evo**, $\text{Enc}^{(i)}$, and $\text{Dec}^{(i)}$ can simply be lifted from the fs-DPKE scheme, we note that for each period j in the fs-PRE scheme, we use two periods, i.e., $2j - 1$ and $2j$, of the fs-DPKE scheme. The period $2j - 1$ is used for level 1 ciphertexts whereas the period $2j$ is used for level 2 ciphertexts¹³. We use Del_{DPKE} and $\text{DelEvo}_{\text{DPKE}}$ for **ReGen** and **ReEvo**, respectively. For the re-encryption algorithm **ReEnc**, we apply **Adapt**. $\text{Dec}^{(1)}$ for re-encrypted ciphertexts then decrypts the ciphertext by running $\text{DelDec}_{\text{DPKE}}$ on the adapted delegation key and ciphertext. The full scheme is presented in Scheme 3.

We prove that our scheme is both fs-IND-CPA-1 and fs-IND-CPA-2 secure. Both security notions follow from the fs-IND-CPA security of the underlying fs-DPKE scheme. In contrast, to achieve fs-RIND-CPA, we require an fs-ADAP-IND-CPA fs-DPKE scheme.

Theorem 3. *If instantiated with a fs-IND-CPA and fs-ADAP-IND-CPA secure fs-DPKE scheme, Scheme 3 is a fs-PRE $^+$ -secure fs-PRE scheme.*

¹³ One can see the keys for period $2j$ as weak keys in the sense of [4, Third Attempt] whereas the keys for period $2j - 1$ constitute the master secret keys.

Let $(\text{Setup}_{\text{DPKE}}, \text{Gen}_{\text{DPKE}}, \text{Evo}_{\text{DPKE}}, \text{Del}_{\text{DPKE}}, \text{Enc}_{\text{DPKE}}, \text{Dec}_{\text{DPKE}}, \text{Adapt}_{\text{DPKE}})$ be fs-DPKE scheme with adaption of ciphertexts and delegation keys.

$\text{Setup}(1^k)$: Return $\text{Setup}_{\text{DPKE}}(1^k)$.

$\text{Gen}(\text{pp}, n)$: Set $(\text{pk}_{\text{DPKE}}, \text{sk}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(0)}) \leftarrow \text{Gen}_{\text{DPKE}}(\text{pp}, 2n+1)$, obtain $(\text{sk}_{\text{DPKE}}^{(1)}, \text{ek}_{\text{DPKE}}^{(1)}) \leftarrow \text{Evo}_{\text{DPKE}}(\text{sk}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(0)})$, and return $(\text{pk}_{\text{DPKE}}, \text{sk}^{(0)}, \text{ek}^{(0)})$, where

$$\text{sk}^{(0)} \leftarrow (\text{sk}_{\text{DPKE}}^{(0)}, \text{sk}_{\text{DPKE}}^{(1)}), \text{ek}^{(0)} \leftarrow (\text{ek}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(1)}).$$

$\text{Evo}(\text{sk}^{(i)}, \text{ek}^{(i)})$: Parse $(\text{sk}^{(i)}, \text{ek}^{(i)})$ as $((\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)}), (\text{ek}_{\text{DPKE}}^{(2i)}, \text{ek}_{\text{DPKE}}^{(2i+1)}))$ and return $(\text{sk}^{(i+1)}, \text{ek}^{(i+1)}) = (\text{sk}_{\text{DPKE}}^{(2i+2)}, \text{sk}_{\text{DPKE}}^{(2i+3)}), (\text{ek}_{\text{DPKE}}^{(2i+2)}, \text{ek}_{\text{DPKE}}^{(2i+3)}))$, where

$$(\text{sk}_{\text{DPKE}}^{(2i+1+j)}, \text{ek}_{\text{DPKE}}^{(2i+1+j)}) \leftarrow \text{Evo}_{\text{DPKE}}(\text{sk}_{\text{DPKE}}^{(2i+j)}, \text{ek}_{\text{DPKE}}^{(2i+j)}) \text{ for } j \in [2].$$

$\text{Enc}^{(1)}(\text{pk}, M, i)$: Return $\text{Enc}_{\text{DPKE}}(\text{pk}, M, 2i)$.

$\text{Enc}^{(2)}(\text{pk}, M, i)$: Return $\text{Enc}_{\text{DPKE}}(\text{pk}, M, 2i+1)$.

$\text{Dec}^{(1)}(\text{sk}^{(i)}, C)$: Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)})$ and return $\text{Dec}_{\text{DPKE}}(\text{sk}^{(2i)}, C)$ if C was not re-encrypted. Otherwise parse C as (C_1, rk) and return $\text{DelDec}_{\text{DPKE}}(\text{sk}^{(2i+1)}, \text{rk}, C_1)$.

$\text{Dec}^{(2)}(\text{sk}^{(i)}, C)$: Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)})$ and return $\text{Dec}_{\text{DPKE}}(\text{sk}^{(2i+1)}, C)$.

$\text{ReGen}(\text{sk}_A^{(i)}, \text{ek}_A^{(i)}, \text{pk}_B)$: Parse $(\text{sk}^{(i)}, \text{ek}^{(i)})$ as $((\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)}), (\text{ek}_{\text{DPKE}}^{(2i)}, \text{ek}_{\text{DPKE}}^{(2i+1)}))$, and $\text{Del}_{\text{DPKE}}(\text{sk}_A^{(2i+1)}, \text{ek}_A^{(2i+1)}, \text{pk}_B)$.

$\text{ReEvo}(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)})$: Return $\text{DelEvo}_{\text{DPKE}}(\text{DelEvo}_{\text{DPKE}}(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)}))$.

$\text{ReEnc}(\text{rk}_{A \rightarrow B}^{(i)}, C_A)$: Choose $\tau \xleftarrow{R} \mathbb{G}$ and return $\text{Adapt}_{\text{DPKE}}(\text{rk}_{A \rightarrow B}^{(i)}, C_A, \tau)$.

Scheme 3. fs-PRE scheme from an adaptable fs-DPKE scheme.

Proof. Informally speaking, the security experiment for fs-IND-CPA-2 with a fixed period j^* corresponds to the fs-IND-CPA experiment for fs-DPKE for period $2j^*$. We can build a straightforward reduction from an adversary against fs-IND-CPA-2, A_2 to fs-IND-CPA for fs-DPKE:

- When started on pp, n and pk , run $(j^*, \text{st}) \leftarrow A_2(\text{pp}, \lceil \frac{n}{2} + 1 \rceil, \text{pk})$. Set $j' \leftarrow 2j^*$ and return (j', st) .
- When started on $\text{st}, \text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$, we simulate the $\text{ReGen}^{(h)}$ and $\text{ReGen}^{(h')}$ oracles using $\text{Del}^{(h)}$ and $\text{Del}^{(h')}$. Indeed, $\text{Del}^{(h)}$ and $\text{Del}^{(h')}$ return delegation keys for period $j' - 1 = 2j^* - 1$, which are re-encryption keys for period $j^* - 1$. Using Evo we evolve $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to period $j' + 1$. Set $(\text{sk}^{(j^*)}, \text{ek}^{(j^*)}) \leftarrow ((\text{sk}_{\text{DPKE}}^{(j')}, \text{sk}_{\text{DPKE}}^{(j'+1)}), (\text{ek}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j'+1)}))$ and start A_2 on $\text{st}, (\text{sk}^{(j^*)}, \text{ek}^{(j^*)})$ and simply forward the result.
- Finally, when started on st and $C_{j'-1}, C_{j'-1}$ is a level 2 ciphertext for $j^* - 1$. Hence we start A_2 on the ciphertext and return its' result.

To show fs-IND-CPA-1 security, we perform a similar reduction:

- When started on pp, n and pk , run $(j^*, \text{st}) \leftarrow A_1(\text{pp}, \lceil \frac{n}{2} + 1 \rceil, \text{pk})$. Set $j' \leftarrow 2j^* - 1$ and return (j', st) .

- When started on $\text{st}, \text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$, we simulate the $\text{ReGen}^{(h)}$ and $\text{ReGen}^{(h')}$ oracles using $\text{Del}^{(h)}$ and $\text{Del}^{(h')}$ and by running DelEvo on the result. Indeed, $\text{Del}^{(h)}$ and $\text{Del}^{(h')}$ return delegation keys for period $j' - 1 = 2j^* - 2$, hence after applying DelEvo we obtain re-encryption keys for period $j^* - 1$. $\text{ReGen}^{(d)}$ is simulated honestly by delegating $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to a dishonest user. Using Evo we evolve $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to period $j' + 2$. Set $(\text{sk}^{(j*)}, \text{ek}^{(j*)}) \leftarrow ((\text{sk}_{\text{DPKE}}^{(j'+1)}, \text{sk}_{\text{DPKE}}^{(j'+2)}), (\text{ek}_{\text{DPKE}}^{(j'+1)}, \text{ek}_{\text{DPKE}}^{(j'+2)}))$ and start A_1 on $\text{st}, (\text{sk}^{(j*)}, \text{ek}^{(j*)})$ and simply forward the result.
- Finally, when started on st and $C_{j'-1}, C_{j'-1}$ is a level 1 ciphertext for $j^* - 1$. Hence we start A_1 on the ciphertext and return its' result.

To show receiver-IND-CPA security we build an fs-ADAP-IND-CPA adversary against the fs-DPKE scheme. The fs-RIND-CPA adversary is denoted as A_r .

- When started on pp, n and pk , run $(j^*, \text{st}) \leftarrow A_r(\text{pp}, \lceil \frac{n}{2} + 1 \rceil, \text{pk})$. Set $j' \leftarrow 2j^* + 1$ and return (j', st) .
- When started on st , we can simulate ReEnc honestly using Adapt .
- When started on st and C , the ciphertext is a level 2 ciphertext for period j^* , hence we return $A_r(\text{st}, C)$.

Note that all values are consistently distributed in all three reductions. \square

4.4 Separating fs-PRE⁻ from fs-PRE⁺

To expand on the gap between fs-PRE⁺ and fs-PRE⁻ schemes and to provide an explicit separation, we construct a counterexample. In particular, it is clear that every scheme that satisfies fs-PRE⁺ also satisfies fs-PRE⁻. For our separation we now present a scheme that is fs-PRE⁻ but trivially violates fs-PRE⁺. The scheme is also built from a fs-DPKE scheme and presented in Scheme 4. In this scheme however, ReEnc simply embeds the delegation key in the re-encrypted ciphertext. The shortcomings of this construction compared to Scheme 3 are obvious: once the receiver is presented with one valid re-encrypted ciphertext, it can recover the delegation key from that ciphertext and can decrypt all level 2 ciphertexts for this period.

In the following Theorem, we first show that Scheme 4 is indeed fs-PRE⁻ secure, i.e., satisfies fs-IND-CPA-1 and fs-IND-CPA-2 security, but trivially does not satisfy fs-RIND-CPA security and thus is not fs-PRE⁺ secure.

Theorem 4. *Scheme 4 when instantiated with a fs-IND-CPA secure fs-DPKE scheme satisfies fs-IND-CPA-1 and fs-IND-CPA-2 security, but not fs-RIND-CPA security.*

Proof. We follow the same strategy as for Theorem 3 to show fs-IND-CPA-2.

- When started on pp, n and pk , run $(j^*, \text{st}) \leftarrow A_2(\text{pp}, \lceil \frac{n}{2} + 1 \rceil, \text{pk})$. Set $j' \leftarrow 2j^*$ and return (j', st) .

<p>Let $(\text{Setup}_{\text{DPKE}}, \text{Gen}_{\text{DPKE}}, \text{Evo}_{\text{DPKE}}, \text{Del}_{\text{DPKE}}, \text{Enc}_{\text{DPKE}}, \text{Dec}_{\text{DPKE}})$ be fs-DPKE scheme.</p> <p><u>$\text{Setup}(1^k)$</u> : Return $\text{Setup}_{\text{DPKE}}(1^k)$.</p> <p><u>$\text{Gen}(\text{pp}, n)$</u> : Set $(\text{pk}_{\text{DPKE}}, \text{sk}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(0)}) \leftarrow \text{Gen}_{\text{DPKE}}(\text{pp}, 2n+1)$, obtain $(\text{sk}_{\text{DPKE}}^{(1)}, \text{ek}_{\text{DPKE}}^{(1)}) \leftarrow \text{Evo}_{\text{DPKE}}(\text{sk}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(0)})$, and return $(\text{pk}_{\text{DPKE}}, \text{sk}^{(0)}, \text{ek}^{(0)})$, where</p> $\text{sk}^{(0)} \leftarrow (\text{sk}_{\text{DPKE}}^{(0)}, \text{sk}_{\text{DPKE}}^{(1)}), \text{ek}^{(0)} \leftarrow (\text{ek}_{\text{DPKE}}^{(0)}, \text{ek}_{\text{DPKE}}^{(1)}).$ <p><u>$\text{Evo}(\text{sk}^{(i)}, \text{ek}^{(i)})$</u> : Parse $(\text{sk}^{(i)}, \text{ek}^{(i)})$ as $((\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)}), (\text{ek}_{\text{DPKE}}^{(2i)}, \text{ek}_{\text{DPKE}}^{(2i+1)}))$ and return $(\text{sk}^{(2i+1)}, \text{ek}^{(2i+1)}) = (\text{sk}_{\text{DPKE}}^{(2i+2)}, \text{sk}_{\text{DPKE}}^{(2i+3)}), (\text{ek}_{\text{DPKE}}^{(2i+2)}, \text{ek}_{\text{DPKE}}^{(2i+3)}))$, where</p> $(\text{sk}_{\text{DPKE}}^{(2i+1+j)}, \text{ek}_{\text{DPKE}}^{(2i+1+j)}) \leftarrow \text{Evo}_{\text{DPKE}}(\text{sk}_{\text{DPKE}}^{(2i+j)}, \text{ek}_{\text{DPKE}}^{(2i+j)}) \text{ for } j \in [2].$ <p><u>$\text{Enc}^{(1)}(\text{pk}, M, i)$</u> : Return $\text{Enc}_{\text{DPKE}}(\text{pk}, M, 2i)$.</p> <p><u>$\text{Enc}^{(2)}(\text{pk}, M, i)$</u> : Return $\text{Enc}_{\text{DPKE}}(\text{pk}, M, 2i+1)$.</p> <p><u>$\text{Dec}^{(1)}(\text{sk}^{(i)}, C)$</u> : Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)})$ and return $\text{Dec}_{\text{DPKE}}(\text{sk}^{(2i)}, C)$ if C was not re-encrypted. Otherwise parse C as (C_1, rk) and return $\text{DelDec}_{\text{DPKE}}(\text{sk}^{(2i+1)}, \text{rk}, C_1)$.</p> <p><u>$\text{Dec}^{(2)}(\text{sk}^{(i)}, C)$</u> : Parse $\text{sk}^{(i)}$ as $(\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)})$ and return $\text{Dec}_{\text{DPKE}}(\text{sk}^{(2i+1)}, C)$.</p> <p><u>$\text{ReGen}(\text{sk}_A^{(i)}, \text{ek}_A^{(i)}, \text{pk}_B)$</u> : Parse $(\text{sk}^{(i)}, \text{ek}^{(i)})$ as $((\text{sk}_{\text{DPKE}}^{(2i)}, \text{sk}_{\text{DPKE}}^{(2i+1)}), (\text{ek}_{\text{DPKE}}^{(2i)}, \text{ek}_{\text{DPKE}}^{(2i+1)}))$, and return $(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)})$, where</p> $(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)}) \leftarrow \text{Del}_{\text{DPKE}}(\text{sk}_A^{(2i+1)}, \text{ek}_A^{(2i+1)}, \text{pk}_B).$ <p><u>$\text{ReEvo}(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)})$</u> : Return $\text{DelEvo}_{\text{DPKE}}(\text{DelEvo}_{\text{DPKE}}(\text{rk}_{A \rightarrow B}^{(i)}, \text{rek}_{A \rightarrow B}^{(i)}))$.</p> <p><u>$\text{ReEnc}(\text{rk}_{A \rightarrow B}^{(i)}, C_A)$</u> : Return $(C_A, \text{rk}_{A \rightarrow B}^{(i)})$.</p>

Scheme 4. fs-PRE scheme from a fs-DPKE scheme without adaption.

- When started on $\text{st}, \text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$, we simulate the ReGen^h and $\text{ReGen}^{h'}$ oracles using Del^h and $\text{Del}^{h'}$. Indeed, Del^h and $\text{Del}^{h'}$ return delegation keys for period $j' - 1 = 2j^* - 1$, which are re-encryption keys for period $j^* - 1$. Using Evo we evolve $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to period $j' + 1$. Set $(\text{sk}^{(j^*)}, \text{ek}^{(j^*)}) \leftarrow ((\text{sk}_{\text{DPKE}}^{(j')}, \text{sk}_{\text{DPKE}}^{(j'+1)}), (\text{ek}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j'+1)}))$ and start A_2 on $\text{st}, (\text{sk}^{(j^*)}, \text{ek}^{(j^*)})$ and simply forward the result.
- Finally, when started on st and $C_{j'-1}, C_{j'-1}$ is a level 2 ciphertext for $j^* - 1$. Hence we start A_2 on the ciphertext and return its' result.

To show fs-IND-CPA-1 security, we perform a similar reduction:

- When started on pp, n and pk , run $(j^*, \text{st}) \leftarrow A_1(\text{pp}, \lceil \frac{n}{2} + 1 \rceil, \text{pk})$. Set $j' \leftarrow 2j^* - 1$ and return (j', st) .
- When started on $\text{st}, \text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$, we simulate the ReGen^h and $\text{ReGen}^{h'}$ oracles using Del^h and $\text{Del}^{h'}$ and by running DelEvo on the result. Indeed, Del^h and $\text{Del}^{h'}$ return delegation keys for period $j' - 1 = 2j^* - 2$, hence after applying DelEvo we obtain re-encryption keys for period $j^* - 1$. ReGen^d

is simulated honestly by delegating $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to a dishonest user. Using Evo we evolve $\text{sk}_{\text{DPKE}}^{(j')}, \text{ek}_{\text{DPKE}}^{(j')}$ to period $j' + 2$. Set $(\text{sk}^{(j*)}, \text{ek}^{(j*)}) \leftarrow ((\text{sk}_{\text{DPKE}}^{(j'+1)}, \text{sk}_{\text{DPKE}}^{(j'+2)}), (\text{ek}_{\text{DPKE}}^{(j'+1)}, \text{ek}_{\text{DPKE}}^{(j'+2)}))$ and start A_1 on $\text{st}, (\text{sk}^{(j*)}, \text{ek}^{(j*)})$ and simply forward the result.

- Finally, when started on st and $C_{j'-1}, C_{j'-1}$ is a level 1 ciphertext for $j^* - 1$. Hence we start A_1 on the ciphertext and return its' result.

Following the initial observation on the recoverability of delegation keys, an receiver-IND-CPA adversary is straightforward to define:

- When started on pp, n and pk , honestly generate a key $(\text{pk}^*, \text{sk}^{(0)}, \text{ek}^{(0)}) \leftarrow \text{Gen}(\text{pp}, n)$ and store it in st . Choose $j^* \xleftarrow{R} [n]$ and store it together with pk in st , and return $(j^*, \text{pk}^*, \text{st})$.
- When started on st to output the challenge messages, choose $M_0, M_1, M_2 \xleftarrow{R} \mathcal{M}$. Invoke the ReEnc oracle as $(\cdot, \text{dk}) \leftarrow \text{ReEnc}(\text{rk}, \text{Enc}^{(2)}(\text{pk}, M_2, j^*))$ and store M_0, M_1, dk in st . Return M_0, M_1, st .
- Now when started on st and the challenge ciphertext C , use dk stored in st and obtain $M \leftarrow \text{DelDec}_{\text{DPKE}}(\text{sk}^{(2j^*+1)}, \text{dk}, C)$. Check for which $i \in \{0, 1\}$ $M = M_i$ and return i .

Regardless of the chosen period the adversary always wins, rendering the scheme insecure with respect to the fs-RIND-CPA notion. \square

From this theorem we obtain the following corollary:

Corollary 2. *fs-PRE⁺ is a strictly stronger notion than fs-PRE⁻.*

Note that this also shows that for conventional PRE scheme there is a separation between the classical security notion of PRE (PRE⁻) as defined by Ateniese et al. and the PRE⁺ notion.

Acknowledgments. Supported by H2020 project PRISMACLOUD, grant agreement n°644962 and by H2020 project CREDENTIAL, grant agreement n°653454. We thank all anonymous reviewers for their valuable comments.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 418–433. Springer, Heidelberg (2002). <https://doi.org/10.1007/3-540-46035-7.28>
2. Applebaum, B., Harnik, D., Ishai, Y.: Semantic security under related-key attacks and applications. In: ICS (2011)
3. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-00862-7.19>

4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS (2005)
5. Ateniese, G., Fu, K., 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)
6. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) *CRYPTO 1999*. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_28
7. Bellare, M., Yee, B.: Forward-security in private-key cryptography. In: Joye, M. (ed.) *CT-RSA 2003*. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36563-X_1
8. Berners-Lee, E.: Improved security notions for proxy re-encryption to enforce access control. In: *LATINCRYPT* (2017)
9. 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>
10. Blazy, O., Bultel, X., Lafourcade, P.: Two secure anonymous proxy-based data storages. In: *SECRYPT* (2016)
11. 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
12. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) *CRYPTO 2005*. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_16
13. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: *CCS* (2006)
14. Borceaa, C., Gupta, A.B.D., Polyakova, Y., Rohloff, K., Ryana, G.: PICADOR: end-to-end encrypted publish-subscribe information distribution with proxy re-encryption. *Future Gener. Comput. Syst.* **71**, 177–191 (2016)
15. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_16
16. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: *CCS* (2007)
17. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-ciphertext secure fully homomorphic encryption. In: Fehr, S. (ed.) *PKC 2017*. LNCS, vol. 10175, pp. 213–240. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_8
18. 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
19. Chandran, N., Chase, M., Vaikuntanathan, V.: Functional re-encryption and collusion-resistant obfuscation. In: Cramer, R. (ed.) *TCC 2012*. LNCS, vol. 7194, pp. 404–421. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_23
20. Cohen, A.: What about Bob? The inadequacy of CPA security for proxy re-encryption. *Cryptology ePrint Archive*, Report 2017/785 (2017)

21. Cohen, A., Holmgren, J., Nishimaki, R., Vaikuntanathan, V., Wichs, D.: Watermarking cryptographic capabilities. In: STOC (2016)
22. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_12
23. Fan, X., Liu, F.H.: Proxy re-encryption and re-signatures from lattices. Cryptology ePrint Archive, Report 2017/456 (2017)
24. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC (2009)
25. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34
26. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_19
27. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: IEEE S&P (2015)
28. Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_5
29. Günther, F., Hale, B., Jager, T., Lauer, S.: 0-RTT key exchange with full forward secrecy. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 519–548. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_18
30. Hanaoka, G., Kawai, Y., Kunihiko, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen ciphertext secure proxy re-encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_22
31. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. *J. Cryptol.* **24**, 694–719 (2011)
32. Libert, B., Vergnaud, D.: Tracing malicious proxies in proxy re-encryption. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 332–353. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_22
33. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_21
34. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. Inf. Theory* **57**, 1786–1802 (2011)
35. Myers, S., Shull, A.: Efficient hybrid proxy re-encryption for practical revocation and key rotation. Cryptology ePrint Archive, Report 2017/833 (2017)
36. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.* **20**(4), 14 (2017)
37. Ren, Y., Gu, D., Wang, S., Zhang, X.: Hierarchical identity-based proxy re-encryption without random oracles. *Int. J. Found. Comput. Sci.* **21**(6), 1049–1063 (2010)
38. Sakai, R., Furukawa, J.: Identity-based broadcast encryption. IACR Cryptology ePrint Archive (2007)
39. Tang, Q.: Type-based proxy re-encryption and its construction. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 130–144. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_11

40. Tessaro, S., Wilson, D.A.: Bounded-collusion identity-based encryption from semantically-secure public-key encryption: generic constructions with short ciphertexts. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 257–274. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_15
41. Weng, J., Yang, Y., Tang, Q., Deng, R.H., Bao, F.: Efficient conditional proxy re-encryption with chosen-ciphertext security. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 151–166. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04474-8_13
42. Xu, P., Xu, J., Wang, W., Jin, H., Susilo, W., Zou, D.: Generally hybrid proxy re-encryption: a secure data sharing among cryptographic clouds. In: AsiaCCS (2016)

Encryption with Bad Randomness



Hedged Nonce-Based Public-Key Encryption: Adaptive Security Under Randomness Failures

Zhengan Huang¹, Junzuo Lai^{2,3(✉)}, Wenbin Chen¹, Man Ho Au⁴, Zhen Peng⁵,
and Jin Li^{1(✉)}

¹ School of Computer Science, Guangzhou University, Guangzhou, China
zhahuang.sjtu@gmail.com, cwb2011@gzhu.edu.cn, jinli71@gmail.com

² College of Information Science and Technology, Jinan University,
Guangzhou, China
laijunzuo@gmail.com

³ State Key Laboratory of Cryptology, Beijing, China

⁴ Department of Computing, The Hong Kong Polytechnic University,
Hung Hom, Hong Kong
csallen@comp.polyu.edu.hk

⁵ Westone Cryptologic Research Center, Beijing, China
peng.zhen@westone.com.cn

Abstract. Nowadays it is well known that randomness may fail due to bugs or deliberate randomness subversion. As a result, the security of traditional public-key encryption (PKE) cannot be guaranteed any more. Currently there are mainly three approaches dealing with the problem of randomness failures: deterministic PKE, hedged PKE, and nonce-based PKE. However, these three approaches only apply to different application scenarios respectively. Since the situations in practice are dynamic and very complex, it's almost impossible to predict the situation in which a scheme is deployed, and determine which approach should be used beforehand.

In this paper, we initiate the study of hedged security for nonce-based PKE, which adaptively applies to the situations whenever randomness fails, and achieves the best-possible security. Specifically, we lift the hedged security to the setting of nonce-based PKE, and formalize the notion of chosen-ciphertext security against chosen-distribution attacks (IND-CDA2) for nonce-based PKE. By presenting two counterexamples, we show a separation between our IND-CDA2 security for nonce-based PKE and the original NBP1/NBP2 security defined by Bellare and Tackmann (EUROCRYPT 2016). We show two nonce-based PKE constructions meeting IND-CDA2, NBP1 and NBP2 security simultaneously. The first one is a concrete construction in the random oracle model, and the second one is a generic construction based on a nonce-based PKE scheme and a deterministic PKE scheme.

Keywords: Hedged security · Nonce-based public-key encryption
Deterministic public-key encryption · Randomness failures

1 Introduction

Background. It is well known that randomness plays a key role in cryptography. For most cryptographic constructions, their security is guaranteed on condition that the random coins employed are uniformly and independently chosen. For example, IND-CCA security [19], one universally accepted security notion for PKE, requires that the randomness employed during the encryption is uniformly chosen and independent of any other elements. However, randomness may fail because of bugs or randomness subversion. Recently, it is well-known that the randomness failures are actual threats, and bring new challenges to cryptographic constructions and information security products.

As far as we know, there are mainly three kinds of PKE which have been proposed to provide good privacy under randomness failures. The first one is *deterministic PKE* (D-PKE) [1, 4, 9], where the encryption algorithm does not need to use any randomness for encryption, and its security is guaranteed on condition that the messages have high min-entropy. D-PKE was proposed to provide fast search on encrypted data at first. Since the encryption does not use randomness, D-PKE is an important class of PKE dealing with the subsequently revealed problem of randomness subversion. The second one is *hedged PKE* (H-PKE) [2, 5], which can be seen as an extension of D-PKE. For hedged PKE, the encryption algorithm is randomized, and its security is guaranteed only if the messages and the randomness *jointly* have high min-entropy. The third one is *nonce-based PKE* (N-PKE) [8], the encryption algorithm of which is randomized, and the messages can be arbitrarily chosen. For each encryption, instead of taking fresh randomness, the encryption algorithm takes a uniform seed, which can be used repeatedly, and a nonce as input. A significant benefit brought by N-PKE is that it's not necessary for the senders to generate fresh, uniform and independent randomness at every encryption. The security of N-PKE is guaranteed as long as either the seed is confidential and the message-nonce pairs do not repeat, or the seed is exposed but the nonces are unpredictable.

The above three approaches focus on different scenarios. D-PKE is only suitable for the situations that the messages have sufficient min-entropy. H-PKE applies to the situations that the messages and the randomness have jointly sufficient min-entropy. Generally speaking, both of these two approaches require that the messages are independent of the public keys. N-PKE just applies to the case that either the seed or the nonces can provide sufficient randomness. Besides these three kinds of PKE schemes, currently the most commonly used ones in practice are the traditional PKE schemes (i.e., the security is guaranteed assuming that the randomness is good, and the messages can be arbitrarily chosen), such as RSA [18, 22].

However, unfortunately none of the aforementioned approaches is able to provide good privacy in all application scenarios. The messages we want to encrypt regularly do not have sufficient min-entropy [12] and sometimes may depend on the public key, and the randomness may fail because of bugs or deliberate randomness subversion [13, 15]. These facts limit the application of D-PKE and H-PKE. On the other hand, N-PKE can provide good privacy only if either

the seed or the nonces have sufficient min-entropy from the adversaries' point of view. If one uses N-PKE, when both the seed and the nonces do not have sufficient min-entropy, the security of the scheme cannot be guaranteed. These facts limit the application of N-PKE. More importantly, it's almost unrealistic to determine beforehand which kinds of PKE should be used because the situations in which the scheme is deployed are dynamic.

Hedged security for nonce-based PKE. In this paper, we formalize the notions of hedged security for nonce-based PKE, and provide some constructions. N-PKE schemes achieving our hedged security are able to adaptively apply to the situations whenever randomness fails, and achieve the best-possible security. Specifically, we formalize the notion of chosen-ciphertext security against chosen-distribution attacks (IND-CDA2) for N-PKE, which can be seen as the CCA-and-N-PKE version of the original IND-CDA security for PKE formalized in [2]¹. This security is guaranteed on condition that the seeds, the messages and the nonces have jointly sufficient min-entropy.

We separate our IND-CDA2 security notion and the security notion proposed in [8] for N-PKE (i.e., NBP1 and NBP2 security), by presenting two counterexamples. Our counterexamples actually show that even extending the original IND-CDA security (for H-PKE) to the nonce-based setting, IND-CDA security is still separated from NBP1/NBP2 security.

Since the original NBP1/NBP2 security and IND-CDA2 security do not imply each other, when we consider the security of N-PKE, we have to require that the N-PKE schemes achieve NBP1, NBP2 and IND-CDA2 security simultaneously. For simplicity, we call it HN-IND security.

In order to handle the potential problem of randomness failures, we recommend that one use HN-IND secure N-PKE if possible, and, especially, *employ a combination of a variety of things which do not repeat (e.g., the current time), and fresh, uniform and independent chosen randomness as nonce at every encryption* (and the seed can be reused). The reasons are as follows. If there are no randomness failures, the N-PKE schemes meet the universally accepted IND-CCA security. If some randomness failures present, the security which is as good as possible can be guaranteed. More specifically, if the randomness of the nonces is compromised, as long as the seed is uniformly chosen and confidential and the message-nonce pairs do not repeat, then NBP1 security guarantees that the schemes still achieve IND-CCA security. If the seed is exposed, but if the nonces are still unpredictable, then NBP2 security guarantees IND-CCA security. For the case that neither the seeds nor the nonces have sufficient min-entropy, as long as the seed-message-nonce tuples have sufficient min-entropy, and the messages are independent of the public key, then the N-PKE schemes achieve IND-CDA2 security, which is defined under chosen-ciphertext attacks and strictly stronger than IND-CDA security. We also note that for an extreme situation that both the seed and the nonces are arbitrarily determined by the adversaries, but the

¹ Very recently, Boldyreva, Patton and Shrimpton [10] formalized one CCA version of IND-CDA security for traditional PKE. There are some differences between their formalizations and ours. See Remark 2 for details.

messages still have sufficient min-entropy, then the schemes are actually D-PKE schemes achieving adaptive IND security (i.e., the adversary is allowed to access to the encryption oracle adaptively multiple times) in the CCA setting.

We note that the HN-IND secure N-PKE is able to adaptively handle the above cases, and achieves IND-CCA security even if there are some randomness failures. It's not necessary to decide which kind of PKE (i.e., traditional PKE, H-PKE, N-PKE or D-PKE) should be used according to the specific cases beforehand.

Besides, in the setting of D-PKE, there is another kind of adaptive security notion proposed by Raghunathan, Segev and Vadhan (RSV) in [20], where the messages are allowed to depend on the public key, but an upper bound on the number of the message distributions is required. For completeness, we also formalize a similar version of IND-CDA2 security *for N-PKE*, and call it the RSV version of HN-IND security.

HN-IND secure constructions. In this paper we provide an N-PKE scheme achieving HN-IND security in the random oracle model (ROM). Our approach is from the ROM construction of N-PKE in [8]. We notice that in [8], the nonce-based PKE schemes were constructed with a building block called *hedged extractor*. There are two constructions of hedged extractor proposed in [8], where the first one is in the ROM, and the second one is in the standard model. We emphasize that *under the security of hedged extractor*, both of the N-PKE schemes based on these two hedged extractors respectively are *not* HN-IND secure. The reason is that the security of hedged extractor is guaranteed only if either the seed or the nonce has enough min-entropy. Therefore, it seems that all the *generic constructions* of N-PKE based on hedged extractors do not achieve HN-IND security.

We also provide a generic construction of HN-IND secure N-PKE. The main idea of our scheme is from [16], which is a combination of an N-PKE scheme and a D-PKE scheme. Our conclusion shows that if the underlying N-PKE scheme is NBP1 and NBP2 secure, and the D-PKE scheme is adaptively IND secure in the CCA setting and unique-ciphertext secure, then the construction is HN-IND secure. If both the underlying constructions are built in the standard model, then our construction achieves HN-IND security in the standard model.

Moreover, we show that both of the constructions achieve the RSV version of HN-IND security.

Related work. Deterministic PKE was formally introduced by Bellare et al. [1] in CRYPTO 2007. A security notion called PRIV for D-PKE was defined, and some PRIV secure ROM constructions were proposed in [1]. Later, several equivalent security notions were formalized in [4], including the IND security used in this paper. Some variants of PRIV/IND security or D-PKE also appeared [5, 9, 11, 17, 20], and more D-PKE constructions were proposed [5, 6, 14]. Wicks [23] pointed out that the fully IND security of D-PKE in the standard model can not be achieved under any single-stage assumption. Later with the help of UCE [6], Bellare and Hoang [5] gave the first fully IND secure D-PKE scheme in the standard model. Selective opening security for D-PKE was also formalized

and achieved in the ROM [3, 16]. We note that the most commonly used security for D-PKE (i.e., PRIV or IND security) is a *non-adaptive* security notion. In other words, in the game defining the security, the adversary is allowed to make the challenge query only once.

Hedged PKE was introduced by Bellare et al. [2]. In [2], an *adaptive* security notion called IND-CDA, which is an extension of IND, is formalized, and a PKE scheme is called H-IND secure if it achieves IND-CPA and IND-CDA security simultaneously. Very recently, Boldyreva et al. [10] formalized the CCA version of IND-CDA security (which they named MMR-CCA security) for PKE with associated data. Both ROM constructions and standard-model constructions achieving fully H-IND security (i.e., the message-randomness pairs may be arbitrarily correlated) have been proposed [2, 5, 10]. The use of H-PKE in practice was explored in [10, 21].

Nonce-based PKE was introduced by Bellare and Tackmann in [8]. They formalized two security notions called NBP1 and NBP2, and showed ROM and standard-model constructions achieving both of the two security. Their constructions are based on a new primitive called hedged extractor. Nonce-based signatures was also defined and built in [8]. Recently, Hoang et al. [16] formalized SOA security for N-PKE, and lifted the security notion to H-PKE. To the best of our knowledge, it's the first security notion for hedged N-PKE. But their security is defined in the SOA setting, and more importantly, it is a *non-adaptive* security notion. Furthermore, we note that their security notion is a *comparison*-based security (see [4]), and our IND-CDA2 security is an *indistinguishability*-based one. Informally, denote by *COM-CDA2 security* the HN-SO-CCA security formalized in [16] with the restriction that I is *empty* (i.e., the adversaries do not perform corruptions. We refer the readers to [16] for the details). Exploring the relations among COM-CDA2 security and the *non-adaptive version* of our IND-CDA2 security is an interesting topic for future research.

2 Preliminaries

Notations and conventions. Vectors are written in boldface, e.g., \mathbf{x} . For a vector \mathbf{x} , let $|\mathbf{x}|$ denote its length and $\mathbf{x}[i]$ denote its i^{th} component for $i \in [|\mathbf{x}|]$. For a finite set X (resp. a string x), let $|X|$ (resp. $|x|$) denote its size (resp. length). We extend the set membership notations to vectors. For any game \mathbf{G} presented in this paper, denote by $\Pr[\mathbf{G}]$ the probability that the final output of \mathbf{G} is 1.

Public-key encryption. A (general) public-key encryption (PKE) scheme is a tuple of PPT algorithms $\text{PKE} = (\text{Kg}, \text{Enc}, \text{Dec})$. The key generation algorithm Kg , taking 1^k as input, generates a public/secret key pair (pk, sk) . The encryption algorithm Enc , taking pk and message $m \in \{0, 1\}^*$ as input, outputs a ciphertext c . The deterministic decryption algorithm Dec , taking sk and c as input, returns a value in $\{0, 1\}^* \cup \{\perp\}$. Standard correctness is required, which means that for any valid message $m \in \{0, 1\}^*$, $(pk, sk) \leftarrow$

Game $G_{PKE,A}^{ind-cca}(k)$ $(pk, sk) \leftarrow K_g(1^k)$ $b \leftarrow \{0, 1\}; C \leftarrow \emptyset$ $b' \leftarrow A^{ENC,DEC}(pk)$ Return $(b' = b)$	ENC(m_0, m_1) $c \leftarrow Enc(pk, m_b)$ $C \leftarrow C \cup c$ Return c	DEC(c) If $c \in C$, then Return \perp $m \leftarrow Dec(sk, c)$ Return m	
Game $G_{DE,A}^{de-ind}(k)$ $(pk, sk) \leftarrow DKg(1^k); b \leftarrow \{0, 1\}$ $\mathcal{M} \leftarrow A_1(1^k)$ $(m_0, m_1) \leftarrow \mathcal{M}; c \leftarrow DEnc(pk, m_b)$ $b' \leftarrow A_2(pk, c)$ Return $(b' = b)$	Game $G_{DE,A}^{de-cca}(k)$ $(pk, sk) \leftarrow DKg(1^k)$ $b \leftarrow \{0, 1\}; C \leftarrow \emptyset$ $St \leftarrow A_1^{ENC}(1^k)$ $b' \leftarrow A_2^{DEC}(pk, St)$ Return $(b' = b)$	ENC(\mathcal{M}) $(m_0, m_1) \leftarrow \mathcal{M}$ $c \leftarrow DEnc(pk, m_b)$ $C \leftarrow C \cup c$ Return c	DEC(c) If $c \in C$, then Return \perp $m \leftarrow DDec(sk, c)$ Return m

Fig. 1. Games for defining IND-CCA security of a standard PKE scheme PKE, IND security and adaptively CCA security of a D-PKE scheme DE.

$Kg(1^k)$ and $c \leftarrow Enc(pk, m)$, $Dec(sk, c) = m$ with overwhelming probability. For vectors \mathbf{m}, \mathbf{r} with $|\mathbf{m}| = |\mathbf{r}|$, we denote by $Enc(pk, \mathbf{m}; \mathbf{r}) := (Enc(pk, \mathbf{m}[1]; \mathbf{r}[1]), Enc(pk, \mathbf{m}[2]; \mathbf{r}[2]), \dots, Enc(pk, \mathbf{m}[|\mathbf{m}|]; \mathbf{r}[|\mathbf{m}|]))$.

IND-CCA security for PKE is defined by game $G_{PKE,A}^{ind-cca}$ in Fig. 1. For any (m_0, m_1) submitted to the encryption oracle $ENC(\cdot)$ in $G_{PKE,A}^{ind-cca}$, we require that $|m_0| = |m_1|$, and for every $i \in [|\mathbf{m}_0|]$, $|m_0[i]| = |m_1[i]|$. PKE is called *IND-CCA secure* if $Adv_{PKE,A}^{ind-cca}(k) = 2Pr[G_{PKE,A}^{ind-cca}(k)] - 1$ is negligible for any PPT adversary A , and called *IND-CPA secure* if A is not allowed to access to the decryption oracle $DEC(\cdot)$.

Following [1], the maximum public-key collision probability of PKE is defined by $\max_{pk} Pr[pk = \omega : (pk, sk) \leftarrow K_g(1^k)]$.

PKE secure under randomness failures. Currently, there are mainly three approaches to deal with the problems of randomness failures for PKE: deterministic PKE, hedged PKE, and nonce-based PKE. We recall their definitions and security notions as follows.

Deterministic PKE. A PKE scheme is called *deterministic* if the encryption algorithm is deterministic. This notion was formally introduced by Bellare et al. [1]. For a D-PKE scheme $DE = (DKg, DEnc, DDec)$, IND security [4] is defined by game $G_{DE,A}^{de-ind}$ in Fig. 1. An IND adversary $A = (A_1, A_2)$ in game $G_{DE,A}^{de-ind}$ is called *legitimate*, if for any (m_0, m_1) sampled by \mathcal{M} , associated with some polynomial $p(\cdot)$, the following two conditions hold: (i) $|m_0| = |m_1| = p(k)$, and for every $i \in [p(k)]$, $|m_0[i]| = |m_1[i]|$; (ii) for any $b \in \{0, 1\}$, $m_b[1], \dots, m_b[p(k)]$ are distinct. The guessing probability of A is denoted by $Guess_A(k)$, which returns the maximum of $Pr[m_b[i] = m]$ over all $b \in \{0, 1\}$, all $i \in [p(k)]$, all $m \in \{0, 1\}^*$, and all \mathcal{M} submitted by A_1 , where the probability is taken over $(m_0, m_1) \leftarrow \mathcal{M}(1^k)$. The block-source guessing probability of A is denoted by $Guess_A^{b-s}(k)$, which returns the maximum of $Pr[m_b[i] = m_i \mid m_b[j] = m_j, \forall j \in [i - 1]]$ over all $b \in \{0, 1\}$, all $i \in [p(k)]$, all $m_1, \dots, m_i \in \{0, 1\}^*$, and all \mathcal{M} submitted by A_1 , where the probability is taken over $(m_0, m_1) \leftarrow \mathcal{M}(1^k)$. We say that A has *high min-entropy* (resp. *high block-source min-entropy* [9]) if $Guess_A(k)$ (resp. $Guess_A^{b-s}(k)$) is negligible. Scheme DE is *fully IND secure* (resp. *block-source IND*

secure) if $\text{Adv}_{\text{DE},A}^{\text{de-ind}}(k) = 2\Pr[\mathbf{G}_{\text{DE},A}^{\text{de-ind}}(k)] - 1$ is negligible for any legitimate PPT adversary A of high min-entropy (resp. high block-source min-entropy).

We say that a PPT adversary is *adaptive* if it is allowed to query the challenge oracle multiple times, and each query may depend on the replies to the previous queries. IND is a non-adaptive security notion. A stronger adaptive security notion for D-PKE, *adaptively CCA security*, is defined by game $\mathbf{G}_{\text{DE},A}^{\text{de-cca}}$ in Fig. 1. We similarly define adaptively CCA adversary that is *legitimate* and *has high min-entropy*. Scheme DE is *fully adaptively CCA secure* if $\text{Adv}_{\text{DE},A}^{\text{de-cca}}(k) = 2\Pr[\mathbf{G}_{\text{DE},A}^{\text{de-cca}}(k)] - 1$ is negligible for any legitimate PPT adversary A of high min-entropy. Block-source adaptively CCA security for D-PKE is similarly defined.

DE is called *unique-ciphertext* [5], if for any k , any (pk, sk) generated by DKg , and any message $m \in \{0,1\}^*$, there is at most one $c \in \{0,1\}^*$ such that $\text{DDec}(sk, c) = m$. Each D-PKE scheme can be efficiently transformed to a unique-ciphertext one [5].

Hedged PKE. In ASIACRYPT 2009, Bellare, et al. [2] introduced the notion of IND-CDA security, which formalized the security for PKE when the messages and the randomness jointly have high entropy. A PKE scheme is called *hedged* if it achieves both IND-CPA security and IND-CDA security, which means that it achieves IND-CPA security when the random coins employed during the encryption are truly random, and achieves IND-CDA security when bad random coins are employed but the messages and the random coins jointly have high min-entropy.

For a hedged PKE (H-PKE) scheme $\text{HE} = (\text{HKg}, \text{HEnc}, \text{HDec})$, IND-CDA security is defined by game $\mathbf{G}_{\text{HE},A}^{\text{ind-cda}}$ in Fig. 2. An IND-CDA adversary $A = (A_1, A_2)$ in game $\mathbf{G}_{\text{HE},A}^{\text{ind-cda}}$ is called *legitimate*, if for any $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ sampled by \mathcal{M} , associated with some polynomial $\mathbf{p}(\cdot)$, which is the message sampler submitted to oracle $\text{LR}(\cdot)$ by A_1 , the following two conditions hold: (i) $|\mathbf{m}_0| = |\mathbf{m}_1| = |\mathbf{r}| = \mathbf{p}(k)$, and for every $i \in [\mathbf{p}(k)]$, $|\mathbf{m}_0[i]| = |\mathbf{m}_1[i]|$; (ii) for any $b \in \{0,1\}$, $(\mathbf{m}_b[1], \mathbf{r}[1]), \dots, (\mathbf{m}_b[\mathbf{p}(k)], \mathbf{r}[\mathbf{p}(k)])$ are distinct. The guessing probability of A is denoted by $\text{Guess}_A(k)$, which returns the maximum of $\Pr[(\mathbf{m}_b[i], \mathbf{r}[i]) = (m, r)]$ over all $b \in \{0,1\}$, all $i \in [\mathbf{p}(k)]$, all $m \in \{0,1\}^*$, all $r \in \{0,1\}^*$, and all \mathcal{M} submitted by A_1 , where the probability is taken over $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow \mathcal{M}(1^k)$. We say that A has *high min-entropy* if $\text{Guess}_A(k)$ is negligible. Scheme HE is *IND-CDA secure* if $\text{Adv}_{\text{HE},A}^{\text{ind-cda}}(k) = 2\Pr[\mathbf{G}_{\text{HE},A}^{\text{ind-cda}}(k)] - 1$ is negligible for any legitimate PPT adversary A of high min-entropy. The notion of *block-source* IND-CDA security is similarly defined [2].

Nonce-based PKE. A nonce-based public-key encryption (N-PKE) scheme with nonce space NE.NS is a tuple of PPT algorithms $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$. The key generation algorithm NKg , taking 1^k as input, generates a public/secret key pair (pk, sk) . The seed generation algorithm NSKg taking 1^k returns a sender seed xk . Let NE.SD denote the seed space. We say that NSKg is *trivial*, if it returns a uniformly chosen xk from $\text{NE.SD} = \{0,1\}^k$. The *deterministic* encryption algorithm NEnc , taking pk, xk , message $m \in \{0,1\}^*$, and nonce $n \in \text{NE.NS}$ as input, outputs a ciphertext c . The deterministic

Game $\mathbf{G}_{\text{NE,NG},A}^{\text{nbp1}}(k)$	Game $\mathbf{G}_{\text{NE,NG},A}^{\text{nbp2}}(k)$	Game $\mathbf{G}_{\text{HE},A}^{\text{ind-cda}}(k)$
$(pk, sk) \leftarrow \text{NKG}(1^k)$ $xk \leftarrow \text{NSKG}(1^k)$ $b \leftarrow \{0, 1\}; St \leftarrow \epsilon$ $C, Q_0, Q_1 \leftarrow \emptyset$ $b' \leftarrow A^{\text{ENC,DEC}}(pk)$ Return $(b' = b)$	$(pk, sk) \leftarrow \text{NKG}(1^k)$ $xk \leftarrow \text{NSKG}(1^k)$ $b \leftarrow \{0, 1\}; St \leftarrow \epsilon$ $C \leftarrow \emptyset$ $b' \leftarrow A^{\text{ENC,DEC}}(pk, xk)$ Return $(b' = b)$	$(pk, sk) \leftarrow \text{HKg}(1^k)$ $b \leftarrow \{0, 1\}$ $St \leftarrow A_1^{\text{LR}}(1^k)$ $b' \leftarrow A_2(pk, St)$ Return $(b' = b)$
$\text{ENC}(m_0, m_1, \eta)$ If $(m_0 \neq m_1)$, then return \perp If $((m_0, n) \in Q_0)$ or $((m_1, n) \in Q_1)$, then return \perp $(n, St) \leftarrow \text{NG}(1^k, \eta, St); c \leftarrow \text{NEnc}(pk, xk, m_b, n)$ $Q_0 \leftarrow Q_0 \cup \{(m_0, n)\}; Q_1 \leftarrow Q_1 \cup \{(m_1, n)\}$ $C \leftarrow C \cup \{c\}$ Return c	$\text{ENC}(m_0, m_1, \eta)$ If $(m_0 \neq m_1)$, then return \perp Return \perp $(n, St) \leftarrow \text{NG}(1^k, \eta, St)$ $c \leftarrow \text{NEnc}(pk, xk, m_b, n)$ $C \leftarrow C \cup \{c\}$ Return c	$\text{LR}(\mathcal{M})$ $(m_0, m_1, r) \leftarrow \mathcal{M}(1^k)$ $c \leftarrow \text{HEnc}(pk, m_b; r)$ Return c
$\text{DEC}(c)$ If $c \in C$, then return \perp $m \leftarrow \text{NDec}(sk, c)$ Return m	$\text{DEC}(c)$ If $c \in C$, then return \perp $m \leftarrow \text{NDec}(sk, c)$ Return m	

Fig. 2. Games for defining NBP1, NBP2 security of a N-PKE scheme NE, and IND-CDA security for a H-PKE scheme HE.

decryption algorithm NDec is the same as that of the traditional PKE schemes, on input sk and c , returns a value in $\{0, 1\}^* \cup \{\perp\}$. The nonce is not necessary for decryption. Standard correctness is required, which means that for any valid message $m \in \{0, 1\}^*$, $(pk, sk) \leftarrow \text{NKG}(1^k)$, $xk \leftarrow \text{NSKG}(1^k)$, $n \in \text{NE.NS}$ and $c \leftarrow \text{NEnc}(pk, xk, m, n)$, $\text{Dec}(sk, c) = m$ with overwhelming probability.

The notion of N-PKE was introduced by Bellare and Tackmann [8]. In their N-PKE constructions, the nonces are generated by a building block called *nonce generator* NG with nonce space NE.NS . A nonce generator NG is a PPT algorithm taking 1^k , a current state St , and a *nonce selector* η as input, returns a nonce $n \in \text{NE.NS}$ and a new state St , i.e., $(n, St) \leftarrow \text{NG}(1^k, \eta, St)$. Standard security of NG requires that the generated nonces should be unpredictable and never repeat. We refer the readers to [8, 16] for the formal definition.

Two kinds of security notions for N-PKE were introduced in [8], which we recall in Fig. 2. An N-PKE scheme NE , with respect to NG , is NBP1 (resp. NBP2) secure if $\text{Adv}_{\text{NE,NG},A}^{\text{nbp1}}(k) = 2\text{Pr}[\mathbf{G}_{\text{NE,NG},A}^{\text{nbp1}}(k)] - 1$ (resp. $\text{Adv}_{\text{NE,NG},A}^{\text{nbp2}}(k) = 2\text{Pr}[\mathbf{G}_{\text{NE,NG},A}^{\text{nbp2}}(k)] - 1$) is negligible for any PPT adversary A , where game $\mathbf{G}_{\text{NE,NG},A}^{\text{nbp1}}$ (resp. $\mathbf{G}_{\text{NE,NG},A}^{\text{nbp2}}$) is defined in Fig. 2. According to [8], NBP1 security is achieved for any nonce generator (even for predictable nonce generator), as long as the message-nonce pairs do not repeat; NBP2 security is achieved for any unpredictable nonce generators.

3 Hedged Security for Nonce-Based Public-Key Encryption

In this section, we introduce hedged security for nonce-based public-key encryption. We first formalize chosen-ciphertext security against chosen-distribution attacks (IND-CDA2 security) for N-PKE. Then, we explore the relations among

the security notions of N-PKE. Lastly, we formalize a special version (the Raghunathan et al. [20] version) of IND-CDA2 security for N-PKE.

3.1 Chosen-Ciphertext Security Against Chosen-Distribution Attacks

Notice that the original message samplers were defined for the general PKE schemes, which do not sample the seeds and the nonces. Therefore, we firstly formalize the notion of message samplers for N-PKE as follows.

Definition 1 (Message sampler for N-PKE). A message sampler \mathcal{M} for N-PKE is a PPT algorithm taking 1^k as input, and returning $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$.

For any N-PKE scheme $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$ w.r.t. nonce generator NG , consider game $\mathbf{G}_{\text{NE}, A}^{\text{ind-cda2}}$ as shown in Fig. 3.

We say that the adversary $A = (A_1, A_2)$ in game $\mathbf{G}_{\text{NE}, A}^{\text{ind-cda2}}$ is *legitimate*, if for any $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n})$ sampled by \mathcal{M} which is associated with some polynomial $p(\cdot)$, the following two conditions hold: (i) $|\mathbf{m}_0| = |\mathbf{m}_1| = |\mathbf{xk}| = |\mathbf{n}| = p(k)$, and for every $i \in [p(k)]$, $|\mathbf{m}_0[i]| = |\mathbf{m}_1[i]|$; (ii) for any $b \in \{0, 1\}$, $(\mathbf{xk}[1], \mathbf{m}_b[1], \mathbf{n}[1]), \dots, (\mathbf{xk}[p(k)], \mathbf{m}_b[p(k)], \mathbf{n}[p(k)])$ are distinct.

Similarly, the guessing probability of A is denoted by $\text{Guess}_A(k)$, which returns the maximum of $\Pr[(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk, m, n)]$ over all $b \in \{0, 1\}$, all $i \in [p(k)]$, all $xk \in \{0, 1\}^*$, all $m \in \{0, 1\}^*$, all $n \in \{0, 1\}^*$, and all \mathcal{M} submitted by A_1 , where the probability is taken over $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$. The block-source guessing probability of A is denoted by $\text{Guess}_A^{\text{b-s}}(k)$, which returns the maximum of $\Pr[(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk, m, n) \mid (\mathbf{xk}[j], \mathbf{m}_b[j], \mathbf{n}[j]) = (xk_j, m_j, n_j), \forall j \in [i-1]]$ over all $b \in \{0, 1\}$, all $i \in [p(k)]$, all $xk_1, \dots, xk_i \in \{0, 1\}^*$, all $m_1, \dots, m_i \in \{0, 1\}^*$, all $n_1, \dots, n_i \in \{0, 1\}^*$, and all \mathcal{M} submitted by A_1 , where the probability is taken over $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$. We say that the IND-CDA2 adversary A has *high min-entropy* (resp. *high block-source min-entropy*) if $\text{Guess}_A(k)$ (resp. $\text{Guess}_A^{\text{b-s}}(k)$) is negligible.

Game $\mathbf{G}_{\text{NE}, A}^{\text{ind-cda2}}(k)$	$\text{LR}(\mathcal{M})$	$\text{DEC}(c)$
$(pk, sk) \leftarrow \text{NKg}(1^k)$	$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$	If $c \in C$, then return \perp
$b \leftarrow \{0, 1\}; C \leftarrow \emptyset$	$c \leftarrow \text{NEnc}(pk, \mathbf{xk}, \mathbf{m}_b, \mathbf{n})$	$m \leftarrow \text{NDec}(sk, c)$
$St \leftarrow A_1^{\text{LR}}(1^k)$	$C \leftarrow C \cup c$	Return m
$b' \leftarrow A_2^{\text{DEC}}(pk, St)$	Return c	
Return $(b' = b)$		

Fig. 3. Game for defining IND-CDA2 security of a N-PKE scheme NE

Definition 2 (IND-CDA2). An N-PKE scheme $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$, with respect to nonce generator NG , is *IND-CDA2 secure* (resp. *block-source IND-CDA2 secure*), if for any legitimate PPT adversary $A = (A_1, A_2)$ having *high min-entropy* (resp. *high block-source min-entropy*), its advantage $\text{Adv}_{\text{NE}, A}^{\text{ind-cda2}}(k) = 2\Pr[\mathbf{G}_{\text{NE}, A}^{\text{ind-cda2}}(k)] - 1$ is negligible, where game $\mathbf{G}_{\text{NE}, A}^{\text{ind-cda2}}$ is defined in Fig. 3.

Remark 1. Note that if the adversary A is not allowed to access to the decryption oracle $\text{DEC}(\cdot)$, then we call the defining security notion “IND-CDA security in the nonce-based setting”. Note that in [2] the notion of “IND-CDA security” was defined for the general PKE schemes, not for N-PKE. For simplicity, in this paper we abuse the notation, still using “IND-CDA security” when we refer to “IND-CDA security in the nonce-based setting”.

Remark 2. Recently, Boldyreva et al. [10] formalized the CCA version of IND-CDA security for PKE, and called it MMR-CCA security. The notion of MMR-CCA security is defined for PKE with associated data, and in the experiment defining MMR-CCA security, the adversary is allowed to access to the decryption oracle *before* seeing the public key. Our IND-CDA2 security is formalized for N-PKE (without associated data), and the adversary is not allowed to access to the decryption oracle until it receives the public key. If the lengths of the seed and the nonce are both restricted to be 0, our security will naturally become adaptive CCA security for D-PKE.

3.2 Separations Between NBP1/NBP2 Security and IND-CDA2 Security

We now show that NBP1/NBP2 security and IND-CDA2 security do not imply each other. Our separation results are based on the following observations. In the game defining IND-CDA2 security, (i) the sender seed xk is specified by the adversary through the generated message sampler \mathcal{M} , instead of being generated by NSKg in the game defining NBP1/NBP2 security; (ii) the challenge messages are independent of the public key, instead of being chosen by the adversary after seeing the public key in the game defining NBP1/NBP2 security.

NBP1/NBP2 $\not\Rightarrow$ IND-CDA2. Actually, we provide a stronger conclusion here “NBP1/NBP2 $\not\Rightarrow$ IND-CDA”. For an NBP1/NBP2 secure N-PKE scheme $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$ w.r.t. a nonce generator NG , where NSKg is trivial, we construct a new N-PKE scheme $\text{NE}' = (\text{NKg}', \text{NSKg}', \text{NEnc}', \text{NDec}')$, w.r.t. the same NG , as shown in Fig. 4.

Since NSKg is trivial, we have that $xk \leftarrow \{0, 1\}^k$. As a result, the probability that $xk = 0^k$ is negligible. Therefore, NBP1/NBP2 security of NE' is guaranteed by NBP1/NBP2 security of NE .

Now we show an adversary $A = (A_1, A_2)$ attacking NE' in the sense of IND-CDA. For simplicity, we assume that the message space is $\{0, 1\}^k$. A_1 makes an $\text{LR}(\cdot)$ query by submitting a message sampler \mathcal{M} (with $\text{p}(k) = 1$), which is defined as follows:

1. Set $xk = 0^k$.
2. For any $b \in \{0, 1\}$, choose m_b uniformly random from $\{0, 1\}^k$, conditioned on that the last bit of m_b is b .
3. Choose n uniformly random from nonce space NE.NS .

Note that n is uniformly chosen from NE.NS , and m_0, m_1 are both uniformly chosen from $\{0, 1\}^{k-1}$. So adversary A is legitimate and has high min-entropy.

$\text{NKg}'(1^k)$ $(pk, sk) \leftarrow \text{NKg}(1^k)$ Return (pk, sk)	$\text{NSKg}'(1^k)$ $xk \leftarrow \text{NSKg}(1^k)$ Return xk	$\text{NEnc}'(pk, xk, m, n)$ If $xk = 0^k$, then $c \leftarrow m, c' \leftarrow (c 0)$ Else, $c \leftarrow \text{NEnc}(pk, xk, m, n)$ $c' \leftarrow (c 1)$ Return c'	$\text{NDec}'(sk, c')$ Parse $c' = (c b)$ If $b = 0$, then $m \leftarrow c$ Else, $m \leftarrow \text{NDec}(sk, c)$ Return m
$\text{NKg}''(1^k)$ $(pk, sk) \leftarrow \text{NKg}(1^k)$ Return (pk, sk)	$\text{NSKg}''(1^k)$ $xk \leftarrow \text{NSKg}(1^k)$ Return xk	$\text{NEnc}''(pk, xk, m, n)$ If $m = pk$, then $c \leftarrow m, c'' \leftarrow (c 0)$ Else, $c \leftarrow \text{NEnc}(pk, xk, m, n)$ $c'' \leftarrow (c 1)$ Return c''	$\text{NDec}''(sk, c'')$ Parse $c'' = (c b)$ If $b = 0$, then $m \leftarrow c$ Else, $m \leftarrow \text{NDec}(sk, c)$ Return m

Fig. 4. Counterexamples $\text{NE}' = (\text{NKg}', \text{NSKg}', \text{NEnc}', \text{NDec}')$ and $\text{NE}'' = (\text{NKg}'', \text{NSKg}'', \text{NEnc}'', \text{NDec}'')$.

After receiving the ciphertext $c' = (c||0)$ from $\text{LR}(\cdot)$, A returns the last bit of c as its final output. The advantage of A is obviously 1.

IND-CDA2 $\not\Rightarrow$ NBP1/NBP2. Assuming that there is an N-PKE scheme $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$, w.r.t. a nonce generator NG , achieving IND-CDA2 security and having negligible maximum public-key collision probability maxpk_{NE} . Note that the requirement that maxpk_{NE} is negligible is very mild, since any IND-CPA secure PKE has negligible maxpk_{NE} [1]. Based on NE , we present a new N-PKE scheme $\text{NE}'' = (\text{NKg}'', \text{NSKg}'', \text{NEnc}'', \text{NDec}'')$, w.r.t. the same NG , as shown in Fig. 4.

For any IND-CDA2 adversary $A = (A_1, A_2)$, A does not receive pk until it finishes the process of $\text{LR}(\cdot)$ query. The negligible maxpk_{NE} guarantees that

$$\max_{i \in [|m_0|]} \Pr[(\mathbf{m}_0[i] = pk) \vee (\mathbf{m}_1[i] = pk) : \mathcal{M} \text{ is generated by } A_1^{\text{LR}}, \\ (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)]$$

is negligible, where the probability is taken over A_1^{LR} and $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$. Therefore, NE'' is IND-CDA2 secure.

Note that in the game defining NBP1/NBP2 security, the adversary generates the challenge messages (m_0, m_1) after seeing the public key. So we construct a NBP1/NBP2 adversary A as follows. Upon receiving pk , A sets $m_0 = pk$, and chooses an arbitrary distinct m_1 from the message space such that $|m_1| = |m_0|$, and an arbitrary valid nonce selector η . Then A submits the generated (m_0, m_1, η) to the encryption oracle $\text{ENC}(\cdot)$. After receiving the ciphertext $c'' = (c||b)$, A returns b as its final output. The advantage of A is obviously 1.

Formally, we have the following theorem.

Theorem 1. *NBP1/NBP2 security and IND-CDA2 security do not imply each other.*

Remark 3. The aforementioned NBP1/NBP2 adversary attacking NE'' does not make any decryption query. So we actually proved that IND-CDA2 security

does not imply the CPA version of NBP1/NBP2 security. Therefore, our results also show the separations between NBP1/NBP2 security and IND-CDA security.

3.3 The RSV Version of IND-CDA2 Security

In EUROCRYPT 2013, Raghunathan et al. [20] formalized another security notion for D-PKE, ACD-CPA/CCA security, which allows the adversaries to adaptively choose message distributions *after* seeing the public key, with the following two restrictions: (1) the adversaries have high min-entropy; (2) for each adversary, there is an upper bound on the number of the message distributions from which the adversary is allowed to adaptively choose. The upper bound is $2^{p(k)}$ where $p(\cdot)$ is any a-priori fixed polynomial. Raghunathan et al. [20] proposed a D-PKE scheme achieving ACD-CCA security in the standard model, based on a primitive called \mathcal{R} -lossy trapdoor function.

Considering that this is an important optional security notion for D-PKE, and as far as we know, ACD-CCA is neither weaker nor stronger than adaptive CCA security, we formalize a similar version of IND-CDA2 security here, which we call the RSV version of IND-CDA2 security (RIND-CDA2).

Definition 3 (RSV message sampler for N-PKE). An RSV message sampler \mathcal{M} for N-PKE is a PPT algorithm taking 1^k as input, and returning $(\mathbf{m}, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$.

Definition 4 (Uniform message sampler with respect to \mathcal{M}). For an RSV message sampler \mathcal{M} for N-PKE, a PPT algorithm \mathcal{U} is a uniform message sampler with respect to \mathcal{M} if for any message vector sampled by \mathcal{M} (i.e., $(\mathbf{m}, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$), $\mathbf{m}_u \leftarrow \mathcal{U}(\mathcal{M}, \mathbf{m})$ is uniformly distributed over the same message space specified by \mathcal{M} , such that $|\mathbf{m}_u| = |\mathbf{m}|$ and $|\mathbf{m}_u[i]| = |\mathbf{m}[i]|$ for any $i \in [|\mathbf{m}|]$.

For any N-PKE scheme $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$ w.r.t. nonce generator NG , consider game $\mathbf{G}_{\text{NE},A}^{\text{rind-cda2}}$ as shown in Fig. 5.

The adversary A in game $\mathbf{G}_{\text{NE},A}^{\text{rind-cda2}}$ is *legitimate*, if for any $(\mathbf{m}, \mathbf{xk}, \mathbf{n})$ sampled by \mathcal{M} which is associated with some polynomial $p(\cdot)$, the following two conditions hold: (i) $|\mathbf{m}| = |\mathbf{xk}| = |\mathbf{n}| = p(k)$; (ii) $(\mathbf{m}[1], \mathbf{xk}[1], \mathbf{n}[1]), \dots, (\mathbf{m}[p(k)], \mathbf{xk}[p(k)], \mathbf{n}[p(k)])$ are distinct.

Similar to that of Sect. 3.1, we have the guessing probabilities $\text{Guess}_A(k)$ and $\text{Guess}_A^{\text{b-s}}(k)$. We say that the RIND-CDA2 adversary A has *high min-entropy* (resp. *high block-source min-entropy*) if $\text{Guess}_A(k)$ (resp. $\text{Guess}_A^{\text{b-s}}(k)$) is negligible.

For any given polynomial $p(\cdot)$, we have the following definition.

Definition 5 ($2^{p(k)}$ -bounded adversary). For any PPT legitimate adversary A having high min-entropy (resp. high block-source min-entropy) in game $\mathbf{G}_{\text{NE},A}^{\text{rind-cda2}}$, let \mathcal{S}_{mg} be the set of message samplers which A may submit to the RoR oracle as a query with non-zero probability. A is a $2^{p(k)}$ -bounded (resp. $2^{p(k)}$ -bounded block-source) adversary if for every $k \in \mathbb{N}$, $|\mathcal{S}_{mg}| \leq 2^{p(k)}$.

Game $G_{NE,A}^{\text{rind-cda2}}(k)$	RoR(\mathcal{M})	DEC(c)
$(pk, sk) \leftarrow \text{NKg}(1^k)$	$(\mathbf{m}, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$	If $c \in C$, then return \perp
$b \leftarrow \{0, 1\}; C \leftarrow \emptyset$	$\mathbf{m}_1 \leftarrow \mathbf{m}; \mathbf{m}_0 \leftarrow \mathcal{U}(\mathcal{M}, \mathbf{m})$	$m \leftarrow \text{NDec}(sk, c)$
$b' \leftarrow A^{\text{RoR,DEC}}(pk)$	$\mathbf{c} \leftarrow \text{NEnc}(pk, \mathbf{xk}, \mathbf{m}_b, \mathbf{n})$	Return m
Return $(b' = b)$	$C \leftarrow C \cup \mathbf{c}$	
	Return \mathbf{c}	

Fig. 5. Game for defining RIND-CDA2 security of a N-PKE scheme NE, where \mathcal{U} is defined in Definition 4.

Definition 6 (RIND-CDA2). An N-PKE scheme NE, w.r.t. nonce generator NG, is RIND-CDA2 secure (resp. block-source RIND-CDA2 secure), if for any $2^{p(k)}$ -bounded (resp. $2^{p(k)}$ -bounded block-source) adversary A, its advantage $\text{Adv}_{NE,A}^{\text{rind-cda2}}(k) = 2\text{Pr}[G_{NE,A}^{\text{rind-cda2}}(k)] - 1$ is negligible, where game $G_{NE,A}^{\text{rind-cda2}}$ is defined in Fig. 5.

4 Construction of H-PKE in the Random Oracle Model

In EUROCRYPT 2016, Bellare and Tackmann [8] proposed an NBP1/ NBP2 secure N-PKE scheme in the random oracle model. Their construction is based on a building block which they introduced and called *hedged extractor*.

In this section, we show that the Bellare-Tackmann ROM construction actually achieves HN-IND security. But we note that this construction cannot be generalized to the schemes based on hedged extractors like [8, Fig. 6].

Firstly, we recall the N-PKE scheme RtP [8], w.r.t. a nonce generator NG, as follows. Let PKE = (Kg, Enc, Dec) be a traditional probabilistic PKE scheme with message space MSP and randomness space \mathcal{R}_{Enc} , and $\text{RO} : \{0, 1\}^* \rightarrow \mathcal{R}_{\text{Enc}}$ be a random oracle. The N-PKE scheme RtP is presented in Fig. 6.

Now we turn to the security. It has been proved in [8] that RtP is NBP1/NBP2 secure. So what remains is to prove its IND-CDA2 security. Formally, we have the following theorem.

Theorem 2. If PKE is a traditional IND-CCA secure PKE scheme, then N-PKE scheme RtP, w.r.t. a nonce generator NG, is IND-CDA2 secure in the random oracle model.

RKg(1^k)	RSKg(1^k)	REnc(pk, xk, m, n)	RDec(sk, c)
$(pk, sk) \leftarrow \text{Kg}(1^k)$	$xk \leftarrow \text{NE.SD}$	$r \leftarrow \text{RO}(xk, m, n)$	$m \leftarrow \text{Dec}(sk, c)$
Return (pk, sk)	Return xk	$c \leftarrow \text{Enc}(pk, m; r)$	Return m
		Return c	

Fig. 6. N-PKE scheme RtP = (RKg, RSKg, REnc, RDec).

Proof. For any legitimate PPT IND-CDA2 adversary A having high min-entropy, let $q_r(k)$ (resp. $q_l(k)$) denote the number of random-oracle queries (resp. LR queries) of A.

Consider a sequence of games $\mathbf{G}_0 - \mathbf{G}_6$ in Figs. 7 and 8. In each game, there is a random oracle RO which maintains a local array H as shown in Fig. 7. Denote by RO_A the random-oracle interface of A . Note that in games \mathbf{G}_4 and \mathbf{G}_5 , the oracle answers of RO_A and the answers given to the LR oracle in reply to its RO queries are independent, so we introduce another local array H_A for RO_A . In game \mathbf{G}_6 , the LR oracle does not access to RO, so we omit the procedure “On query RO” in Fig. 8. For convenience, the RO queries made by A through RO_A is called RO_A queries in this proof. Without loss of generality, we assume that in each game, A does not repeat any RO_A queries.

Now we explain the sequence of games.

Game \mathbf{G}_0 implements game $\mathbf{G}_{\text{RtP},A}^{\text{ind-cda}2}$. So we have

$$\text{Adv}_{\text{RtP},A}^{\text{ind-cda}2}(k) = 2\Pr[\mathbf{G}_0(k)] - 1. \tag{1}$$

In game \mathbf{G}_1 , we introduces two sets T_1 and T_2 . T_1 denotes the set of RO queries made by A (i.e., RO_A queries), and T_2 denotes the set of RO queries made by the LR oracle. The changes made in \mathbf{G}_1 does not affect the final output. Therefore,

$$\Pr[\mathbf{G}_1(k)] = \Pr[\mathbf{G}_0(k)]. \tag{2}$$

Games \mathbf{G}_2 and \mathbf{G}_1 are identical-until- bad_1 . Denote by $\Pr[\text{bad}_1]$ the probability that \mathbf{G}_2 sets bad_1 . According to the fundamental lemma of game-playing [7], we have that $|\Pr[\mathbf{G}_2(k)] - \Pr[\mathbf{G}_1(k)]| \leq \Pr[\text{bad}_1]$.

Let \mathcal{M}' , associated with some polynomial $\mathfrak{p}(\cdot)$, denote the message sampler leading to bad_1 . Game \mathbf{G}_2 sets bad_1 only if A has made some RO_A query (xk', m', n') beforehand, such that for the \mathcal{M}' and $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}'$, there are some $b \in \{0, 1\}$ and some $i \in [|\mathbf{n}|]$ satisfying $(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk', m', n')$. Since A has high min-entropy, for any RO_A query (xk', m', n') , we have that for any \mathcal{M}' , any $b \in \{0, 1\}$, and any $i \in [|\mathfrak{p}(k)|]$,

$$\Pr[(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk', m', n') : (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}'] \leq \text{Guess}_A(k).$$

In other words, for any RO_A query (xk', m', n') ,

$$\begin{aligned} & \max_{\mathcal{M}', b, i} \Pr[(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk', m', n') : (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}'] \\ & \leq \text{Guess}_A(k). \end{aligned}$$

Notice that A makes totally $q_r(k)$ random-oracle queries and $q_l(k)$ LR queries. So we have $\Pr[\text{bad}_1] \leq 2q_r(k)q_l(k)\mathfrak{p}(k)\text{Guess}_A(k)$. Therefore,

$$|\Pr[\mathbf{G}_2(k)] - \Pr[\mathbf{G}_1(k)]| \leq \Pr[\text{bad}_1] \leq 2q_r(k)q_l(k)\mathfrak{p}(k)\text{Guess}_A(k). \tag{3}$$

Games \mathbf{G}_3 and \mathbf{G}_2 are identical-until- bad_2 . \mathbf{G}_3 sets bad_2 only if the current RO_A query (xk', m', n') has been queried by the LR oracle previously. In game \mathbf{G}_3 , if bad_2 is set, then the $H[xk', m', n']$ is overwritten with a random element from \mathcal{R}_{Enc} . Denote by $\Pr[\text{bad}_2]$ the probability that \mathbf{G}_3 sets bad_2 . Then, we have that $|\Pr[\mathbf{G}_3(k)] - \Pr[\mathbf{G}_2(k)]| \leq \Pr[\text{bad}_2]$. In order to bound $\Pr[\text{bad}_2]$, we present the following lemma and postpone its proof.

<p>Games $\mathbf{G}_0, \boxed{\mathbf{G}_1 - \mathbf{G}_3}, \boxed{\mathbf{G}_2 - \mathbf{G}_3}, \mathbf{G}_3$</p> <p>$(pk, sk) \leftarrow \text{Kg}(1^k); b \leftarrow \{0, 1\}; C \leftarrow \emptyset; \boxed{T_1, T_2 \leftarrow \emptyset}$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k); b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return $(b' = b)$</p> <p>On query $\text{RO}_A(xk', m', n')$: $\boxed{T_1 \leftarrow T_1 \cup \{(xk', m', n')\}}$ If $(xk', m', n') \in T_2$, then $\text{bad}_2 \leftarrow \text{true}; H[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $\text{RO}(xk', m', n')$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ For $i \in [\mathbf{n}]$, then $\boxed{T_2 \leftarrow T_2 \cup \{(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i])\}}$ If $(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) \in T_1$, then $\text{bad}_1 \leftarrow \text{true}; H[\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]] \leftarrow \mathcal{R}_{\text{Enc}}$ $\mathbf{r}[i] \leftarrow \text{RO}(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i])$ $\mathbf{c}[i] \leftarrow \text{Enc}(pk, \mathbf{m}_b[i]; \mathbf{r}[i])$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: If $c' \in C$, then return \perp $m' \leftarrow \text{Dec}(sk, c')$ Return m'</p> <p>On query $\text{RO}(xk', m', n')$: If $H[xk', m', n'] = \perp$, then $H[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H[xk', m', n']$</p>	<p>Games $\mathbf{G}_4, \boxed{\mathbf{G}_5}$</p> <p>$(pk, sk) \leftarrow \text{Kg}(1^k); b \leftarrow \{0, 1\}; C \leftarrow \emptyset; T_1, T_2 \leftarrow \emptyset$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k); b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return $(b' = b)$</p> <p>On query $\text{RO}_A(xk', m', n')$: $T_1 \leftarrow T_1 \cup \{(xk', m', n')\}$ If $H_A[xk', m', n'] = \perp$, then $H_A[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H_A[xk', m', n']$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ For $i \in [\mathbf{n}]$, then $T_2 \leftarrow T_2 \cup \{(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i])\}$ $\mathbf{r}[i] \leftarrow \text{RO}(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i])$ $\mathbf{c}[i] \leftarrow \text{Enc}(pk, \mathbf{m}_b[i]; \mathbf{r}[i])$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: If $c' \in C$, then return \perp $m' \leftarrow \text{Dec}(sk, c')$ Return m'</p> <p>On query $\text{RO}(xk', m', n')$: If $H[xk', m', n'] = \perp$, then $H[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ If $H[xk', m', n'] \neq \perp$, then $\text{bad}_3 \leftarrow \text{true}; H[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H[xk', m', n']$</p>
--	---

Fig. 7. Games $\mathbf{G}_0 - \mathbf{G}_5$ in the proof of Theorem 2. Boxed code is only executed in the games specified by the game names in the same box style.

Lemma 1. *There is an IND-CCA adversary B_{upr} attacking PKE with advantage $\text{Adv}_{\text{PKE}, B_{\text{upr}}}^{\text{ind-cca}}(k)$, such that*

$$\Pr[\text{bad}_2] \leq 2\text{Adv}_{\text{PKE}, B_{\text{upr}}}^{\text{ind-cca}}(k) + (q_r(k) + \frac{q_l(k)\mathbf{p}(k) - 1}{2})q_l(k)\mathbf{p}(k)\text{Guess}_A(k).$$

It follows that

$$\begin{aligned} & |\Pr[\mathbf{G}_3(k)] - \Pr[\mathbf{G}_2(k)]| \\ & \leq 2\text{Adv}_{\text{PKE}, B_{\text{upr}}}^{\text{ind-cca}}(k) + (q_r(k) + \frac{q_l(k)\mathbf{p}(k) - 1}{2})q_l(k)\mathbf{p}(k)\text{Guess}_A(k). \end{aligned} \quad (4)$$

Note that in game \mathbf{G}_3 , the oracle answers of RO_A and the answers given to the LR oracle in reply to its RO queries are independent. Therefore, game \mathbf{G}_4 is a simplified version of \mathbf{G}_3 , which implies that

$$\Pr[\mathbf{G}_4(k)] = \Pr[\mathbf{G}_3(k)]. \quad (5)$$

Games \mathbf{G}_5 and \mathbf{G}_4 are identical-until- bad_3 . Similarly, denote by $\Pr[\text{bad}_3]$ the probability that \mathbf{G}_5 sets bad_3 . We have that $|\Pr[\mathbf{G}_5(k)] - \Pr[\mathbf{G}_4(k)]| \leq$

<p>Game G_6 $(pk, sk) \leftarrow \text{Kg}(1^k)$; $b \leftarrow \{0, 1\}$; $C \leftarrow \emptyset$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k)$; $b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return $(b' = b)$</p> <p>On query $\text{RO}_A(xk', m', n')$: If $H_A[xk', m', n'] = \perp$, then $H_A[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H_A[xk', m', n']$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ For $i \in [\mathbf{n}]$, then $\mathbf{r}[i] \leftarrow \mathcal{R}_{\text{Enc}}$ $\mathbf{c}[i] \leftarrow \text{Enc}(pk, \mathbf{m}_b[i]; \mathbf{r}[i])$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: If $c' \in C$, then return \perp $m' \leftarrow \text{Dec}(sk, c')$ Return m'</p>	<p>Adv $B^{\text{ENC}_B, \text{DEC}_B}(pk)$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k)$ $b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return b'</p> <p>On query $\text{RO}_A(xk', m', n')$: If $H_A[xk', m', n'] = \perp$, then $H_A[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H_A[xk', m', n']$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{c} \leftarrow \text{ENC}_B(\mathbf{m}_0, \mathbf{m}_1)$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: $m' \leftarrow \text{DEC}_B(c')$ Return m'</p>
---	--

Fig. 8. Game G_6 (left) and adversary B (right) in the proof of Theorem 2. Note that in this paper we extend the set membership notations to vectors, writing $X \cup \mathbf{x}$ to mean $X \cup \{\mathbf{x}[i] | i \in [|\mathbf{x}|]\}$.

$\Pr[\text{bad}_3]$. G_5 sets bad_3 only if there is some tuple (xk', m', n') which has been queried by the LR oracle at least twice. Since A has high min-entropy, for any $(xk', m', n') \in T_2$, any \mathcal{M} queried by A , any $b \in \{0, 1\}$, and any $i \in [p(k)]$, $\Pr[(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) = (xk', m', n') : (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}] \leq \text{Guess}_A(k)$ is negligible. Notice that A makes totally $q_l(k)$ LR queries, and for each LR query \mathcal{M} , the LR oracle makes $p(k)$ RO queries, so we derive that $\Pr[\text{bad}_3] \leq \frac{q_l(k)p(k)(q_l(k)p(k)-1)}{2} \text{Guess}_A(k)$. Therefore,

$$|\Pr[G_5(k)] - \Pr[G_4(k)]| \leq \frac{q_l(k)p(k)(q_l(k)p(k) - 1)}{2} \text{Guess}_A(k). \tag{6}$$

Note that in game G_5 , both T_1 and T_2 are useless, and the vector \mathbf{r} generated by the LR oracle is truly random from A 's point of view. Therefore, game G_6 is a simplified version of G_5 , which implies that

$$\Pr[G_6(k)] = \Pr[G_5(k)]. \tag{7}$$

Next, we construct an IND-CCA adversary B attacking PKE as shown in Fig. 8. In order to distinguish B 's own decryption oracle (in the sense of IND-CCA) and A 's decryption oracle (in the sense of IND-CDA2), we denote by DEC_B (resp. ENC_B) B 's decryption (resp. encryption) oracle. B uses ENC_B to answer A 's LR queries, and uses DEC_B to answer A 's decryption queries. B perfectly simulates game G_6 for A , and that B wins game $G_{\text{PKE}, B}^{\text{ind-cca}}$ if and only if A wins game G_6 . Hence,

$$\Pr[G_{\text{PKE}, B}^{\text{ind-cca}}(k)] = \Pr[G_6(k)]. \tag{8}$$

Combining Eqs. (1)-(8), we derive that

$$\begin{aligned} \mathbf{Adv}_{\text{RtP},A}^{\text{ind-cda2}}(k) &\leq \mathbf{Adv}_{\text{PKE},B}^{\text{ind-cca}}(k) + 4\mathbf{Adv}_{\text{PKE},B_{\text{upr}}}^{\text{ind-cca}}(k) \\ &\quad + (6q_r(k) + 2q_l(k)\mathfrak{p}(k) - 2)q_l(k)\mathfrak{p}(k)\text{Guess}_A(k). \end{aligned}$$

Now, we catch up with the proof of Lemma 1.

Proof (of Lemma 1). We say that “ \mathbf{G}_4 sets bad_2 ” (resp. “ \mathbf{G}_5 sets bad_2 ”) if A submits an RO_A query (xk', m', n') , such that $(xk', m', n') \in T_2$, in \mathbf{G}_4 (resp. \mathbf{G}_5).

Since \mathbf{G}_4 is a simplified version of \mathbf{G}_3 , and \mathbf{G}_5 and \mathbf{G}_4 are identical-until- bad_3 ,

$$\begin{aligned} \Pr[\text{bad}_2] &= \Pr[\mathbf{G}_4 \text{ sets } \text{bad}_2] \leq \Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2] + \Pr[\text{bad}_3] \\ &\leq \Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2] + \frac{q_l(k)\mathfrak{p}(k)(q_l(k)\mathfrak{p}(k) - 1)}{2}\text{Guess}_A(k). \end{aligned} \quad (9)$$

To bound $\Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2]$, we consider an IND-CCA adversary B_{upr} as shown in Fig. 9. Similarly, denote by $\text{ENC}_{B_{\text{upr}}}$ (resp. $\text{DEC}_{B_{\text{upr}}}$) B_{upr} 's encryption (resp. decryption) oracle in the sense of IND-CCA. Let \tilde{b} be the challenge bit in game $\mathbf{G}_{\text{PKE},B_{\text{upr}}}^{\text{ind-cca}}$. Denote by $\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}}$ the game simulated by B_{upr} for A (as shown in Fig. 9). B_{upr} 's advantage is as follows.

$$\mathbf{Adv}_{\text{PKE},B_{\text{upr}}}^{\text{ind-cca}}(k) = 2\Pr[\mathbf{G}_{\text{PKE},B_{\text{upr}}}^{\text{ind-cca}}(k)] - 1 = 2\Pr[b^* = \tilde{b}] - 1 \quad (10)$$

$$= 2(\Pr[b^* = \tilde{b} \mid \tilde{b} = a]\Pr[\tilde{b} = a] + \Pr[b^* = \tilde{b} \mid \tilde{b} \neq a]\Pr[\tilde{b} \neq a]) - 1 \quad (11)$$

$$= \Pr[b^* = \tilde{b} \mid \tilde{b} = a] + \Pr[b^* = \tilde{b} \mid \tilde{b} \neq a] - 1 \quad (12)$$

Equations (10)-(11) are trivial. Since a is uniformly random chosen from $\{0, 1\}$, $\Pr[\tilde{b} = a] = \Pr[\tilde{b} \neq a] = \frac{1}{2}$. This justifies Eq. (12).

For $\Pr[b^* = \tilde{b} \mid \tilde{b} = a]$, we have the following equations.

$$\begin{aligned} &\Pr[b^* = \tilde{b} \mid \tilde{b} = a] \\ &= \Pr[b^* = \tilde{b} \mid (\tilde{b} = a) \wedge (\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2)]\Pr[\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2 \mid \tilde{b} = a] \\ &\quad + \Pr[b^* = \tilde{b} \mid (\tilde{b} = a) \wedge \neg(\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2)] \\ &\quad \cdot \Pr[\neg(\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2) \mid \tilde{b} = a] \end{aligned} \quad (13)$$

$$= \Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2] + \frac{1}{2}\Pr[\neg(\mathbf{G}_5 \text{ sets } \text{bad}_2)] \quad (14)$$

$$= \frac{1}{2}\Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2] + \frac{1}{2}. \quad (15)$$

Equation (13) is trivial. We notice that when $\tilde{b} = a$, the simulated game $\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}}$ is the same as \mathbf{G}_5 from A 's point of view, so we have $\Pr[\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2 \mid \tilde{b} = a] = \Pr[\mathbf{G}_5 \text{ sets } \text{bad}_2]$. We also note that if $\mathbf{G}_{B_{\text{upr}},A}^{\text{sim}} \text{ sets } \text{bad}_2$, then B_{upr}

outputs $b^* = a$, otherwise B_{upr} outputs $b^* \leftarrow \{0, 1\}$. Therefore, $\Pr[b^* = \tilde{b} \mid (\tilde{b} = a) \wedge (\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] = 1$ and $\Pr[b^* = \tilde{b} \mid (\tilde{b} = a) \wedge \neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] = \frac{1}{2}$. This justifies Eq. (14). Equation (15) is because $\Pr[\neg(\mathbf{G}_5 \text{ sets bad}_2)] = 1 - \Pr[\mathbf{G}_5 \text{ sets bad}_2]$.

With similar analysis, for $\Pr[b^* = \tilde{b} \mid \tilde{b} \neq a]$, we have the following equations.

$$\begin{aligned} & \Pr[b^* = \tilde{b} \mid \tilde{b} \neq a] \\ &= \Pr[b^* = \tilde{b} \mid (\tilde{b} \neq a) \wedge (\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] \Pr[\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2 \mid \tilde{b} \neq a] \\ & \quad + \Pr[b^* = \tilde{b} \mid (\tilde{b} \neq a) \wedge \neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] \\ & \quad \cdot \Pr[\neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2) \mid \tilde{b} \neq a] \end{aligned} \quad (16)$$

$$= 0 + \frac{1}{2} \Pr[\neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2) \mid \tilde{b} \neq a] \quad (17)$$

$$= \frac{1}{2} (1 - \Pr[\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2 \mid \tilde{b} \neq a]) \quad (18)$$

$$\geq \frac{1}{2} (1 - q_l(k)q_r(k)\mathfrak{p}(k)\text{Guess}_A(k)). \quad (19)$$

Equation (16) is trivial. B_{upr} outputs $b^* = a$ when $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$ sets bad_2 , so we have that $\Pr[b^* = \tilde{b} \mid (\tilde{b} \neq a) \wedge (\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] = 0$. Considering that B_{upr} outputs $b^* \leftarrow \{0, 1\}$ when $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$ does not set bad_2 , so we have $\Pr[b^* = \tilde{b} \mid (\tilde{b} \neq a) \wedge \neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2)] = \frac{1}{2}$. We have justified Eq. (17). Equation (18) is because $\Pr[\neg(\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2) \mid \tilde{b} \neq a] = 1 - \Pr[\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2 \mid \tilde{b} \neq a]$. Notice that $\tilde{b} \neq a$ implies $\tilde{b} = 1 - a$, i.e., the challenge ciphertext vectors A received are the encryption of some uniformly random chosen message vectors. Thus the challenge ciphertext vectors do not contain any information about any \mathbf{m}_a . Besides, in the simulated game $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$, the answers (of RO_A , the LR oracle, and the decryption oracle) given to A do not contain any information about the \mathbf{xk} and \mathbf{n} sampled by the LR oracle. Therefore, for any tuple $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}$ sampled by the LR oracle in game $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$, A has no additional information about any element of $\{(\mathbf{xk}[i], \mathbf{m}_b[i], \mathbf{n}[i]) \mid i \in [\mathfrak{p}(k)], b \in \{0, 1\}\}$. Recall that $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$ sets bad_2 only if A succeeds in guessing some element in $\{(\mathbf{xk}[i], \mathbf{m}_a[i], \mathbf{n}[i]) \mid i \in [\mathfrak{p}(k)]\}$ for some $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}$ sampled by the LR oracle and the a sampled by B_{upr} . Notice that the total number of random-oracle (resp. LR-oracle) queries of A is $q_r(k)$ (resp. $q_l(k)$). So we derive that $\Pr[\mathbf{G}_{B_{upr}, A}^{\text{sim}} \text{ sets bad}_2 \mid \tilde{b} \neq a] \leq q_l(k)q_r(k)\mathfrak{p}(k)\text{Guess}_A(k)$. We have justified Eq. (19).

Combining Eqs. (12), (15) and (19), we derive that

$$\mathbf{Adv}_{\text{PKE}, B_{upr}}^{\text{ind-cca}}(k) \geq \frac{1}{2} (\Pr[\mathbf{G}_5 \text{ sets bad}_2] - q_l(k)q_r(k)\mathfrak{p}(k)\text{Guess}_A(k)). \quad (20)$$

Hence,

$$\Pr[\mathbf{G}_5 \text{ sets bad}_2] \leq 2\mathbf{Adv}_{\text{PKE}, B_{upr}}^{\text{ind-cca}}(k) + q_l(k)q_r(k)\mathfrak{p}(k)\text{Guess}_A(k). \quad (21)$$

Combining Eqs. (9) and (21), we obtain that

$$\Pr[\text{bad}_2] \leq 2\mathbf{Adv}_{\text{PKE}, B_{\text{upr}}}^{\text{ind-cca}}(k) + (q_r(k) + \frac{q_l(k)\rho(k) - 1}{2})q_l(k)\rho(k)\text{Guess}_A(k).$$

□

Remark 4. The N-PKE scheme RtP is a special case of the ROM scheme NPE in [8, Fig. 6], but it seems that the original, generic ROM scheme NPE proposed in [8, Fig. 6] does not achieve IND-CDA2 security. The reason is as follows. In [8], the security of NPE is guaranteed by the IND-CCA security of the traditional PKE scheme, and the prf security and the ror security (defined in [8]) of their proposed building block, hedged extractor. The prf security focuses on the case that the seeds are random and confidential, and the ror security focuses on the case that the nonces are unpredictable. In other words, the security of hedged extractor just considers the case that either the seeds or the nonces have high entropy. And the IND-CDA2 security of N-PKE should be guaranteed as long as the seeds, messages and nonces jointly have high min-entropy.

With respect to RIND-CDA2 security, with similar technique we have the following corollary.

Corollary 1. *If PKE is a traditional IND-CCA secure PKE scheme, then N-PKE scheme RtP, w.r.t. a nonce generator NG, is RIND-CDA2 secure in the random oracle model.*

5 Construction of H-PKE in the Standard Model

Generic construction. Let $\text{NE} = (\text{NKg}, \text{NSKg}, \text{NEnc}, \text{NDec})$ be an N-PKE scheme, w.r.t. a nonce generator NG. Let $\text{DE} = (\text{DKg}, \text{DEnc}, \text{DDec})$ be a D-PKE scheme. Recall the transform Nonce-then-Deterministic NtD = $(\text{NDKg}, \text{NDSKg}, \text{NDEnc}, \text{NDDec})$ proposed in [16] as shown in Fig. 10.

In [16], Hoang et al. consider SOA security of NtD, showing that if NE is N-SO-CPA (resp. N-SO-CCA) secure, and DE is D-SO-CPA (resp. D-SO-CCA and unique-ciphertext) secure, then NtD is HN-SO-CPA (resp. HN-SO-CCA) secure. The HN-SOA security notions formalized in [16] are non-adaptive. Therefore, the HN-SO-CCA security formalized in [16] does not imply our HN-IND security.

In this section, we point out that NtD also applies to the HN-IND setting. Specifically, we assume NE is NBP1 and NBP2 secure, and DE is adaptively CCA secure and unique-ciphertext. Additionally, we require that NE is *entropy-preserving*, which is a property of N-PKE formalized by Hoang et al. [16].

Denote by $\text{Entr}_{\text{NE}}(\theta(k))$ the conditional min-entropy of $\text{NEnc}(pk_n, xk, m, n)$ given X , where X is a random variable such that the conditional min-entropy of (xk, m, n) is at least $\theta(k)$, and $(pk_n, sk_n) \leftarrow \text{NKg}(1^k)$ is independent of (xk, m, n, X) . NE is called *entropy-preserving*, if for any $\theta(k)$ satisfying that $2^{-\theta(k)}$ is negligible, then $2^{-\text{Entr}_{\text{NE}}(\theta(k))}$ is also negligible.

Formally, we have the following theorem.

<p>Adversary $\text{ENC}_{B_{upr}} \cdot \text{DEC}_{B_{upr}}(pk)$</p> <p>$a, b^* \leftarrow \{0, 1\}; T_1, T_2 \leftarrow \emptyset$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k); b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return b^*</p> <p>On query $\text{RO}_A(xk', m', n')$: $T_1 \leftarrow T_1 \cup \{(xk', m', n')\}$ If $(xk', m', n') \in T_2$, then $b^* \leftarrow a$ If $H_A[xk', m', n'] = \perp$, then $H_A[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H_A[xk', m', n']$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{m}_a^{\text{ch}} \leftarrow \mathbf{m}_a$ $\mathbf{m}_{1-a}^{\text{ch}} \leftarrow \text{MSP}^{ \mathbf{n} }$ For $i \in [\mathbf{n}]$, $T_2 \leftarrow T_2 \cup \{(\mathbf{xk}[i], \mathbf{m}_a[i], \mathbf{n}[i])\}$ $\mathbf{c} \leftarrow \text{ENC}_{B_{upr}}(\mathbf{m}_0^{\text{ch}}, \mathbf{m}_1^{\text{ch}})$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: $m' \leftarrow \text{DEC}_{B_{upr}}(c')$ Return m'</p>	<p>Game $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$</p> <p>$a, b^* \leftarrow \{0, 1\}; C, T_1, T_2 \leftarrow \emptyset; \tilde{b} \leftarrow \{0, 1\}$ $St \leftarrow A_1^{\text{RO}_A, \text{LR}}(1^k); b' \leftarrow A_2^{\text{RO}_A, \text{DEC}}(pk, St)$ Return b^*</p> <p>On query $\text{RO}_A(xk', m', n')$: $T_1 \leftarrow T_1 \cup \{(xk', m', n')\}$ If $(xk', m', n') \in T_2$, then $\text{bad}_2 \leftarrow \text{true}; b^* \leftarrow a$ If $H_A[xk', m', n'] = \perp$, then $H_A[xk', m', n'] \leftarrow \mathcal{R}_{\text{Enc}}$ Return $H_A[xk', m', n']$</p> <p>On query $\text{LR}(\mathcal{M})$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{m}_a^{\text{ch}} \leftarrow \mathbf{m}_a$ $\mathbf{m}_{1-a}^{\text{ch}} \leftarrow \text{MSP}^{ \mathbf{n} }$ For $i \in [\mathbf{n}]$, $T_2 \leftarrow T_2 \cup \{(\mathbf{xk}[i], \mathbf{m}_a[i], \mathbf{n}[i])\}$ $\mathbf{r}[i] \leftarrow \mathcal{R}_{\text{Enc}}$ $\mathbf{c}[i] \leftarrow \text{Enc}(pk, \mathbf{m}_b^{\text{ch}}[i]; \mathbf{r}[i])$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query $\text{DEC}(c')$: If $c' \in C$, then return \perp $m' \leftarrow \text{Dec}(sk, c')$ Return m'</p>
---	--

Fig. 9. Adversary B_{upr} (left) and game $\mathbf{G}_{B_{upr}, A}^{\text{sim}}$ (right) in the proof of Lemma 1.

$\text{NDKkg}(1^k)$ $(pk_n, sk_n) \leftarrow \text{NKg}(1^k)$ $(pk_d, sk_d) \leftarrow \text{DKg}(1^k)$ $pk \leftarrow (pk_n, pk_d)$ $sk \leftarrow (sk_n, sk_d)$ Return (pk, sk)	$\text{NDSKg}(1^k)$ $xk \leftarrow \text{NSKg}(1^k)$ Return xk	$\text{NDEnc}(pk, xk, m, n)$ Parse $pk = (pk_n, pk_d)$ $y \leftarrow \text{NEnc}(pk_n, xk, m, n)$ $c \leftarrow \text{DEnc}(pk_d, y)$ Return c	$\text{NDDec}(sk, c)$ Parse $sk = (sk_n, sk_d)$ $y \leftarrow \text{DDec}(sk_d, c)$ $m \leftarrow \text{NDec}(sk_n, y)$ Return m
--	--	--	--

Fig. 10. N-PKE scheme NtD = (NDKg, NDSKg, NDEnc, NDDec).

Theorem 3. For an NBP1, NBP2 secure and entropy-preserving N-PKE scheme NE and a D-PKE scheme DE, let NtD be an N-PKE scheme defined in Fig. 10.

- (i) If DE is adaptively CCA secure and unique-ciphertext, then NtD is HN-IND secure.
- (ii) If DE is ACD-CCA secure and unique-ciphertext, then NtD is RSV-version HN-IND secure.

Proof. Firstly, we prove that NtD is NBP1 secure. The proof of NBP2 security is similar, which we will omit here.

For any NBP1 adversary A attacking NtD, we present an NBP1 adversary B_{nbp1} attacking NE as shown in Fig. 11. Denote by ENC_B (resp. DEC_B) B_{nbp1} 's encryption (resp. decryption) oracle in the sense of NBP1. Note that DE is unique-ciphertext. As a result, for any decryption query c' of A , if $y' \leftarrow \text{DDec}(sk_d, c')$ is one of the challenge ciphertext B_{nbp1} received, then c' is also one of the challenge ciphertext A received. Thus the DEC oracle

simulated by B_{nbp1} is identical to the real DEC oracle in game $\mathbf{G}_{\text{NtD},A}^{\text{nbp1}}$. It's easy to see that the ENC oracle simulated by B_{nbp1} is identical to the real ENC oracle of A . Therefore, B_{nbp1} perfectly simulates game $\mathbf{G}_{\text{NtD},A}^{\text{nbp1}}$ for A , and B_{nbp1} wins game $\mathbf{G}_{\text{NE},B_{nbp1}}^{\text{nbp1}}$ if and only if A wins $\mathbf{G}_{\text{NtD},A}^{\text{nbp1}}$. So we derive that $\text{Adv}_{\text{NtD},A}^{\text{nbp1}}(k) = \text{Adv}_{\text{NE},B_{nbp1}}^{\text{nbp1}}(k)$.

$\text{Adv}_{B_{nbp1}}^{\text{ENC}_B, \text{DEC}_B}(pk_n)$ $(pk_d, sk_d) \leftarrow \text{DKg}(1^k)$ $pk \leftarrow (pk_n, pk_d)$ $b' \leftarrow A_{\text{ENC}, \text{DEC}}^{\text{ENC}_B}(pk)$ Return b'	On query $\text{ENC}(m_0, m_1, \eta)$: $y \leftarrow \text{ENC}_B(m_0, m_1, \eta)$ $c \leftarrow \text{DEnc}(pk_d, y)$ Return c	On query $\text{DEC}(c')$: $y' \leftarrow \text{DDec}(sk_d, c')$ $m' \leftarrow \text{DEC}_B(y')$ Return m'
$\text{Adv}_{B_1}^{\text{ENC}_B}(1^k)$ $(pk_n, sk_n) \leftarrow \text{NKg}(1^k)$ $St \leftarrow A_1^{\text{LR}}(1^k)$ $St_B \leftarrow (pk_n, sk_n, St)$ Return St_B	$\text{Adv}_{B_2}^{\text{DEC}_B}(pk_d, St_B)$ Parse $St_B = (pk_n, sk_n, St)$ $pk \leftarrow (pk_n, pk_d)$ $b' \leftarrow A_2^{\text{DEC}}(pk, St)$ Return b'	On query $\text{LR}(\mathcal{M})$: $c \leftarrow \text{ENC}_B(\text{MST}_{\text{n-d}}(\mathcal{M}, pk_n))$ Return c
On query $\text{DEC}(c')$: $y' \leftarrow \text{DEC}_B(c')$ If $y' = \perp$, then return \perp $m' \leftarrow \text{NDec}(sk_n, y')$ Return m'	Alg. $\text{MST}_{\text{n-d}}(\mathcal{M}, pk_n)(1^k)$: $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{y}_0 \leftarrow \text{NEnc}(pk_n, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $\mathbf{y}_1 \leftarrow \text{NEnc}(pk_n, \mathbf{xk}, \mathbf{m}_1, \mathbf{n})$ Return $(\mathbf{y}_0, \mathbf{y}_1)$	

Fig. 11. Adversary B_{nbp1} (up) and adversary B (down) in the proof of Theorem 3.

Next, we show that NtD is IND-CDA2 secure. We call a PPT algorithm $\text{MST}_{\text{n-d}}$ a *message sampler transformer from N-PKE to D-PKE*, if it takes a message sampler for N-PKE (and some state information) as input, and acts as a message sampler for D-PKE (see Fig. 11). For any legitimate PPT IND-CDA2 adversary A having high min-entropy, we construct a $\text{MST}_{\text{n-d}}$ and an adaptively CCA adversary $B = (B_1, B_2)$ attacking DE as shown in Fig. 11. Similarly, denote by ENC_B (resp. DEC_B) B 's encryption (resp. decryption) oracle in the sense of adaptive CCA. B perfectly simulates game $\mathbf{G}_{\text{NtD},A}^{\text{ind-cda2}}$ for A . Since NE is entropy-preserving, the construction of $\text{MST}_{\text{n-d}}$ guarantees that B is legitimate and has high min-entropy. Note that B wins game $\mathbf{G}_{\text{DE},B}^{\text{cca}}$ if and only if A wins $\mathbf{G}_{\text{NtD},A}^{\text{ind-cda2}}$. So we derive that $\text{Adv}_{\text{NtD},A}^{\text{ind-cda2}}(k) = \text{Adv}_{\text{DE},B}^{\text{cca}}(k)$.

With similar techniques, we can prove the RIND-CDA2 security of NtD. \square

Remark 5. Theorem 3 applies to both the ROM constructions and the standard-model constructions.

Concrete constructions. According to Theorem 3, let NE be the NBP1 and NBP2 secure standard-model construction proposed in [8], and DE be the ACD-CCA secure standard-model construction proposed in [20], then we obtain an RSV-version HN-IND secure N-PKE scheme NtD in the standard model.

Now we turn to HN-IND security of NtD. According to Theorem 3, what remains is to construct a (unique-ciphertext) standard-model D-PKE scheme achieving adaptively CCA security. Considering IND-CDA2 security in the setting of H-PKE, instead of N-PKE, if the length of the randomness is zero (i.e.,

$|\mathbf{r}[i]| = 0$ for all $i \in [|p(k)|]$), then IND-CDA2 security actually becomes adaptive CCA security for D-PKE. Therefore, the problem that construct an IND-CDA2 secure N-PKE scheme in the standard model is at least as hard as the one that construct a fully adaptively CCA secure D-PKE scheme in the standard model. To the best of our knowledge, the latter is still an open problem. On the other hand, Theorem 3 shows that if an adaptively CCA secure (and unique-ciphertext) standard-model D-PKE scheme is constructed, then we will have an N-PKE scheme achieving HN-IND security in the standard model.

Some notes on adaptively CCA secure D-PKE. Recall that Bellare et al. [2] presented an adaptively IND secure D-PKE scheme, by showing any PKE scheme, achieving a special anonymity (i.e., the ANON security in [2]) and non-adaptive IND-CDA security simultaneously, achieves (adaptively) IND-CDA security. Although the conclusion cannot be employed to show an adaptively CCA secure D-PKE scheme directly, we note that it can be transformed to the setting of N-PKE under CCA attacks. For completeness, we present the transform in Appendix A.

More specifically, in Appendix A, we formalize the notion of ANON-CCA security for N-PKE, and show that if an N-PKE scheme achieves non-adaptive IND-CDA (*not IND-CDA2*) and ANON-CCA security, then it achieves IND-CDA2 security. We stress that very recently, Boldyreva et al. [10] showed a similar conclusion (for general PKE). But their formalized ANON-CCA security is stronger than ours (i.e., informally, the adversary can make decryption queries under two secret keys). More importantly, their conclusion, informally with our notations in this paper, is that “non-adaptive IND-CDA2 + stronger ANON-CCA \Rightarrow IND-CDA2”. Our conclusion shows that the same result can be obtained under some weaker assumptions.

Acknowledgment. We thank the anonymous reviewers for their helpful comments. The first author was supported by National Natural Science Foundation of China (No. 61702125), and Scientific Research Foundation for Post-doctoral Researchers of Guangzhou (No. gdbsh2016020). The second author was National Natural Science Foundation of China (No. 61572235), Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2015A030306045), and Pearl River S&T Nova Program of Guangzhou. The third author was partly supported by the Program for Innovative Research Team in Education Department of Guangdong Province Under Grant No. 2015KCXTD014. and No. 2016KCXTD017. The sixth author was supported by National Natural Science Foundation of China (No. 61472091), National Natural Science Foundation for Outstanding Youth Foundation (No. 61722203), and the State Key Laboratory of Cryptology, Beijing, China.

A From Non-adaptive IND-CDA to Adaptive IND-CDA2

Firstly, we formalize the notion of anonymity for IND-CDA2 for N-PKE. Then we will present our theorem. Consider game $\mathbf{G}_{\text{NE},A}^{\text{anon-cca}}$ as shown in Fig. 12. For the adversary A in game $\mathbf{G}_{\text{NE},A}^{\text{anon-cca}}$, we can similarly define *legitimate* adversary,

and the adversary *having high min-entropy (resp. high block-source min-entropy)*. We do not repeat the details here.

Definition 7 (ANON-CCA). *An N-PKE scheme NE is ANON-CCA secure (resp. block-source ANON-CCA secure), if for any legitimate PPT adversary A having high min-entropy (resp. high block-source min-entropy), its advantage $\text{Adv}_{\text{NE},A}^{\text{anon-cca}}(k) = 2\text{Pr}[\mathbf{G}_{\text{NE},A}^{\text{anon-cca}}(k)] - 1$ is negligible, where game $\mathbf{G}_{\text{NE},A}^{\text{anon-cca}}$ is defined in Fig. 12.*

Game $\mathbf{G}_{\text{NE},A}^{\text{anon-cca}}(k)$	$\text{ENC}_0(\mathcal{M})$	$\text{LR}(\mathcal{M})$	$\text{DEC}_0(c)$
$(pk_0, sk_0) \leftarrow \text{NKG}(1^k)$	If $kr = \text{true}$, then	If $kr = \text{true}$, then	If $kr = \text{false}$, then
$(pk_1, sk_1) \leftarrow \text{NKG}(1^k)$	return \perp	return \perp	return \perp
$b \leftarrow \{0, 1\}; C \leftarrow \emptyset$	$(\mathbf{m}, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$	$kr \leftarrow \text{true}$	If $c \in C$, then
$kr \leftarrow \text{false}$	$\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}, \mathbf{n})$	$(\mathbf{m}, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$	return \perp
$b' \leftarrow A^{\text{ENC}_0, \text{LR}, \text{DEC}_0}(1^k)$	$C \leftarrow C \cup \mathbf{c}$	$\mathbf{c} \leftarrow \text{NEnc}(pk_b, \mathbf{xk}, \mathbf{m}, \mathbf{n})$	$m \leftarrow \text{NDec}(sk_0, c)$
Return $(b' = b)$	Return \mathbf{c}	$C \leftarrow C \cup \mathbf{c}$	Return m
		Return (pk_0, pk_1, \mathbf{c})	

Fig. 12. Game for defining ANON-CCA security of an N-PKE scheme NE.

Theorem 4. *Let NE be an N-PKE scheme. If NE achieves non-adaptive IND-CDA security and ANON-CCA security simultaneously, then it also achieves adaptive IND-CDA2 security.*

Proof. The proof is based on the approach proposed in [2]. For any PPT legitimate IND-CDA2 adversary A having high min-entropy and making $q(k)$ LR queries, denote by $\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-}b}$ ($b \in \{0, 1\}$) the game as follows: $\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-}b}$ is the same as $\mathbf{G}_{\text{NE},A}^{\text{ind-cda2}}$, except that b is a fixed value in $\{0, 1\}$, and the final output of this game is b' . A standard argument shows that the advantage of A can be written as $\text{Adv}_{\text{NE},A}^{\text{ind-cda2}}(k) = |\text{Pr}[\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-1}}(k)] - \text{Pr}[\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-0}}(k)]|$. Games $\mathbf{G}_{\text{NE},A}^{\text{ind-cda-}b}$, $\mathbf{G}_{\text{NE},A}^{\text{anon-cca-}b}$ ($b \in \{0, 1\}$) (resp. the corresponding advantages) can be defined (resp. written) similarly.

Consider the sequence of games in Fig. 13. Game \mathbf{G}_{-1} is the same as game $\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-0}}$, i.e., $\text{Pr}[\mathbf{G}_{-1}] = \text{Pr}[\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-0}}]$. Game \mathbf{G}_0 introduces (pk_1, sk_1) , which is useless in \mathbf{G}_0 . So we have $\text{Pr}[\mathbf{G}_0] = \text{Pr}[\mathbf{G}_{-1}]$. For $0 \leq i \leq q(k)$, game \mathbf{G}_i is the same as \mathbf{G}_0 , except that for the j^{th} LR query of A , if $j \leq i$, the challenge ciphertext vector \mathbf{c} is an encryption of \mathbf{m}_1 (under the public key pk_0), instead of an encryption of \mathbf{m}_0 . Therefore, $\mathbf{G}_{q(k)}$ is identical to $\mathbf{G}_{\text{NE},A}^{\text{ind-cda2-1}}$. Hence, what remains is to show the indistinguishability between \mathbf{G}_i and \mathbf{G}_{i+1} for any $0 \leq i \leq q(k) - 1$.

As shown in Fig. 13, for $0 \leq i \leq q(k) - 1$, game \mathbf{H}_i is the same as \mathbf{G}_i , except that for the i^{th} LR query of A , the challenge ciphertext vector \mathbf{c} is an encryption of \mathbf{m}_0 under pk_1 , instead of an encryption of \mathbf{m}_1 under pk_0 . Game \mathbf{H}'_i is the same as \mathbf{H}_i , except that for the i^{th} LR query of A , the challenge ciphertext vector \mathbf{c} is an encryption of \mathbf{m}_1 under pk_1 , instead of an encryption of \mathbf{m}_0 under pk_1 . Formally, we have three claims below.

<p>Games $\mathbf{G}_{-1}, \boxed{\mathbf{G}_0}$</p> <p>$(pk_0, sk_0) \leftarrow \text{NKG}(1^k); (pk_1, sk_1) \leftarrow \text{NKG}(1^k)$ $C \leftarrow \emptyset; j \leftarrow 0$ $St \leftarrow A_1^{\text{LR}}(1^k); b' \leftarrow A_2^{\text{DEC}}(pk_0, St)$ Return b'</p> <p>On query LR(\mathcal{M}): $j \leftarrow j + 1; (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query DEC(c'): If $c' \in C$, then return \perp $m' \leftarrow \text{NDec}(c')$ Return m'</p>	<p>Games $\mathbf{G}_i, \boxed{\mathbf{H}_i}, \mathbf{H}'_i$</p> <p>$(pk_0, sk_0) \leftarrow \text{NKG}(1^k); (pk_1, sk_1) \leftarrow \text{NKG}(1^k)$ $C \leftarrow \emptyset; j \leftarrow 0$ $St \leftarrow A_1^{\text{LR}}(1^k); b' \leftarrow A_2^{\text{DEC}}(pk_0, St)$ Return b'</p> <p>On query LR(\mathcal{M}): $j \leftarrow j + 1; (\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ If $j < i$, then $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_1, \mathbf{n})$ If $j = i$, then $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_1, \mathbf{n})$ $\text{If } j = i, \text{ then } \mathbf{c} \leftarrow \text{NEnc}(pk_1, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $\text{If } j = i, \text{ then } \mathbf{c} \leftarrow \text{NEnc}(pk_1, \mathbf{xk}, \mathbf{m}_1, \mathbf{n})$ If $j > i$, then $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p> <p>On query DEC(c'): If $c' \in C$, then return \perp $m' \leftarrow \text{NDec}(c')$ Return m'</p>
---	---

Fig. 13. Games $\mathbf{G}_{-1} - \mathbf{G}_q$ and $\mathbf{H}_0 - \mathbf{H}_{q-1}$ in the proof of Theorem 4. Boxed code is only executed in the games specified by the game names in the same box style.

Claim 1. For any $1 \leq i \leq q(k)$, there is an ANON-CCA adversary B_{an1} such that $\text{Adv}_{\text{NE}, B_{an1}}^{\text{anon-cca}}(k) = |\Pr[\mathbf{G}_{i-1}] - \Pr[\mathbf{H}_i]|$.

Claim 2. For any $1 \leq i \leq q(k)$, there is a non-adaptively IND-CDA adversary B such that $\text{Adv}_{\text{NE}, B}^{\text{ind-cda}}(k) = |\Pr[\mathbf{H}_i] - \Pr[\mathbf{H}'_i]|$.

Claim 3. For any $1 \leq i \leq q(k)$, there is an ANON-CCA adversary B_{an2} such that $\text{Adv}_{\text{NE}, B_{an2}}^{\text{anon-cca}}(k) = |\Pr[\mathbf{H}'_i] - \Pr[\mathbf{G}_i]|$.

Combining these three claims, we derive that

$$\begin{aligned} \text{Adv}_{\text{NE}, A}^{\text{ind-cda}^2}(k) &= |\Pr[\mathbf{G}_{q(k)}] - \Pr[\mathbf{G}_0]| \\ &= q(k)(\text{Adv}_{\text{NE}, B}^{\text{ind-cda}}(k) + \text{Adv}_{\text{NE}, B_{an1}}^{\text{anon-cca}}(k) + \text{Adv}_{\text{NE}, B_{an2}}^{\text{anon-cca}}(k)). \end{aligned}$$

Therefore, what remains is to prove the above three claims. The proof of Claim 3 is similar to that of Claim 1. So we omit it here.

Proof (of Claim 1). Note that in game \mathbf{G}_{i-1} ($1 \leq i \leq q(k)$), for the j^{th} LR query of A , if $j \leq i - 1$, the challenge ciphertext vector \mathbf{c} is an encryption of \mathbf{m}_1 under pk_0 , and if $j \geq i$, \mathbf{c} is an encryption of \mathbf{m}_0 under pk_0 . Therefore, \mathbf{H}_i is identical to \mathbf{G}_{i-1} , except that the answer A received to its i^{th} LR query is an encryption of \mathbf{m}_0 under pk_1 , instead of the encryption of \mathbf{m}_0 under pk_0 . For any message sampler \mathcal{M} output by A , and any $b \in \{0, 1\}$, we define a new message sampler \mathcal{M}_b as follows: run $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$, and return $(\mathbf{m}_b, \mathbf{xk}, \mathbf{n})$. We construct a PPT legitimate ANON-CCA adversary B_{an1} in Fig. 14. Denote by LR_B the LR oracle of B_{an1} , and b the challenge bit of $\mathbf{G}_{\text{NE}, B_{an1}}^{\text{anon-cca}}$. When $b = 1$ (resp. $b = 0$), B_{an1} perfectly simulates game \mathbf{H}_i (resp. game \mathbf{G}_{i-1}) for A . So we conclude this proof.

<p>Adv $B_{an1}^{\text{ENC}_0, \text{LR}_B, \text{DEC}_0}(1^k)$</p> <p>$C \leftarrow \emptyset; j \leftarrow 0$ $St \leftarrow A_1^{\text{LR}}(1^k)$ $b' \leftarrow A_2^{\text{DEC}}(pk_0, St)$ Return b'</p>	<p>On query $\text{LR}(\mathcal{M})$:</p> <p>$j \leftarrow j + 1$ If $j \leq i - 1$, then $\mathbf{c} \leftarrow \text{ENC}_0(\mathcal{M}_1)$ If $j = i$, then $(pk_0, pk_1, \mathbf{c}) \leftarrow \text{LR}_B(\mathcal{M}_0)$ If $j > i$, then $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p>	<p>On query $\text{DEC}(c')$:</p> <p>If $c' \in C$, then return \perp $m' \leftarrow \text{DEC}_0(c')$ Return m'</p>
<p>Adv $B_1^{\text{LR}_B}(1^k)$</p> <p>$(pk_0, sk_0) \leftarrow \text{NKG}(1^k)$ $C \leftarrow \emptyset; j \leftarrow 0$ $(St', \mathcal{M}) \leftarrow A_1^{\text{LR}_1}(1^k)$ $\mathbf{c}^* \leftarrow \text{LR}_B(\mathcal{M})$ $St_B \leftarrow (C, j, St', \mathbf{c}^*)$ Return St_B</p>	<p>Adv $B_2(pk_1, St_B)$</p> <p>Parse $St_B = (C, j, St', \mathbf{c}^*)$ $St \leftarrow A_1^{\text{LR}_2}(St', \mathbf{c})$ $b' \leftarrow A_2^{\text{DEC}}(pk_0, St)$ Return b'</p>	
<p>On query $\text{LR}_1(\mathcal{M})$:</p> <p>$j \leftarrow j + 1$ If $j \geq i$, then return \perp $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_1, \mathbf{n})$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p>	<p>On query $\text{LR}_2(\mathcal{M})$:</p> <p>$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{xk}, \mathbf{n}) \leftarrow \mathcal{M}(1^k)$ $\mathbf{c} \leftarrow \text{NEnc}(pk_0, \mathbf{xk}, \mathbf{m}_0, \mathbf{n})$ $C \leftarrow C \cup \mathbf{c}$ Return \mathbf{c}</p>	<p>On query $\text{DEC}(c')$:</p> <p>If $c' \in C$, then return \perp $m' \leftarrow \text{NDec}(sk_0, c')$ Return m'</p>

Fig. 14. Adversary B_{an1} (up) in the proof of Claim 1, and adversary B (down) in the proof of Claim 2.

Proof (of Claim 2). Since \mathbf{H}'_i is identical to \mathbf{H}_i , except that the answer A received to its i^{th} LR query is an encryption of \mathbf{m}_1 under pk_1 , instead of the encryption of \mathbf{m}_0 under the same public key. We note that without loss of generality, for any IND-CDA2 adversary $A = (A_1, A_2)$ making $q(k)$ LR queries, and any $i \in [q(k)]$, the procedure of A_1 can be trivially divided into two parts $(A_{1,(I)}, A_{1,(II)})$ as follows, where $A_{1,(I)}$ makes $i - 1$ queries to LR_1 oracle, and $A_{1,(II)}$ makes $q(k) - i$ queries to LR_2 oracle. LR_1, LR_2 denote the LR-oracle interfaces of $A_{1,(I)}, A_{1,(II)}$, respectively.

Adversary $A_1^{\text{LR}}(1^k)$:
 $(St', \mathcal{M}) \leftarrow A_1^{\text{LR}_1}(1^k); \mathbf{c} \leftarrow \text{LR}(\mathcal{M}); St \leftarrow A_1^{\text{LR}_2}(St', \mathbf{c})$
 Return St

We construct a PPT legitimate non-adaptively IND-CDA adversary B as shown in Fig. 14. Let b be the challenge bit of $\mathbf{G}_{\text{NE}, B}^{\text{ind-cda}}$. When $b = 1$ (resp. $b = 0$), B perfectly simulates game \mathbf{H}'_i (resp. game \mathbf{H}_i) for A . Therefore, we conclude this proof. \square

References

1. Bellare, M., Boldyreva, A., O’Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30
2. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: how to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_14

3. Bellare, M., Dowsley, R., Keelveedhi, S.: How secure is deterministic encryption? In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 52–73. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_3
4. Bellare, M., Fischlin, M., O’Neill, A., Ristenpart, T.: Deterministic encryption: definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_20
5. Bellare, M., Hoang, V.T.: Resisting randomness subversion: fast deterministic and hedged public-key encryption in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 627–656. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_21
6. 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
7. 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
8. Bellare, M., Tackmann, B.: Nonce-based cryptography: retaining security when randomness fails. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 729–757. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_28
9. Boldyreva, A., Fehr, S., O’Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_19
10. Boldyreva, A., Patton, C., Shrimpton, T.: Hedging public-key encryption in the real world. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 462–494. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_16
11. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: the auxiliary-input setting. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543–560. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_31
12. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: ACM CCS 2015, pp. 668–679. ACM Press (2015)
13. Checkoway, S., Fredrikson, M., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H.: On the practical exploitability of dual EC in TLS implementations. In: USENIX Security, vol. 1 (2014)
14. Fuller, B., O’Neill, A., Reyzin, L.: A unified approach to deterministic encryption: new constructions and a connection to computational entropy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_33
15. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: detection of widespread weak keys in network devices. In: USENIX Security Symposium, pp. 205–220 (2012)
16. Hoang, V.T., Katz, J., O’Neill, A., Zaheri, M.: Selective-opening security in the presence of randomness failures. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 278–306. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_10

17. Mironov, I., Pandey, O., Reingold, O., Segev, G.: Incremental deterministic public-key encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 628–644. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_37
18. Kaliski, B.: Public-Key Cryptography Standards (PKCS) # 1: RSA Cryptography Specifications Version 2.1, RFC 3447 (2003). <https://tools.ietf.org/html/rfc3447>
19. 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
20. Raghunathan, A., Segev, G., Vadhan, S.: Deterministic public-key encryption for adaptively chosen plaintext distributions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 93–110. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_6
21. Ristenpart, T., Yilek, S.: When good randomness goes bad: virtual machine reset vulnerabilities and hedging deployed cryptography. In: NDSS (2010)
22. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)
23. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science. ACM (2013)



Related Randomness Security for Public Key Encryption, Revisited

Takahiro Matsuda and Jacob C. N. Schuldt^(✉)

National Institute of Advanced Industrial Science and Technology (AIST),
Tokyo, Japan
{t-matsuda,jacob.schuldt}@aist.go.jp

Abstract. Motivated by the history of randomness failures in practical systems, Paterson, Schuldt, and Sibborn (PKC 2014) introduced the notion of related randomness security for public key encryption. In this paper, we firstly show an inherent limitation of this notion: if the family of related randomness functions is sufficiently rich to express the encryption function of the considered scheme, then security cannot be achieved. This suggests that achieving security for function families capable of expressing more complex operations, such as those used in random number generation, might be difficult. The current constructions of related randomness secure encryption in the standard model furthermore reflect this; full security is only achieved for function families with a convenient algebraic structure. We additionally revisit the seemingly optimal random oracle model construction by Paterson et al. and highlight its limitations.

To overcome this difficulty, we propose a new notion which we denote *related refreshable randomness* security. This notion captures a scenario in which an adversary has limited time to attack a system before new entropy is added. More specifically, the number of encryption queries with related randomness the adversary can make before the randomness is refreshed, is bounded, but the adversary is allowed to make an unbounded total number of queries. Furthermore, the adversary is allowed to influence how entropy is added to the system. In this setting, we construct an encryption scheme which remains secure in the standard model for arbitrary function families of size 2^p (where p is polynomial in the security parameter) that satisfy certain collision-resistant and output-unpredictability properties. This captures a rich class of functions, which includes, as a special case, circuits of polynomial size. Our scheme makes use of a new construction of a (bounded) related-key attack secure pseudorandom function, which in turn is based on a new flavor of the leftover hash lemma. These technical results might be of independent interest.

1 Introduction

Most cryptographic primitives are designed under the assumption that perfect uniform randomness is available. However, in practice, this is often not the case.

The design of random number generators (RNGs), which are used to generate the required randomness, is a complex and difficult task, and several examples of RNGs failing in practice are known [20, 23, 24, 26, 27]. The consequences of this might be fatal, and the examples of attacks made possible by randomness failures are many (e.g. see [12, 13, 29, 32, 39]). To make matters worse, some cryptographic designs are particularly fragile with respect to randomness failures. An example of this, is the DSA signature scheme [33], which allows the signing key to be recovered from two signatures on different messages constructed using the same randomness. This property enabled the compromise of the security mechanisms in the Sony Playstation 3 [12], the theft of Bitcoins from wallets managed on Android devices [16], and the recovery of TLS server signing keys from virtualized servers [39]. The latter example highlights an important aspect: even if the used RNG is not flawed by itself, randomness failures might still occur when the RNG is used in virtualized environments which enable virtual machines (including the state of the RNG) to be cloned or reset. Given the risk of randomness failures occurring in practical systems, it is prudent to design cryptographic primitives that provide resilience against these to the extent that this is possible. While it is possible to address this via generic derandomization for primitives like signature schemes¹, this is not the case for other primitives like public key encryption, which inherently relies on randomness for security.

1.1 The Related Randomness Setting

Motivated by the challenge of designing public key encryption secure under randomness failure, Paterson et al. [34] introduced the notion of *related randomness attack* (RRA) security. This notion allows the adversary to control the randomness used in the encryption scheme, but still requires that messages encrypted under an honestly generated public key remain hidden, given that certain restrictions are placed on the adversary's queries. More specifically, the RRA security game defines a set of initially well-distributed random values which are hidden to the adversary. Via an encryption oracle, the adversary will be able to request encryptions under public keys and on messages of his choice, using functions ϕ of these random values. The adversary will furthermore have access to a challenge oracle, which, given two messages, consistently returns the encryption of the first or the second message under an honestly generated public key; the task of the adversary is to guess which of the messages is encrypted. However, even for the challenge encryptions, the adversary can specify functions ϕ of the random values defined in the game, which will be used as randomness in the encryptions. The RRA model is inspired by the practical attacks illustrated by Ristenpart and Yilek [39], which exploits weaknesses of randomness generation in virtual machines, and furthermore captures as a special case the reset attacks by Yilek [43] in which encryptions using repeated random values are considered.

¹ Specifically, the folklore approach of generating any required randomness via a keyed PRF evaluated on the message to be signed, will work for any signature scheme. See also discussion of deterministic DSA in [36].

In [34], Paterson *et al.* showed several constructions of schemes secure in the RRA setting. Specifically, assuming the functions ϕ are drawn from a function family Φ of *output-unpredictable* and *collision-resistant* functions (which are also necessary conditions for achieving RRA security), the simple randomized-encrypt-with-hash (REwH) scheme by Bellare *et al.* [6] is shown to achieve RRA security in the random oracle model (however, as will be explained below, this construction still suffers from limitations inherent to the RRA model). Furthermore, in the standard model, a generic construction based on a Φ -related key attack secure pseudo-random function (RKA-PRF) [7] and any standard encryption scheme, is shown to yield a RRA-secure encryption for functions Φ . Using recent constructions of RKA-PRFs, e.g. [3], an encryption scheme RRA-secure for polynomial functions Φ can be obtained. Likewise, a generic construction based on a Φ -correlated input-secure (CIS) hash function [25], a standard PRF, and an encryption scheme, is shown to yield a RRA-secure encryption scheme for functions Φ , albeit in a weaker *honest-key* model. Furthermore, the only known standard model construction of a CIS hash function only provides selective security for polynomial functions Φ . In more recent work, Paterson *et al.* [35] showed a generic construction based on a reconstructive extractor and an encryption scheme, which yields security for *hard-to-invert* function families, but only in a selective security model in which the adversary is forced to commit to the functions used in the security game before seeing the public key. Furthermore, the concrete construction obtained in [35] only allows the adversary to maliciously modify the randomness used by his encryption oracle; the challenge oracle is required to use uniformly distributed randomness.

Hence, the best known construction achieving a reasonable level of security in the standard model, only obtains RRA-security for polynomial function families Φ . However, it seems unlikely that the randomness relations encountered in practice can be expressed with a function class with such convenient algebraic structure. While obtaining security for more complex function classes is clearly desirable, it is challenging to construct provably secure schemes for function families without an algebraic structure that can be exploited in the proof. This challenge is additionally reflected by the current state-of-the-art RKA-secure PRFs [1,3] which can only handle polynomial function families.

1.2 Our Results

First of all, we observe that if the function family Φ becomes sufficiently complex, RRA-security cannot be achieved for Φ . More precisely, if Φ is sufficiently rich to be able to express the encryption function of the scheme we are considering, a direct attack against the scheme in the RRA setting becomes possible. The attack is relatively simple, and is based on the ability of the adversary to derive the randomness used in his challenge encryption with the help of his encryption oracle. Assuming the encryption scheme satisfies ordinary IND-CPA security, the attack does not violate the properties required to make the RRA-security notion meaningful, which are the equality-respecting property, output unpredictability, and collision resistance. The details of this are given in Sect. 4. At first, this

might appear to contradict the results by Paterson *et al.* [34] regarding the REwH construction in the random oracle model. However, closer inspection reveals that the results from [34] implicitly assume that the functions Φ are *independent* of the random oracle, and hence, Φ will not be able to capture the encryption function of the REwH construction.

Considering the above, we revisit the security of the REwH construction in the random oracle model, and show that if additional restrictions are placed on the adversary, security can be obtained. More specifically, if the adversary respects *indirect H-query uniqueness*, which is a property requiring that the random oracle queries caused by the adversary's encryption and challenge queries are all distinct, then RRA-security is obtained, even for function families Φ which are dependent on the random oracle, as long as the functions in Φ are output-unpredictable. The details of this are in Sect. 5. Our results are reminiscent of the results by Albrecht *et al.* [5] regarding cipher-dependent related-key attacks in the ideal cipher model.

However, the indirect H-query uniqueness property is an artificial restriction to place on the adversary, and the above result seems unsatisfactory. Furthermore, the above negative result suggests that, achieving security for function families that reflect more complex operations, which might be used in random number generators, could be difficult.

Hence, to overcome this difficulty, we propose a new notion which we denote *related refreshable randomness* security. In this notion, we bound the number of queries an adversary can make before new entropy is added to the system, but allow an unbounded total number of queries. We refer to the periods between refreshes as *epochs*. Furthermore, we allow the adversary to maliciously influence how entropy is added between epochs. This is implemented by giving the adversary access to a *refresh* oracle through which the adversary can submit update functions ψ . These functions take as input the current random values and a update seed chosen uniformly at random, and output new random values which will be used in the security game. For this update mechanism to be meaningful, we restrict the functions ψ to come from a function family Ψ in which all functions have the property, that their output has a certain level of min-entropy conditioned on the random values being updated (i.e. it is required that a certain amount of the entropy contained in the update seed, will be carried over to the output of the update function). With this requirement in place, we consider adversaries who makes at most n queries to their encryption and challenge oracles, before querying the refresh oracle. The details of the security model are given in Sect. 3.

The related refreshable randomness setting models the arguably realistic scenario in which an attacker only has limited time to interact with a system that is in a state where no new entropy is being added to the system, and highly correlated randomness values are used for encryption. This furthermore resembles the observations made in [39] regarding virtual machine reset attacks; the attacks were only possible in a relatively short window after the virtual machine

was reset, before sufficient entropy was gathered from the network, clock synchronization, and similar sources.

The related refreshable randomness setting furthermore allows us to obtain positive results in the standard model. Specifically, we construct a scheme which is secure in the related refreshable randomness setting for *arbitrary* function families Φ and Ψ satisfying certain output unpredictability and collision resistance properties. We do, however, require the size of the function families to be bounded by an a priori known bound of the form 2^p , where p is a polynomial in the security parameter. This allows us to capture a rich class of functions which include, for example, the set of all functions that can be described by circuits of polynomial size. Our construction is based on the same high-level approach as taken in [34, 43], and combines a standard encryption scheme with a PRF (see below for the details). However, by relying on a new construction of a (bounded) RKA-secure PRF, we are able to prove security in the related refreshable randomness setting for much more interesting function classes than considered in [34, 43]. Notably, in contrast to our scheme, the scheme from [43] is only reset secure ($\Phi = \{\text{id}\}$), and the scheme from [34] only achieves selective security for polynomial functions Φ , and hence cannot capture non-algebraic functions such as bit-flipping and bit-fixing, which are highly relevant to randomness failures in practice. The full details can be found in Sect. 7.

1.3 Technique

As highlighted above, the main tool we use to obtain our standard model encryption scheme secure in the related refreshable randomness setting, is a new construction of a RKA-secure PRF. We consider this construction to be our main technical contribution. As an intermediate step, we construct (a variant of) a CIS hash function. This type of hash function was originally introduced by Goyal et al. [25]. While different security notions for CIS hash functions were introduced in [25], the one we will be concerned with here, is pseudo-randomness. This notion requires that, for a hash function $H : D \rightarrow R$ and a randomly chosen value $x \in D$, an adversary cannot distinguish an oracle which returns $H(\phi(x))$ for adversarially chosen functions ϕ , from an oracle that returns a random value from R . In [25], a construction obtaining selective security for a polynomial function family Φ was shown. However, we show that by bounding the number of queries to the adversary's oracle, we can obtain a construction achieving security for a class Φ of arbitrary functions that are output-unpredictable and collision-resistant, where the size of Φ is bounded a priori. This construction is in turn based on a new flavor of the leftover hash lemma [28] for correlated inputs that might depend on the description of the hash function. Then, by applying this CIS hash function H to the key of a standard PRF prf , we obtain a new PRF $\text{prf}'(k, x) := \text{prf}(H(k), x)$ that provides RKA security, as long as the adversary will only query a bounded number of different key derivation functions. However, the adversary is allowed to obtain an unbounded number of evaluation results under the derived keys. The detailed proofs of security can be found in Sect. 6.

Finally, we obtain a standard model encryption scheme in the related refreshable randomness setting via the same transformation used in [34, 43]: to encrypt a message m under public key pk using randomness r , we compute $\text{Enc}(pk, m; r')$, where $r' = \text{prf}'(r, pk \| m)$. The security properties of the constructed PRF prf' allows us to prove security via a hybrid argument with respect to the epochs. Note, however, that the parameters of the scheme will grow linearly in the in the number of queries an adversary is allowed to make in each epoch, as a description of H must be included. See Sect. 7 for the details.

Our construction of a RKA-secure PRF, CIS hash function, and our new flavor of the leftover hash lemma, might find applications outside of related randomness security, and hence, might be of independent interest. For example, by directly applying our RKA-secure PRF in combination with the framework of Bellare et al. [8], we can obtain RKA-secure signatures, public key encryption, and identity-based encryption for function families of size bounded by 2^p and with the appropriate collision-resistant and output-unpredictability properties. Security is only guaranteed for a bounded number of related key derivation queries, but the total number of allowed signatures, decryption queries, and key queries for identities, respectively, is unbounded. Furthermore, it is not hard to see that our PRF construction only requires the PRF keys to have high min-entropy (as opposed to being uniformly distributed), as long as the considered function family remains collision-resistant and output-unpredictable. This indicates that the construction can additionally tolerate leakage, and we conjecture that bounded leakage and tampering security as defined by Damgård et al. [18, 19], can be achieved.

1.4 Related Work

A number of works in the literature have considered security of various cryptographic primitives in the event of randomness failures. In the symmetric key setting, Rogaway and Shrimpton [40] considered the security of authenticated encryption in the case nonces are repeated, and Katz and Kamara [31] considered chosen randomness attacks which allows the adversary to freely choose the randomness, except for the challenge encryption. In the public key setting, Bellare et al. [6] considered *hedged* encryption, which remains secure as long as the joint distribution of messages and randomness contains sufficient entropy. Note that the security notion formalized for hedged encryption in [6], security against chosen distribution attacks (CDA), is incomparable to RRA-security which does not rely on message entropy. Furthermore, whereas RRA-security allows the adversary to obtain encryptions under maliciously chosen public keys using randomness related to the randomness of the challenge encryptions, there is no equivalent in CDA-security, and CDA-security does not allow messages and randomness to depend on the public key. Additionally, the known standard model constructions of CDA-secure schemes are only shown secure for block sources which require each message/randomness pair to have high min-entropy conditioned on all previous pairs, whereas the standard model RRA-secure schemes from [34, 35] and the schemes in this paper do not have similar restrictions. Vergnaud and

Xaio [42] slightly strengthened the CDA-security considered in [6] by allowing the message/randomness pair to partly depend on the public key. Yilek [43] considered reset attacks in which encryptions with repeated randomness values might occur, and gave a construction based on a standard encryption scheme and a PRF. This is a special case of the RRA-setting. Bellare and Tackmann [11] introduced the notion of nonce-based public key encryption, and achieved a number of strong results. However, the constructions assume a stateful scheme, and is hence not applicable to a number of scenarios in which we are interested in related randomness security, e.g. virtual machine resets. Extending [6] and [11], Hoang et al. [30] considered security of hedged encryption and nonce-based public key encryption under selective opening attack.

Appelbaum and Widder [4] constructed a (bounded) RKA-secure PRF for additions, while Abdalla *et al.* [2] constructed a RKA-secure PRF for XORs from multilinear maps. In contrast, our PRF construction achieves security for arbitrary functions satisfying collision resistance and unpredictability, for a bounded number of related keys. We stress, however, that the bound is only on the number of keys, and that our construction remains secure for an unbounded number of PRF evaluations.

2 Preliminaries

2.1 Notation and Basic Notions

Throughout the paper we will use $\lambda \in \mathbb{N}$ to denote the security parameter, which will sometimes be written in its unary representation, 1^λ . Furthermore, we sometimes suppress the dependency on λ , when λ is clear from the context. We denote by $y \leftarrow x$ the assignment of y to x , and by $s \leftarrow_{\$} S$ we denote the selection of an element s uniformly at random from the set S . The notation $[n]$ represents the set $\{1, 2, \dots, n\}$. For an algorithm A , we denote by $y \leftarrow A(x; r)$ that A is run with input x and random coins r , and that the output is assigned to y . For a vector $\mathbf{x} = (x_1, x_2, \dots)$, we denote by $A(\mathbf{x})$ the vector $(A(x_1), A(x_2), \dots)$. For a random variable X defined over a set S , we denote by $H_\infty(X)$ the min-entropy of X (i.e. $H_\infty(X) = -\log_2 \max_{x \in S} \Pr[X = x]$), and for two random variables X and Y defined over the same set S , we denote the statistical distance between X and Y as $\Delta[X, Y]$ (i.e. $\Delta[X, Y] = \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$).

2.2 t -wise Independent Hash Functions

One of the basic building blocks of our construction is t -wise independent hash functions, which we define here. We furthermore recall a tail inequality for t -wise independent variables due to Bellare and Rompel [10], which we will make use of in our proofs of security.

Definition 1 (t -wise independent hash function family). Let $\mathcal{H} = \{H \mid H : D \rightarrow R\}$ be a family of hash functions. \mathcal{H} is said to be a t -wise independent hash function family, if for all mutually distinct $x_1, \dots, x_t \in D$ and all $y_1, \dots, y_t \in R$, it holds that $\Pr_{H \leftarrow_{\$} \mathcal{H}}[H(x_1) = y_1 \wedge \dots \wedge H(x_t) = y_t] = \frac{1}{|R|^t}$.

Theorem 1 (Tail inequality[10]). *Let t be an even integer larger than 8, and let X_1, \dots, X_n be t -wise independent variables² assuming values in the interval $[0, 1]$. Furthermore, let $X = X_1 + \dots + X_n$, $\mu = \mathbb{E}[X]$, and $\epsilon < 1$. Then*

$$\Pr[|X - \mu| \geq \epsilon\mu] \leq \left(\frac{t}{\epsilon^2\mu}\right)^{t/2}.$$

2.3 Output Unpredictability and Collision Resistance

We will consider function families which are output-unpredictable and collision-resistant. These properties were originally defined by Bellare and Kohno [9] in the context of RKA security, and used by Paterson et al. [34] in the context of RRA security. The following definitions are slightly simplified compared to [9, 34].

Definition 2 (Output unpredictability). *Let $\Phi = \{\phi : D \rightarrow R\}$ be a family of functions with domain $D = D_\lambda$ and range $R = R_\lambda$. The output unpredictability of Φ is defined as $\text{UP}^\Phi(\lambda) = \max_{\phi \in \Phi, y \in R} \Pr[x \leftarrow_{\$} D : \phi(x) = y]$. When $\text{UP}^\Phi(\lambda) < \epsilon$ for a negligible function $\epsilon = \epsilon(\lambda)$, we simply say that Φ is output-unpredictable.*

Definition 3 (Collision resistance). *Let $\Phi = \{\phi : D \rightarrow R\}$ be a family of functions with domain $D = D_\lambda$ and range $R = R_\lambda$. The collision resistance of Φ is defined as $\text{CR}^\Phi(\lambda) = \max_{\phi_1, \phi_2 \in \Phi, \phi_1 \neq \phi_2} \Pr[x \leftarrow_{\$} D : \phi_1(x) = \phi_2(x)]$. When $\text{CR}^\Phi(\lambda) < \epsilon$ for a negligible function $\epsilon = \epsilon(\lambda)$, we simply say that Φ is collision-resistant.*

2.4 Pseudorandom Function

A pseudorandom function F is given by the following three algorithms: $F.\text{Setup}(1^\lambda)$ which on input the security parameter, returns public parameters par (required to describe a domain D and a range R); $F.\text{KeyGen}(par)$ which, on input par , returns a key k ; and $F.\text{Eval}(par, k, x)$ which, on input par , key k , and input $x \in D$, returns an output value $y \in R$. For notational convenience, we will sometimes suppress par from the input.

We will consider the security of a pseudorandom function in a multi-key setting. This is for convenience only; by a standard hybrid argument, it is easily seen that this definition is equivalent to a definition considering a single key, as also shown by Bellare et al. [15]. We define security via the security game shown in Fig. 1.

Definition 4. *Let the advantage of an adversary \mathcal{A} playing the security game in Fig. 1 with respect to a pseudorandom function $F = (\text{Setup}, \text{KeyGen}, \text{Eval})$ be defined as $\text{Adv}_{F, \mathcal{A}}^{\text{PRF}}(\lambda) = 2 \left| \Pr[\text{PRF}_{\mathcal{A}}^F(\lambda) \Rightarrow 1] - \frac{1}{2} \right|$. F is said to be secure if for all PPT adversaries \mathcal{A} , $\text{Adv}_{F, \mathcal{A}}^{\text{PRF}}(\lambda)$ is negligible in the security parameter λ .*

² Random variables X_1, \dots, X_n are t -wise independent if for all distinct indices $i_1, \dots, i_t \in [n]$ and all x_1, \dots, x_t , $\Pr[\bigwedge_{j \in [t]} (X_{i_j} = x_j)] = \prod_{j \in [t]} \Pr[X_{i_j} = x_j]$ holds.

$\begin{array}{l} \text{PRF}_{\mathcal{A}}^{\mathcal{F}}(\lambda): \\ \text{par} \leftarrow \text{F.Setup}(1^\lambda) \\ b \leftarrow_{\mathcal{S}} \{0, 1\} \\ \mathcal{F} \leftarrow \emptyset \\ \text{ctr} \leftarrow 0 \\ b' \leftarrow \mathcal{A}^{\text{EVAL}, \text{NEW}}(\text{par}) \\ \text{return } (b = b') \end{array}$	$\begin{array}{l} \text{proc. EVAL}(i, x): \\ \text{if } i > \text{ctr, return } \perp \\ \text{if } b = 1 \\ \quad y \leftarrow \text{F.Eval}(k_i, x) \\ \text{else} \\ \quad \text{if } \mathcal{F}[i, x] = \perp, \mathcal{F}[i, x] \leftarrow_{\mathcal{S}} R \\ \quad y \leftarrow \mathcal{F}[i, x] \\ \text{return } y \end{array}$	$\begin{array}{l} \text{proc. NEW:} \\ \text{ctr} \leftarrow \text{ctr} + 1 \\ k_{\text{ctr}} \leftarrow \text{F.KeyGen}(\text{par}) \\ \text{return } \text{ctr} \end{array}$
--	--	--

Fig. 1. Game defining security of a pseudorandom function.

2.5 Public Key Encryption

A public key encryption (PKE) scheme PKE is defined by the following four algorithms: PKE.Setup(1^λ) which on input the security parameter, returns public parameters par ; PKE.KeyGen(par) which on input par , returns a public/private key pair (pk, sk) ; PKE.Enc(par, pk, m) which on input par , public key pk , and message m , returns an encryption c of m under pk ; and PKE.Dec(par, sk, c) which on input par , private key sk , and ciphertext c , returns either a message m or the error symbol \perp . For convenience, we often suppress par from the input.

We require that a PKE scheme satisfies *perfect correctness*, that is, for all λ , all $\text{par} \leftarrow \text{PKE.Setup}(1^\lambda)$, all $(pk, sk) \leftarrow \text{PKE.KeyGen}(\text{par})$, and all $m \in \mathcal{M}(pk)$, it holds that $\text{PKE.Dec}(sk, \text{PKE.Enc}(pk, m)) = m$. Security of a PKE scheme is defined via the game shown in Fig. 2.

$\begin{array}{l} \text{IND-CCA}_{\mathcal{A}}^{\text{PKE}}(\lambda): \\ \text{par} \leftarrow \text{PKE.Setup}(1^\lambda) \\ (pk^*, sk^*) \leftarrow \text{PKE.KeyGen}(\text{par}) \\ b \leftarrow_{\mathcal{S}} \{0, 1\} \\ \mathcal{C} \leftarrow \emptyset \\ b' \leftarrow \mathcal{A}^{\text{LR}, \text{DEC}}(\text{par}, pk^*) \\ \text{return } (b = b') \end{array}$	$\begin{array}{l} \text{proc. LR}(m_0, m_1): \\ c \leftarrow \text{PKE.Enc}(pk^*, m_b) \\ \mathcal{C} \leftarrow \mathcal{C} \cup \{c\} \\ \text{return } c \end{array}$	$\begin{array}{l} \text{proc. DEC}(c): \\ \text{if } c \in \mathcal{C}, \\ \quad \text{return } \perp \\ \text{return PKE.Dec}(sk^*, c) \end{array}$
---	--	--

Fig. 2. Game defining IND-CCA security for a PKE scheme.

Definition 5 (IND-CCA security). Let the advantage of an adversary \mathcal{A} playing the IND-CCA game with respect to a PKE scheme $\text{PKE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$, be defined as: $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA}}(\lambda) = 2 \left| \Pr[\text{IND-CCA}_{\mathcal{A}}^{\text{PKE}}(\lambda) \Rightarrow 1] - \frac{1}{2} \right|$. A scheme PKE is said to be IND-CCA secure, if for all PPT adversaries \mathcal{A} , $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-CCA}}(\lambda)$ is negligible in the security parameter λ .

3 Related Refreshable Randomness Security

We will firstly define our new notion of related refreshable randomness security. This builds upon the RRA-security notion defined by Paterson *et al.* [34], but

<pre> IND-RRR-CCA_A^{PKE}(λ): par ← PKE.Setup(1^λ) (pk*, sk*) ← PKE.KeyGen(par) b ←_{\$} {0, 1}; r ←_{\$} \mathcal{R} C ← ∅ b' ← A^{REFRESH, LR, ENC, DEC}(par, pk*) return (b = b') proc. REFRESH(ψ): s ←_{\$} \mathcal{S} r ← ψ(r, s) </pre>	<pre> proc. LR(m₀, m₁, φ): c ← PKE.Enc(pk*, m_b; φ(r)) C ← C ∪ {c} return c proc. ENC(pk, m, φ): return PKE.Enc(pk, m; φ(r)) proc. DEC(c): if c ∈ C, return ⊥ else return PKE.Dec(sk*, c) </pre>
---	--

Fig. 3. Game defining indistinguishability under related refreshable randomness and chosen ciphertext attacks (IND-RRR-CCA).

models a setting in which the adversary has limited time to attack a system before new entropy is added to the system. As in the original RRA security game, we consider a polynomial number of randomness values r_i , and give the adversary access to an encryption oracle ENC which returns encryptions under public keys and messages of the adversary’s choice, and a challenge left-or-right oracle LR, which consistently returns the encryption of either the first or the second message of two submitted messages m_0, m_1 , under an honestly generated challenge public key pk^* . However, for both oracles, the adversary can not only specify which random value r_i to be used, but also a function ϕ which will be applied to r_i before it is used (i.e. the used randomness will be $\phi(r_i)$). We furthermore introduce an additional oracle, REFRESH, which allows the adversary to submit a function ψ that will be used to refresh the random values r_i . The function ψ takes two inputs: the randomness r_i which is to be refreshed, and a seed s . Here, the seed s will be drawn uniformly at random from a seed space \mathcal{S} , and $\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}$, where \mathcal{R} is the randomness space of the encryption scheme. The full security game is defined in Fig. 3. Note that while the security game shown in Fig. 3 is only defined for a single random value r , this is equivalent to a model defined for a polynomial number of randomness values r_i (see the full version of the paper).

Note that, by itself, introducing the REFRESH oracle does not achieve the intended goal, as the adversary is not forced to query REFRESH. However, we will consider a class of adversaries which make at most n ENC and LR queries between each call to REFRESH (but is allowed to make an unrestricted number of queries to DEC). We will furthermore parameterize this class of adversaries by function families Φ and Ψ from which an adversary is allowed to choose related randomness functions ϕ and refresh functions ψ , respectively, and will refer to adversaries in this class as (n, Φ, Ψ) -restricted adversaries³. In the following

³ Note that since the functions ϕ and ψ will depend on the security parameter λ , Φ and Ψ are technically ensembles of function families indexed by λ . However, for notational convenience, we suppress λ .

definitions and proofs, we need to refer to the execution of an adversary in between two calls to REFRESH, which we will denote an *epoch*⁴.

As in the case of RRA-security, since the defined oracles let the adversary control the randomness in the challenge encryptions, a few natural restrictions must be placed on the adversary’s queries to obtain a meaningful definition of security. Specifically, we require that an adversary is *equality respecting*. This is reminiscent of the restriction defined for deterministic encryption schemes [38].

Definition 6 (Equality-respecting adversary). *Consider a (n, Φ, Ψ) -restricted adversary \mathcal{A} playing the IND-RRR-CCA security game for security parameter λ . Let $\mathcal{M}_{\text{Enc}}^{\phi, \delta}$ denote the set of messages \mathcal{A} submits to the ENC oracle for challenge public key pk^* and related randomness function $\phi \in \Phi$ in refresh epoch δ . Furthermore, let $(m_0^{\phi, \delta, 1}, m_1^{\phi, \delta, 1}), \dots, (m_0^{\phi, \delta, q_\phi}, m_1^{\phi, \delta, q_\phi})$ denote the messages \mathcal{A} submits to the LR oracle for function ϕ in refresh epoch δ . Then \mathcal{A} is said to be equality-respecting if, for all $\phi \in \Phi$, for all refresh epochs δ , and for all $i, j \in [q_\phi]$ s.t. $i \neq j$,*

$$m_0^{\phi, \delta, i} = m_0^{\phi, \delta, j} \Leftrightarrow m_1^{\phi, \delta, i} = m_1^{\phi, \delta, j} \quad \text{and} \quad m_0^{\phi, \delta, i}, m_1^{\phi, \delta, j} \notin \mathcal{M}_{\text{Enc}}^{\phi, \delta}.$$

With this definition in place, we are ready to define our notion of security.

Definition 7 (IND-RRR-CCA Security). *Let the advantage of an adversary \mathcal{A} playing the IND-RRR-CCA game with respect to a public key encryption scheme $\text{PKE} = (\text{PKE.Setup}, \text{PKE.KeyGen}, \text{PKE.Enc}, \text{PKE.Dec})$, be defined as:*

$$\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-RRR-CCA}}(\lambda) = 2 \left| \Pr[\text{IND-RRR-CCA}_{\text{PKE}}^{\mathcal{A}}(\lambda) \Rightarrow 1] - \frac{1}{2} \right|.$$

A scheme PKE is said to be (n, Ψ, Φ) -IND-RRR-CCA secure, if for all PPT (n, Φ, Ψ) -restricted and equality-respecting adversaries \mathcal{A} , $\text{Adv}_{\text{PKE}, \mathcal{A}}^{\text{IND-RRR-CCA}}(\lambda)$ is negligible in the security parameter λ .

The original RRA-security notion defined in [34] can be obtained from the above definition by not allowing the adversary access to the REFRESH oracle (i.e. considering only the first refresh epoch) and considering an unbounded value n . In this case, Ψ is irrelevant, and we simply write Φ -IND-RR-CCA security to denote this security notion⁵. Lastly, note that ordinary IND-CCA security can be obtained from the above definition by setting $n = 1$, $\Phi = \{\text{id}\}$, and $\Psi = \{\text{id}_2 : (r, s) \rightarrow s\}$ (assuming $\mathcal{S} = \mathcal{R}$).

3.1 Basic Function Family Restrictions

Unsurprisingly, related randomness security for all function families Φ and Ψ is not achievable. This is similar to the security notions for related key attacks (e.g.

⁴ Hence, if an adversary \mathcal{A} submits q queries to REFRESH in total, then the execution of \mathcal{A} has $q + 1$ epochs.

⁵ Note that in [34], the notation Φ -RRA-CCA was used for this security notion.

see [7]), which must restrict the class of related-key deriving functions that can be applied to the private key, in order to become achievable. We will now establish basic restriction which must be placed on Φ and Ψ to make the IND-RRR-CCA notion defined above achievable.

The two basic properties we consider are output-unpredictability and collision-resistance of the functions in Φ . However, as the IND-RRR-CCA security game allows the adversary to update the challenge randomness using the functions Ψ , we will consider output-unpredictability and collision-resistance of Φ with respect to Ψ i.e. the functions in Φ must be output-unpredictable and collision-resistant, even when the input is modified using functions from Ψ . In the following definitions we will use the notation $\overline{\Psi}^q$ to denote the q -closure of the functions in Ψ . More specifically, each function $\overline{\psi} \in \overline{\Psi}^q$ corresponds to q updates of a randomness value r using q functions $\psi_1, \dots, \psi_q \in \Psi$, and will take as input r and q seeds $\overline{s} = (s_1, \dots, s_q)$ and return $\overline{\psi}(r, \overline{s}) = \psi_q(\psi_{q-1}(\dots \psi_1(r, s_1) \dots, s_{q-1}), s_q)$. As the seeds s_i are elements of \mathcal{S} , we have that $\overline{\psi} : \mathcal{R} \times \mathcal{S}^q \rightarrow \mathcal{R}$.

Definition 8 (Output-unpredictability of Φ w.r.t. Ψ). Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$. For a positive integer q , the q -output-unpredictability of Φ with respect to Ψ is defined as $\text{UP}_q^{\Phi, \Psi}(\lambda) = \max_{\phi \in \Phi, \overline{\psi} \in \overline{\Psi}^q, y \in \mathcal{R}} \Pr [r \leftarrow_{\S} \mathcal{R}, \overline{s} \leftarrow_{\S} \mathcal{S}^q : \phi(\overline{\psi}(r, \overline{s})) = y]$.

Definition 9 (Collision-resistance of Φ w.r.t. Ψ). Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$. The collision-resistance of Φ with respect to Ψ is defined as

$$\text{CR}_q^{\Phi, \Psi}(\lambda) = \max_{\substack{\phi_1, \phi_2 \in \Phi, \overline{\psi} \in \overline{\Psi}^q \\ \phi_1 \neq \phi_2}} \Pr [r \leftarrow_{\S} \mathcal{R}, \overline{s} \leftarrow_{\S} \mathcal{S}^q : \phi_1(\overline{\psi}(r, \overline{s})) = \phi_2(\overline{\psi}(r, \overline{s}))].$$

In [34], Paterson *et al.* showed that to achieve Φ -IND-RR-CCA security, Φ is required to satisfy standard output-unpredictability and collision-resistance. Likewise, in the IND-RRR-CCA setting, we can show that Φ must be output-unpredictability and collision-resistance w.r.t. Ψ for security to be achievable.

Theorem 2 (Necessity of Φ output-unpredictability w.r.t. Ψ). Let $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be a function family, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$, and suppose that there exist a positive integer $q = \text{poly}(\lambda)$ and a non-negligible function $\epsilon = \epsilon(\lambda)$ such that $\text{UP}_q^{\Phi, \Psi}(\lambda) > \epsilon$. Then no PKE scheme can be (n, Ψ, Φ) -IND-RRR-CCA secure for $n \geq 1$.

Proof (Sketch). The proof is straightforward. Let $\phi \in \Phi$, $\overline{\psi} \in \overline{\Psi}^q$, and $y \in \mathcal{R}$ such that $\Pr[r \leftarrow_{\S} \mathcal{R}, \overline{s} \leftarrow_{\S} \mathcal{S}^q : \phi(\overline{\psi}(r, \overline{s})) = y] > \epsilon$. These are guaranteed to exist since $\text{UP}_q^{\Phi, \Psi}(\lambda) > \epsilon$. Consider an adversary \mathcal{A} submitting functions corresponding to $\overline{\psi}$ as REFRESH queries, and (ϕ, m_0, m_1) in a following LR query. Let c be the challenge ciphertext \mathcal{A} receives. Now, let \mathcal{A} check whether $c = \text{Enc}(pk^*, m_b; y)$ for $b = 0$ and $b = 1$, and if so, return b . Otherwise, let \mathcal{A}

return a random bit. It easily follows that such \mathcal{A} has advantage at least ϵ which is assumed to be non-negligible, and hence the considered PKE scheme cannot be secure. □(Theorem 2)

Theorem 3 (Necessity of Φ collision-resistance w.r.t. Ψ). *Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$. Suppose that there exist a positive integer $q = \text{poly}(\lambda)$ and a non-negligible function $\epsilon = \epsilon(\lambda)$ such that $\text{CR}_q^{\Phi, \Psi}(\lambda) > \epsilon$. Then no PKE scheme can be (n, Ψ, Φ) -IND-RRR-CCA secure for $n \geq 2$.*

The proof of this theorem is similar to the proof of Theorem 2 and is omitted.

Note that, without further assumptions on Ψ , queries to the REFRESH oracle is not guaranteed to change the random value r used to respond to ENC and LR queries. In particular, if $\Psi = \{\text{id}_1 : (r, s) \rightarrow r\}$, the original value of r will be used in every refresh epoch, which essentially corresponds to removing the bound n on the number of ENC and LR queries. However, it is relatively easy to see that security cannot be achieved in this case⁶. Furthermore, the very idea behind introducing the IND-RRR-CCA security notion is to show that a guarantee of new entropy is being added to the system with certain intervals, can be leveraged to provide stronger security properties. Hence, we will consider a function class Ψ for which the output $r' \leftarrow \psi(r, s)$ of all update functions $\psi \in \Psi$ is required to depend on the seed s , or more specifically, that $\psi(r, s)$ will have a certain level of conditional min-entropy given r . We introduce this requirement implicitly via the following slightly stronger notions of output-unpredictability and collision-resistance of Φ w.r.t. Ψ . These notions require that the functions in Φ remain output-unpredictable and collision-resistant on input $\psi(r', s)$, $\psi \in \Psi$, for a randomly chosen seed s and any value r' , as opposed to a value of r' obtained by choosing the initial r at random and then modifying this using a chain of update functions $\bar{\psi} \in \bar{\Psi}^q$ and corresponding seeds $\bar{s} \in \bar{\mathcal{S}}^q$. We refer to these notions as *seed-induced* output-unpredictability and collision-resistance.

Definition 10 (Seed-induced output-unpredictability of Φ w.r.t. Ψ). *Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$. The seed-induced output-unpredictability of Φ with respect to Ψ is defined as*

$$\text{SUP}^{\Phi, \Psi}(\lambda) = \max_{\phi \in \Phi, \psi \in \Psi, r, y \in \mathcal{R}} \Pr [s \leftarrow_{\S} \mathcal{S} : \phi(\psi(r, s)) = y].$$

Definition 11 (Seed-induced collision-resistance of Φ w.r.t. Ψ). *Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{S} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$ and $\mathcal{S} = \mathcal{S}_\lambda$. The seed-induced collision-resistance of Φ with respect to Ψ is defined as*

⁶ In particular, the above definition of an equality-respecting adversary will allow the messages m_0, m_1 and the function ϕ from a LR query in one refresh epoch, to be submitted to the ENC oracle in combination with pk^* in a different refresh epoch, which trivially allows the adversary to break security.

$$\text{sCR}^{\Phi, \Psi}(\lambda) = \max_{\substack{\phi_1, \phi_2 \in \Phi, \psi \in \Psi, r \in \mathcal{R} \\ \phi_1 \neq \phi_2}} \Pr [s \leftarrow_{\S} \mathcal{S} : \phi_1(\psi(r, s)) = \phi_2(\psi(r, s))].$$

4 Restrictions on the Complexity of Function Families

We will now turn our attention to function families which satisfy the basic output-unpredictability and collision-resistant properties, but for which security nevertheless cannot be achieved.

More specifically, when Φ and Ψ become rich enough to express the encryption function itself of a scheme, a direct attack against the scheme becomes possible. This is reminiscent of the results by Albrecht et al. [5] regarding cipher-dependent related-key attacks in the ideal cipher model. The attack is based on the ability of an adversary to force the challenge encryption to be constructed using a value which can be obtain through the ENC and LR oracles available to the adversary. This is captured by the following theorem.

Theorem 4. *Let $\text{PKE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a public key encryption scheme, and let Φ be a family of functions such that $\text{id} \in \Phi$ and $f(\text{Enc}(pk, m, \cdot)) \in \Phi$ for some public key pk , message m , and a mapping function $f : \mathcal{C} \rightarrow \mathcal{R}$, where \mathcal{C} and \mathcal{R} are the ciphertext space and randomness space of PKE, respectively. Then PKE cannot be (n, Ψ, Φ) -IND-RRR-CCA secure for any $n \geq 2$ and any function family Ψ .*

Proof. The proof is straightforward. Since it is assumed that $f(\text{Enc}(pk, m; \cdot)) \in \Phi_\lambda$, an adversary would be able to submit $\phi(\cdot) = f(\text{Enc}(pk, m; \cdot))$ and two distinct messages, m_0 and m_1 , in a LR query to obtain the challenge encryption $c^* = \text{Enc}(pk^*, m_b; f(\text{Enc}(pk, m; r)))$, where pk^* is the challenge public key, b is the challenge bit, and r is the random value chosen in the IND-RRR-CCA game. Then, by submitting (pk, m, id) to his encryption oracle ENC, the adversary will obtain $c_r = \text{Enc}(pk, m; r)$ and can compute $\tilde{r} = f(c_r)$. Finally, the adversary can compute $c_0 = \text{Enc}(pk^*, m_0; \tilde{r})$ and $c_1 = \text{Enc}(pk^*, m_1; \tilde{r})$, and by testing whether $c_0 = c^*$ or $c_1 = c^*$, he will learn the challenge bit b . □(Theorem 4)

Note that the only functions required in the above attack, are $f(\text{Enc}(pk, m, \cdot))$ and $\text{id}(\cdot)$. These functions are easily seen to be output-unpredictable assuming the underlying encryption scheme in the construction is IND-CPA secure, and that an appropriate mapping function f is chosen. They can likewise be seen to be collision-resistant under the same assumptions. Furthermore, it should be noted that the above theorem does not require the REFRESH oracle to be queried, and hence is also true for the IND-RR-CCA notion defined in [34].

While the above theorem holds for all encryption schemes in general, stronger results might hold for concrete schemes. In particular, even if $f(\text{Enc}(pk, m; \cdot)) \notin \Phi$, the structure of a concrete scheme might still allow an adversary to mount a similar attack to the above based on multiple queries to his LR and ENC oracles, for carefully selected functions. However, the IND-RRR-CCA security notion bounds the information an adversary can extract before the randomness is refreshed,

which will allow us to construct a generic conversion of a PKE scheme achieving IND-RR-CCA security for relatively large and complex function classes Φ and Ψ . Interestingly, the above theorem furthermore illustrates some of the limitations of the building blocks used in [34] to achieve related randomness security; see the full version of the paper for a brief discussion of this.

5 On the IND-RR-CCA Security of REwH in the Random Oracle Model

In this section, we will revisit the IND-RR-CCA security of the REwH (Randomized-Encrypt-with-Hash) scheme in the random oracle model.

The REwH scheme was introduced by Bellare *et al.* [6] to hedge against randomness failures, and was furthermore studied by Ristenpart and Yilek [39] in the context of virtual machine reset attacks. The basic idea of the scheme is to modify the encryption function of an existing encryption scheme to use randomness derived by hashing all the inputs to the encryption algorithm: the public key, the message, and the randomness. Assuming the hash function is a random oracle, the scheme will remain secure (in the sense of the security of the underlying encryption scheme), as long as this triple of inputs remains unpredictable to the adversary. The scheme is shown in Fig. 4.

$\begin{array}{l} \text{Alg. REwH.KeyGen}(1^\lambda): \\ H \leftarrow_{\mathcal{S}} \mathcal{H} \\ (pk, sk) \leftarrow \text{PKE.KeyGen}(1^\lambda) \\ pk' \leftarrow (pk, H) \\ \text{return } (pk', sk) \end{array}$	$\begin{array}{l} \text{Alg. REwH.Enc}(pk', m): \\ r \leftarrow_{\mathcal{S}} \mathcal{R} \\ \tilde{r} \leftarrow H(pk \ m \ r) \\ c \leftarrow \text{PKE.Enc}(pk, m; \tilde{r}) \\ \text{return } c \end{array}$	$\begin{array}{l} \text{Alg. REwH.Dec}(sk, c): \\ m \leftarrow \text{PKE.Dec}(sk, m) \\ \text{return } m \end{array}$
--	---	---

Fig. 4. Scheme REwH constructed from a PKE scheme PKE and a hash family \mathcal{H} .

In [34], Paterson *et al.* showed that this scheme is additionally Φ -IND-RR-ATK secure assuming the underlying encryption scheme is IND-ATK secure, where ATK is either CPA or CCA, and Φ is both output-unpredictable and collision-resistant. Considering the impossibility result in the previous section, this might initially appear somewhat surprising. However, as already mentioned, the results in [34] implicitly assume that the functions in Φ are *independent* of the used random oracle i.e. the functions in Φ cannot capture the encryption function $\text{Enc}(pk, m; r) = \text{Enc}'(pk, m; H(pk, m, r))$ of the REwH construction, where Enc' is the encryption function of the underlying encryption scheme.

In this section, we will consider Φ which might depend on the random oracle, i.e. we will assume that functions in Φ might access the random oracle. This is reminiscent of Albrecht *et al.* [5], who considered RKA-security of symmetric encryption in the ideal cipher model with RKA-functions that depend on the ideal cipher. To show security in this stronger setting, we need to place additional restrictions on the adversary (as shown by the direct attack in the previous section). Here, we will consider the following limitation of the adversary's queries.

Definition 12 (Indirect H-query uniqueness). Consider an adversary \mathcal{A} interacting in the Φ -IND-RR-CCA security game in the random oracle model. \mathcal{A} is said to respect indirect H-query uniqueness if, all random oracle queries caused by \mathcal{A} 's queries to his ENC and LR oracles, are unique.

Note that, in the above definition, \mathcal{A} is not restricted in terms of his queries directly to the random oracle; only the indirect queries caused by \mathcal{A} 's ENC and LR queries are restricted. With this definition in place, we can now show the following result for the REwH construction.

Theorem 5. Let PKE be an IND-CCA secure PKE scheme, and let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$, be an output-unpredictable function family, where $\mathcal{R} = \mathcal{R}_\lambda$ is the randomness space of PKE.Enc. Then the REwH scheme based on PKE is Φ -IND-RR-CCA secure against adversaries respecting indirect H-query uniqueness, assuming the hash function in the REwH construction is modeled as a random oracle. More precisely, for all equality and indirect H-query uniqueness respecting adversaries \mathcal{A} making $q_{lr} = q_{lr}(\lambda)$ LR queries, $q_{enc} = q_{enc}(\lambda)$ ENC queries, and $q_{RO} = q_{RO}(\lambda)$ random oracle queries, there exists an algorithm \mathcal{B} such that

$$\text{Adv}_{\text{REwH}, \mathcal{A}}^{\text{IND-RR-CCA}}(\lambda) \leq \text{Adv}_{\text{PKE}, \mathcal{B}}^{\text{IND-CCA}}(\lambda) + 2q_{RO}(q_{lr} + q_{enc}) \cdot \text{UP}^\Phi(\lambda).$$

The proof of the above theorem can be found in the full version of the paper.

Note that in the above theorem, collision resistance of Φ is not required. This is because the indirect H-query uniqueness property will prevent an adversary from submitting functions ϕ_1, ϕ_2 to his ENC and LR oracles, for which a collision $\phi_1(r) = \phi_2(r)$ occurs, assuming the queried public keys and messages are the same. (If the submitted public keys and messages are different, indirect H-query uniqueness will not imply that a collision cannot occur, but this will not affect the proof, since the inputs to the random oracle will remain distinct).

The requirement that the adversary is indirect H-query uniqueness respecting might be considered to be somewhat artificial, in that there seems to be no reasonable argument for this assumption to hold for adversaries in the practical settings in which related randomness attacks might be a concern. In the following sections, we will explore the possibilities of achieving security in the standard model, under the arguably realistic assumption that the adversary can only mount a limited number of queries before new entropy is added to the system on which encryption is being done.

6 Bounded RKA and Correlated-Input Security from t -wise Independent Hash Functions

In this section, we show how to construct the building blocks needed for our standard-model IND-RRR-CCA-secure PKE scheme. More concretely, we will start out by showing a key-dependent variant of the leftover hash lemma for correlated inputs. This, in turn, allows us to show that a family of t -wise independent hash functions leads to a bounded correlated-input secure function family, in the

sense that a bound for the number q of correlated inputs must be known a priori. Finally, we will then show how a PRF (with public parameters) that provides RKA-security as long as an adversary makes at most q related key derivation queries, can be constructed from an ordinary PRF and a q bounded correlated-input secure function family. This type of PRF will be used to construct our IND-RRR-CCA-secure PKE scheme in Sect. 7. We believe that each of the intermediate results might find other applications than the construction of related randomness secure PKE scheme, and hence, might be of independent interest.

6.1 Key-Dependent Leftover Hash Lemma for Correlated Inputs

The ordinary leftover hash lemma [28] requires that the input to the hash function is chosen independently of the description of the hash function (i.e. the hash key). The first key-dependent versions of the leftover hash lemma were shown in [21, 41], and was extended to consider leakage in [14]. A “crooked” version for block sources was shown in [38].

The version of the leftover hash lemma that we will show in the following, differs from the previous work in that we consider unrestricted inputs which can both be arbitrarily correlated and key-dependent. Our lemma is as follows.

Lemma 1. *Let $\mathcal{H} : D \rightarrow R$ be a family of t -wise independent hash functions where $t > 8$ is an even number, and let \mathcal{X} be a family of collections of q (correlated) random variables $\mathbf{X} = (X_1, \dots, X_q)$ over D , such that $\mathbf{H}_\infty(X_i) \geq \gamma$ for all $1 \leq i \leq q$, and $\Pr[X_i = X_j] = 0$ for all $1 \leq i \neq j \leq q$. Furthermore, let $\epsilon, \delta > 0$ be such that*

$$t \geq \log |\mathcal{X}| + q \log |R| + \log \frac{1}{\delta}, \quad \text{and} \quad \gamma \geq q \log |R| + 2 \log \frac{1}{\epsilon} + \log t + 2. \quad (1)$$

Then, with probability $1 - \delta$ over the choice of $H \leftarrow_{\S} \mathcal{H}$,

$$\Delta[H(\mathbf{X}), \underbrace{(U_R, \dots, U_R)}_q] \leq \epsilon$$

holds for all $\mathbf{X} \in \mathcal{X}$, where U_R denotes the uniform distribution on R .

Proof (of Lemma 1). We start by considering a fixed collection of random variables $\mathbf{X} = (X_1, \dots, X_q)$ such that $\mathbf{H}_\infty(X_i) \geq \gamma$ for all $1 \leq i \leq q$ and $\Pr[X_i = X_j] = 0$ for all $1 \leq i \neq j \leq q$, and a fixed value $\mathbf{y} \in R^q$. Note that the condition of \mathbf{X} implies that every coordinate of (an outcome of) \mathbf{X} is always distinct. Therefore, due to the t -wise independence of \mathcal{H} , and that $q < t$, we must have that, for any \mathbf{x} in the support of \mathbf{X} (which is a subset of D^q),

$$\Pr_{H \leftarrow_{\S} \mathcal{H}}[H(\mathbf{x}) = \mathbf{y}] = \frac{1}{|R|^q}. \quad (2)$$

Now let $I_{H(\mathbf{x})=\mathbf{y}}$ be the indicator variable that takes on the value 1 if $H(\mathbf{x}) = \mathbf{y}$ (and 0 otherwise), and let $p_{\mathbf{x}} = \Pr[\mathbf{X} = \mathbf{x}] \cdot I_{H(\mathbf{x})=\mathbf{y}}$ and $p = \sum_{\mathbf{x} \in D^q} p_{\mathbf{x}}$. The expected value of p (over the choice $H \leftarrow_{\S} \mathcal{H}$) is then

$$\mathbb{E}[p] = \mathbb{E}\left[\sum_{\mathbf{x} \in D^q} p_{\mathbf{x}}\right] = \sum_{\mathbf{x} \in D^q} \Pr[\mathbf{X} = \mathbf{x}] \cdot \mathbb{E}[I_{H(\mathbf{x})=\mathbf{y}}] = \frac{1}{|R|^q},$$

where the last equality follows from $\mathbb{E}[I_{H(\mathbf{x})=\mathbf{y}}] = \Pr_{H \leftarrow \mathcal{H}}[H(\mathbf{x}) = \mathbf{y}] = |R|^{-q}$, which in turn follows from Eq. (2). Finally let $P_{\mathbf{x}} = 2^\gamma \cdot p_{\mathbf{x}}$ and

$$P = \sum_{\mathbf{x} \in D^q} P_{\mathbf{x}} = 2^\gamma p.$$

The expected value of P must then be $\mathbb{E}[P] = 2^\gamma \cdot \mathbb{E}[p] = 2^\gamma \cdot |R|^{-q}$.

We will now apply the tail bound from Theorem 1 to P and $\mathbb{E}[P]$ (note that the $P_{\mathbf{x}}$ values are t -wise independent due to \mathcal{H} (and thereby also $I_{H(\mathbf{x})=\mathbf{y}}$) being t -wise independent over the choice of H). Doing so yields

$$\begin{aligned} \Pr_{H \leftarrow \mathcal{H}}[|P - \mathbb{E}[P]| \geq \epsilon \cdot \mathbb{E}[P]] &\leq \left(\frac{t \cdot |R|^q}{\epsilon^2 \cdot 2^\gamma}\right)^{\frac{t}{2}} \\ &= \left(\frac{1}{2^{\gamma-2} \log \frac{1}{\epsilon} - \log t - q \log |R|}\right)^{\frac{t}{2}} \\ &\leq 2^{-t}, \end{aligned}$$

where the last inequality follows from the bound on $\log |R|$ given in the theorem. Note that, due to the definition of P and p , we now have that, for any $\epsilon > 0$,

$$\begin{aligned} \Pr_{H \leftarrow \mathcal{H}} \left[\left| \Pr_{\mathbf{x} \leftarrow \mathbf{X}}[H(\mathbf{x}) = \mathbf{y}] - \frac{1}{|R|^q} \right| \geq \frac{\epsilon}{|R|^q} \right] &= \Pr_{H \leftarrow \mathcal{H}} \left[\left| p - \frac{1}{|R|^q} \right| \geq \epsilon \cdot \frac{1}{|R|^q} \right] \\ &= \Pr_{H \leftarrow \mathcal{H}}[|P - \mathbb{E}[P]| \geq \epsilon \cdot \mathbb{E}[P]] \\ &\leq 2^{-t}. \end{aligned}$$

The above inequality holds for any value $\mathbf{y} \in R^q$ and any set $\mathbf{X} = (X_1, \dots, X_q)$ of random variables over D^q , satisfying the criteria given in the theorem. Taking the union bound over all possible $\mathbf{y} \in R^q$ values and all collections $\mathbf{X} \in \mathcal{X}$, yields that with probability $1 - |\mathcal{X}| \cdot |R|^q \cdot 2^{-t}$ over the choice of H , we have that $|\Pr[H(\mathbf{x}) = \mathbf{y}] - |R|^{-q}| \leq \epsilon |R|^{-q}$ for *all* choices of $\mathbf{y} \in R^q$ and $\mathbf{X} \in \mathcal{X}$. This immediately implies that the statistical distance between $H(\mathbf{X})$ and the uniform distribution over R^q , is at most ϵ .

Finally, setting $t \geq \log |\mathcal{X}| + q \log |R| + \log 1/\delta$ ensures that $\delta \geq |\mathcal{X}| \cdot |R|^q \cdot 2^{-t}$, as required. □(Lemma 1)

6.2 Correlated-Input Secure Functions

Firstly, we will formalize the security notion *correlated-input pseudorandomness* (CIPR).

Definition 13. Let $\mathcal{H} = \{H : D \rightarrow R\}$ be a family of (hash) functions with domain $D = D_\lambda$ and range $R = R_\lambda$, $\Phi = \{\phi : D \rightarrow D\}$ be a function family,

and $q = q(\lambda)$ be a positive polynomial. Then, for an adversary \mathcal{A} , consider the security game shown in Fig. 5. In the game, it is required that all queries ϕ submitted by \mathcal{A} belong to Φ , and must be distinct with each other. The advantage of the adversary \mathcal{A} interacting with the security game with respect to \mathcal{H} , is defined to be

$$\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda) = 2 \left| \Pr[\text{CIPR}_{\mathcal{H},q}^{\mathcal{A},\Phi}(\lambda) \Rightarrow 1] - \frac{1}{2} \right|.$$

\mathcal{H} is said to be (q, Φ) -CIPR secure, if for all PPT adversaries \mathcal{A} , $\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda)$ is negligible in the security parameter λ .

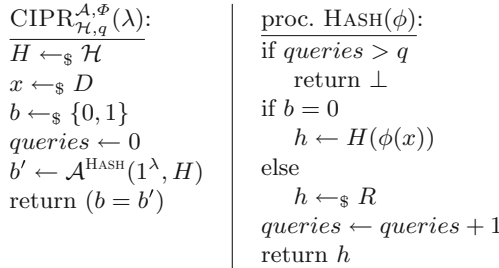


Fig. 5. Game defining correlated-input pseudorandomness (CIPR) of a hash family \mathcal{H} .

The following theorem shows that a t -wise independent hash function family satisfies the above defined CIPR notion.

Theorem 6 (Correlated-Input Pseudorandomness of t -wise Independent Hash Functions). *Let $t = t(\lambda)$, $p = p(\lambda)$, and $q = q(\lambda)$ be integer-valued positive polynomials such that t is always even and larger than 8. Let $\mathcal{H} = \{H : D \rightarrow R\}$ be a family of t -wise independent hash functions with domain $D = D_\lambda$ and range $R = R_\lambda$, let $\Phi = \{\phi : D \rightarrow D\}$ be a function family such that $|\Phi| \leq 2^p$, and let $\text{CR}^\Phi(\lambda) \leq 1/(2\binom{q}{2})$. Furthermore, let $\epsilon = \epsilon(\lambda)$ and $\delta = \delta(\lambda)$ be any functions such that their range is $[0, 1]$ and satisfy:*

$$t \geq q \cdot (p + \log |R|) + \log \frac{1}{\delta} \quad \text{and} \quad \log \frac{1}{\text{UP}^\Phi(\lambda)} \geq q \log |R| + 2 \log \frac{1}{\epsilon} + \log t + 3. \quad (3)$$

Then, for all computationally unbounded adversaries \mathcal{A} that make at most q queries, we have

$$\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda) \leq 2 \cdot |R|^{q-1} \cdot (\epsilon + \delta + \binom{q}{2} \cdot \text{CR}^\Phi(\lambda)).$$

The above theorem immediately gives us the following corollary:

Corollary 1. *Let $t = t(\lambda)$, $p = p(\lambda)$, and $q = q(\lambda)$ be integer-valued positive polynomials such that t is always even and larger than 8. Let $\mathcal{H} = \{H : D \rightarrow R\}$ be a family of t -wise independent hash functions with domain $D = D_\lambda$ and range $R = R_\lambda$ such that $|D| \geq |R| = O(2^\lambda)$. Let $\Phi = \{\phi : D \rightarrow D\}$ be a function family such that $|\Phi| \leq 2^p$. Assume that*

$$\begin{aligned} t &\geq pq + (2q - 1) \log |R| + \lambda, \\ \text{UP}^\Phi(\lambda) &\leq |R|^{-(3q-2)} \cdot 2^{-(2\lambda + O(\log \lambda))}, \\ \text{CR}^\Phi(\lambda) &\leq \binom{q}{2}^{-1} \cdot |R|^{-(q-1)} \cdot 2^{-\lambda}. \end{aligned} \quad (4)$$

Then, for all computationally unbounded adversaries \mathcal{A} that make at most q queries, and for sufficiently large λ , we have

$$\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda) \leq 6 \cdot 2^{-\lambda}.$$

Proof (of Corollary 1). We set $\epsilon = \delta = |R|^{-(q-1)} \cdot 2^{-\lambda}$ in Theorem 6. Then, the assumption on t in Eq. (4) implies the condition required for t in Eq. (3). Furthermore, since p , q , and $\log |R|$ are all polynomials of λ , we have $\log t = O(\log \lambda)$. This fact, combined with the assumption on $\text{UP}^\Phi(\lambda)$ in Eq. (4), implies that $\text{UP}^\Phi(\lambda)$ satisfies the condition required for it in Eq. (3) for all sufficiently large λ . Therefore, we can now invoke Theorem 6: for all computationally unbounded adversaries \mathcal{A} that make at most q queries, and for all sufficiently large λ , we have

$$\begin{aligned} \text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda) &\leq 2 \cdot |R|^{q-1} \cdot (\epsilon + \delta + \binom{q}{2} \cdot \text{CR}^\Phi(\lambda)) \\ &\leq 2 \cdot |R|^{q-1} \cdot (|R|^{-(q-1)} \cdot 2^{-\lambda} + |R|^{-(q-1)} \cdot 2^{-\lambda} + |R|^{-(q-1)} \cdot 2^{-\lambda}) \\ &= 6 \cdot 2^{-\lambda}, \end{aligned}$$

as required. □(Corollary 1)

Now, we proceed to the proof of Theorem 6. The proof consists of two steps. Firstly, we will make use of our variant of the leftover hash lemma (Lemma 1) to show that a t -wise independent hash functions \mathcal{H} satisfies a weaker “non-adaptive” version of correlated-input pseudorandomness, which we denote naCIPR , in which an adversary has to submit all of his hash queries at once parallelly. Then we make use of complexity leveraging to move from naCIPR security to the full CIPR security (this step causes the loss factor $|R|^{q-1}$ appearing in the upperbound of an adversary’s advantage shown in the theorem).

Proof (of Theorem 6). We firstly consider the “non-adaptive” version of the CIPR game shown in Fig. 5, in which an adversary \mathcal{A} has to submit its hash queries non-adaptively (i.e. parallelly). That is, an adversary \mathcal{A} , on input 1^λ and H , submits a set of functions $(\phi_i)_{i \in [q]}$ all at once to the hash oracle HASH , and receives the set of answers $(h_i)_{i \in [q]}$ where each h_i is either the real hash value $H(\phi_i(x))$ or a random value chosen uniformly from the range R of H . Let us denote by $\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{naCIPR}}$ the advantage of an adversary \mathcal{A} in this game.

By using Lemma 1, we show that the advantage of any computationally unbounded non-adaptive adversary, is bounded as stated in the following lemma:

Lemma 2. *Under the same setting as in Theorem 6, for all computationally unbounded adversaries \mathcal{A} that make at most $q = q(\lambda)$ queries, we have*

$$\text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{naCIPR}}(\lambda) \leq 2 \left(\epsilon + \delta + \binom{q}{2} \cdot \text{CR}^\Phi(\lambda) \right). \tag{5}$$

Proof (of Lemma 2). We first introduce several necessary definitions: for a security parameter λ , a hash function $H \in \mathcal{H}$, and a deterministic non-adaptive adversary \mathcal{A} that runs in the naCIPR game and makes q queries, let (ϕ_1, \dots, ϕ_q) be the functions submitted by $\mathcal{A}(1^\lambda, H)$ in \mathcal{A} 's non-adaptive parallel query.⁷ Note that since we are considering a deterministic adversary \mathcal{A} , once we fix \mathcal{A} and $H \in \mathcal{H}$, the functions (ϕ_1, \dots, ϕ_q) are determined without any ambiguity.

Let $\text{NoColl}_{\mathcal{A},H} \subseteq D$ be the subset of D that consists of ‘‘collision-free’’ elements with respect to \mathcal{A} and H , in the following sense:

$$\text{NoColl}_{\mathcal{A},H} := \left\{ x \in D \mid \forall i, j \in [q] \text{ s.t. } i \neq j : \phi_i(x) \neq \phi_j(x) \right\},$$

where each ϕ_i is the i -th function that appears in \mathcal{A} 's parallel query on input $(1^\lambda, H)$. Note that if we pick $x \in D$ uniformly at random, the probability that $\phi_i(x) = \phi_j(x)$ occurs for some (i, j) with $1 \leq i \neq j \leq q$ is upperbounded by $\binom{q}{2} \cdot \text{CR}^\Phi(\lambda)$. This implies $\Pr_{x \leftarrow_{\S} D}[x \in \text{NoColl}_{\mathcal{A},H}] \geq 1 - \binom{q}{2} \cdot \text{CR}^\Phi(\lambda)$. Equivalently, we have

$$|\text{NoColl}_{\mathcal{A},H}| \geq \left(1 - \binom{q}{2} \cdot \text{CR}^\Phi(\lambda)\right) \cdot |D| \geq \frac{1}{2} \cdot |D|, \tag{6}$$

where in the last inequality we use $\text{CR}^\Phi(\lambda) \leq 1/(2\binom{q}{2})$.

Then, we define the random variable $\mathbf{X}_{\mathcal{A},H} = (X_1, \dots, X_q)$, defined over D^q , as follows:

$$\mathbf{X}_{\mathcal{A},H} = (X_1, \dots, X_q) := \left\{ x \leftarrow_{\S} \text{NoColl}_{\mathcal{A},H}; \forall i \in [q] : x_i \leftarrow \phi_i(x) : (x_1, \dots, x_q) \right\}. \tag{7}$$

We then define \mathcal{X} to be the set consisting of the random variables $\mathbf{X}_{\mathcal{A},H}$ for all possible deterministic non-adaptive adversaries \mathcal{A} and all hash functions $H \in \mathcal{H}$. Namely, we define

$$\mathcal{X} := \bigcup_{\mathcal{A}} \left\{ \mathbf{X}_{\mathcal{A},H} \mid H \in \mathcal{H} \right\}, \tag{8}$$

where the union is taken over all possible non-adaptive adversaries \mathcal{A} .

⁷ We will later show an upperbound of the advantage for all *computationally unbounded* non-adaptive adversaries \mathcal{A} in the naCIPR game, in which case considering whether \mathcal{A} is deterministic or probabilistic does not matter because a computationally unbounded adversary can find its best randomness and use this. Hence, considering only deterministic adversaries here is sufficient for our purpose.

We note that each ϕ_i in an adversary \mathcal{A} 's parallel query belongs to the set Φ (no matter what the adversary \mathcal{A} is and no matter what hash function $H \in \mathcal{H}$ \mathcal{A} receives), and note also that $|\Phi| \leq 2^q$ holds. Therefore, the number of distinct random variables $\mathbf{X}_{\mathcal{A},H}$ is at most 2^{pq} , namely, we have $|\mathcal{X}| \leq 2^{pq}$. Furthermore, note also that by definition, we have $\Pr[X_i = X_j] = 0$ for all $i \neq j \in [q]$ and all $\mathbf{X}_{\mathcal{A},H} = (X_1, \dots, X_q) \in \mathcal{X}$ (no matter what \mathcal{A} is and no matter what hash function $H \in \mathcal{H}$ \mathcal{A} receives).

We now consider the min-entropy of each coordinate X_i of the random variables $\mathbf{X}_{\mathcal{A},H} \in \mathcal{X}$. By applying the lemma by Dodis and Yu [22, Lemma 1] and Eq. (6), for every $\phi \in \Phi$ and $y \in D$, we have

$$\Pr_{x \leftarrow_{\S} \text{NoColl}_{\mathcal{A},H}}[\phi(x) = y] \leq \frac{|D|}{|\text{NoColl}_{\mathcal{A},H}|} \cdot \Pr_{x \leftarrow_{\S} D}[\phi(x) = y] \leq 2 \cdot \Pr_{x \leftarrow_{\S} D}[\phi(x) = y]. \quad (9)$$

Furthermore, by definition $\max_{y \in D} \{\Pr_{x \leftarrow_{\S} D}[\phi(x) = y]\} \leq \text{UP}^\Phi(\lambda)$ holds for every $\phi \in \Phi$. By combining this with Eq. (9), for every $i \in [q]$, we have

$$\begin{aligned} H_\infty(X_i) &= -\log\left(\max_{y \in D} \left\{ \Pr_{x \leftarrow_{\S} \text{NoColl}_{\mathcal{A},H}}[\phi_i(x) = y] \right\}\right) \\ &\geq -\log\left(\max_{y \in D} \left\{ 2 \cdot \Pr_{x \leftarrow_{\S} D}[\phi_i(x) = y] \right\}\right) \geq \log \frac{1}{2\text{UP}^\Phi(\lambda)}. \end{aligned} \quad (10)$$

In words, we have seen that for all random variables $\mathbf{X} = (X_1, \dots, X_q) \in \mathcal{X}$, the min-entropy of each X_i is lowerbounded by $\log(1/2\text{UP}^\Phi(\lambda))$.

For a number $\epsilon' > 0$, define the set $\text{GoodHash}_{\epsilon'} \subseteq \mathcal{H}$ by

$$\text{GoodHash}_{\epsilon'} := \left\{ H \in \mathcal{H} \mid \forall \mathbf{X} \in \mathcal{X} : \Delta[H(\mathbf{X}), \underbrace{(U_R, \dots, U_R)}_q] \leq \epsilon' \right\}.$$

Recall that $|\mathcal{X}| \leq 2^{pq}$. Hence, by Eq. (10), if $\delta' > 0$ is a number such that

$$t \geq q \cdot (\log |R| + p) + \log \frac{1}{\delta'} \quad \text{and} \quad \log \frac{1}{2 \cdot \text{UP}^\Phi(\lambda)} \geq q \log |R| + 2 \log \frac{1}{\epsilon'} + \log t + 2,$$

then the condition on t in Eq. (1) in Lemma 1 is satisfied. Furthermore, due to Eq. (10) and the assumption on $\log(1/\text{UP}^\Phi(\lambda))$ in Lemma 2, all random variables $\mathbf{X} = (X_1, \dots, X_q) \in \mathcal{X}$ satisfy the second condition (i.e. the lowerbound on the min-entropy in each entry X_i) in Eq. (1). Hence, by applying Lemma 1 to the set of variables \mathcal{X} (which we have seen satisfies all the requirements for Lemma 1), we have $|\text{GoodHash}_{\epsilon'}| \geq (1 - \delta') \cdot |\mathcal{H}|$.

Having defined the things we need, we are now ready to show an upperbound on the advantage of all non-adaptive adversaries \mathcal{A} in the **naCIPR** game. Fix arbitrarily a computationally unbounded adversary \mathcal{A} that makes at most q queries in the **naCIPR** game. Fix also arbitrarily functions $\epsilon = \epsilon(\lambda)$ and $\delta = \delta(\lambda)$ satisfying Eq. (3). Our goal is to show that Eq. (5) is satisfied for the above \mathcal{A} , and numbers $\epsilon' = \epsilon$, and $\delta' = \delta$.

Let \mathbf{S} be the event that \mathcal{A} succeeds in guessing its challenge bit (i.e. $b' = b$ occurs), and let \mathbf{GH} (which stands for “**G**ood **H**ash”) be the event that the hash function H that \mathcal{A} receives satisfies $H \in \text{GoodHash}_\epsilon$, and let \mathbf{NC} (which stands for “**N**o **C**ollision”) be the event that there exist no indices $i, j \in [q]$ such that $\phi_i(x) = \phi_j(x)$, where $x \in D$ is the value chosen randomly at the non-adaptive game, and ϕ_i (resp. ϕ_j) be the i -th (resp. j -th) function in the parallel query (ϕ_1, \dots, ϕ_q) submitted by \mathcal{A} on input $(1^\lambda, H)$.

We proceed to estimating lower and upperbounds for $\Pr[\mathbf{S}]$. On the one hand, we have

$$\begin{aligned} \Pr[\mathbf{S}] &\geq \Pr[\mathbf{S} \wedge \mathbf{GH} \wedge \mathbf{NC}] \\ &= \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] \cdot \Pr[\mathbf{GH} \wedge \mathbf{NC}] \\ &= \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] \cdot (1 - \Pr[\overline{\mathbf{GH}} \vee \overline{\mathbf{NC}}]) \\ &\geq \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] - \Pr[\overline{\mathbf{GH}}] - \Pr[\overline{\mathbf{NC}}]. \end{aligned} \tag{11}$$

On the other hand, we have

$$\begin{aligned} \Pr[\mathbf{S}] &= \Pr[\mathbf{S} \wedge \mathbf{GH} \wedge \mathbf{NC}] + \Pr[\mathbf{S} \wedge (\overline{\mathbf{GH}} \vee \overline{\mathbf{NC}})] \\ &\leq \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] + \Pr[\overline{\mathbf{GH}} \vee \overline{\mathbf{NC}}] \\ &\leq \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] + \Pr[\overline{\mathbf{GH}}] + \Pr[\overline{\mathbf{NC}}]. \end{aligned} \tag{12}$$

Here, by definition, we have $\Pr[\mathbf{GH}] \geq 1 - \delta$ and $\Pr[\mathbf{NC}] \geq 1 - \binom{q}{2} \cdot \text{CR}^{\bar{\Phi}}(\lambda)$, where the probabilities in the left hand side of both of the inequalities are over the naCIPR game. Furthermore, the event \mathbf{S} conditioned on \mathbf{GH} and \mathbf{NC} , corresponds to the situation where \mathcal{A} , on input 1^λ and $H \in \text{GoodHash}_\epsilon$, receives (h_1, \dots, h_q) that is sampled from either the distribution $H(\mathbf{X}_{\mathcal{A},H})$ where $\mathbf{X}_{\mathcal{A},H} \in \mathcal{X}$ or the uniform distribution $(U_R)^q$ over R^q , and succeeds in guessing which is the case. Here, due to the definitions of GoodHash_ϵ and $\mathbf{X}_{\mathcal{A},H}$, the statistical distance between $H(\mathbf{X}_{\mathcal{A},H})$ and the uniform distribution $(U_R)^q$ is at most ϵ . Hence, we have

$$\frac{1}{2} - \epsilon \leq \Pr[\mathbf{S}|\mathbf{GH} \wedge \mathbf{NC}] \leq \frac{1}{2} + \epsilon.$$

Combining these inequalities with Eqs. (11) and (12), we obtain

$$-(\epsilon + \delta + \binom{q}{2} \cdot \text{CR}^{\bar{\Phi}}(\lambda)) \leq \Pr[\mathbf{S}] - \frac{1}{2} \leq \epsilon + \delta + \binom{q}{2} \cdot \text{CR}^{\bar{\Phi}}(\lambda),$$

which implies

$$\text{Adv}_{\mathcal{H},q,\mathcal{A},\bar{\Phi}}^{\text{naCIPR}}(\lambda) = 2 \left| \Pr[\mathbf{S}] - \frac{1}{2} \right| \leq 2 \left(\epsilon + \delta + \binom{q}{2} \cdot \text{CR}^{\bar{\Phi}}(\lambda) \right),$$

as required. \square (**Lemma 2**)

Finally, as the last step of the proof of Theorem 6, we show that by a complexity leveraging argument, ordinary (adaptive) correlated-input pseudorandomness is implied by its non-adaptive version. More precisely, we show the following lemma:

Lemma 3. *Let $q = q(\lambda)$ be a positive polynomial. Let $\mathcal{H} = \{H : D \rightarrow R\}$ and $\Phi = \{\phi : D \rightarrow D\}$ be families of functions with domain $D = D_\lambda$ and ranges $R = R_\lambda$ and D , respectively. Then, for all computationally unbounded adversaries \mathcal{A} that make q queries, there exists a computationally unbounded non-adaptive adversary \mathcal{B} that makes q queries, such that*

$$\text{Adv}_{\mathcal{H},q,\mathcal{B},\Phi}^{\text{naCIPR}}(\lambda) = \frac{1}{|R|^{q-1}} \cdot \text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda). \quad (13)$$

Proof (of Lemma 3). Fix arbitrarily a positive polynomial q and a computationally unbounded adversary \mathcal{A} that runs in the CIPR game and makes q queries. Using \mathcal{A} as a building block, we show how to construct another computationally unbounded adversary \mathcal{B} that runs in the naCIPR game, makes in exactly the same number of queries as \mathcal{A} , and has the advantage as stated in Eq. (13). The description of \mathcal{B} is as follows:

$\mathcal{B}(1^\lambda, H)$: \mathcal{B} first chooses $q - 1$ values $h'_1, \dots, h'_{q-1} \leftarrow_{\S} R$ uniformly at random, and runs $\mathcal{A}(1^\lambda, H)$, where \mathcal{B} answers to \mathcal{A} 's i -th query ϕ_i by h'_i (no matter what ϕ_i is). When \mathcal{A} makes the q -th query ϕ_q , \mathcal{B} submits q functions $(\phi_i)_{i \in [q]}$ as its “parallel” query to \mathcal{B} 's hash oracle, and receives the results $(h_i^*)_{i \in [q]}$. Then, \mathcal{B} proceeds as follows:

- If $h_i^* = h'_i$ holds for all $i \in [q - 1]$, then \mathcal{B} finds that its simulation for \mathcal{A} was “good”, and returns h_q^* as the answer to \mathcal{A} 's q -th query. When \mathcal{A} terminates with output b' , \mathcal{B} sets $\sigma' \leftarrow b'$.
- Otherwise (i.e. $h_i^* \neq h'_i$ holds for some $i \in [q - 1]$), \mathcal{B} decides that it does not use \mathcal{A} 's output, and sets $\sigma' \leftarrow_{\S} \{0, 1\}$ uniformly at random.

Finally, \mathcal{B} terminates with output σ' .

The above completes the description of \mathcal{B} . Let σ be \mathcal{B} 's challenge bit in its non-adaptive game. Furthermore, let S be the event that $\sigma' = \sigma$ occurs, and G be the event that $h_i^* = h'_i$ holds for all $i \in [q - 1]$ (where both of the events are defined in \mathcal{B} 's naCIPR game). By definition, \mathcal{B} 's advantage in the naCIPR game can be estimated as follows:

$$\begin{aligned} \text{Adv}_{\mathcal{H},q,\mathcal{B},\Phi}^{\text{naCIPR}}(\lambda) &= 2 \left| \Pr[\text{S}] - \frac{1}{2} \right| \\ &= 2 \left| \Pr[\text{S}|\text{G}] \cdot \Pr[\text{G}] + \Pr[\text{S}|\overline{\text{G}}] \cdot \Pr[\overline{\text{G}}] - \frac{1}{2}(\Pr[\text{G}] + \Pr[\overline{\text{G}}]) \right| \\ &= 2 \left| \Pr[\text{G}] \cdot (\Pr[\text{S}|\text{G}] - \frac{1}{2}) + \Pr[\overline{\text{G}}] \cdot (\Pr[\text{S}|\overline{\text{G}}] - \frac{1}{2}) \right|. \end{aligned} \quad (14)$$

Now, since all $\{h'_i\}_{i \in [q-1]}$ are chosen uniformly at random, independently of \mathcal{A} 's behavior and \mathcal{B} 's challenge bit, we have $\Pr[\text{G}] = 1/|R|^{q-1}$. Moreover, once G occurs, \mathcal{B} simulates the CIPR game perfectly for \mathcal{A} so that \mathcal{A} 's challenge bit is that of \mathcal{B} 's, and thus $\Pr[\text{S}|\text{G}]$ is equal to the probability that \mathcal{A} succeeds in guessing the challenge bit in the CIPR game. This implies $2|\Pr[\text{S}|\text{G}] - 1/2| = \text{Adv}_{\mathcal{H},q,\mathcal{A},\Phi}^{\text{CIPR}}(\lambda)$. On the other hand, if G does not occur, \mathcal{B} uses a uniformly chosen random bit as its final output bit σ' , which implies $\Pr[\text{S}|\overline{\text{G}}] = 1/2$. Using the above in Eq. (14), we obtain Eq. (13), as required. \square (**Lemma 3**)

Theorem 6 follows from the combination of Lemmas 2 and 3. \square (Theorem 6)

6.3 Bounded RKA-Secure PRF

Finally, we show that by combining a (q, Φ) -CIPR-secure function family with a standard PRF, we obtain a PRF that provides Φ -RKA security, as long as an adversary uses at most q functions for deriving related keys in the security game. We stress that although the number of *functions* is a-priori bounded by q , the number of *evaluations* that an adversary may observe (through EVAL queries) is unbounded. We refer to this slightly weaker variant of Φ -RKA security of a PRF as (q, Φ) -RKA security.

We formally define (q, Φ) -RKA security of a PRF via the security game shown in Fig. 6. This game is a simple modification of the PRF game in Sect. 2.4. Specifically, in the (q, Φ) -RKA security game, an initial key k^* is picked, and the game maintains a counter ctr (initialized to 0) that tracks the number of related keys the adversary has requested. The oracle RKD (which stands for **Related-Key Derivation**) takes a function $\phi \in \Phi$ as input, increments the counter $ctr \leftarrow ctr + 1$, computes a related key $k_{ctr} \leftarrow \phi(k^*)$, and returns the handle ctr that can be used in an EVAL query to specify the index of the key under which an adversary wish to see an evaluation result. Furthermore, like the HASH oracle in the CIPR game, the oracle RKD can be used at most q times, and all functions used in RKD queries are required to be distinct. However, we again stress that there is no restriction on the number of queries on the EVAL oracle.

Definition 14. Let Φ be a function family, and let the advantage of an adversary \mathcal{A} playing the security game in Fig. 6 with respect to a PRF $F = (\text{Setup}, \text{KeyGen}, \text{Eval})$ be defined as

$$\text{Adv}_{F,q,\mathcal{A},\Phi}^{\text{RKAPRF}}(\lambda) = 2 \left| \Pr[\text{RKAPRF}_{q,\mathcal{A},\Phi}^F(\lambda) \Rightarrow 1] - \frac{1}{2} \right|.$$

F is said to be a (q, Φ) -RKA secure if for all PPT adversaries \mathcal{A} , $\text{Adv}_{F,q,\mathcal{A},\Phi}^{\text{RKAPRF}}(\lambda)$ is negligible in the security parameter λ .

$\text{RKAPRF}_{q,\mathcal{A},\Phi}^F(\lambda):$ $par \leftarrow F.\text{Setup}(1^\lambda)$ $b \leftarrow_{\mathcal{S}} \{0, 1\}$ $\mathcal{F} \leftarrow \emptyset$ $k^* \leftarrow F.\text{KeyGen}(par)$ $ctr \leftarrow 0$ $b' \leftarrow \mathcal{A}^{\text{FUNC}, \text{RKD}}(par)$ return $(b = b')$	proc. EVAL(i, x): if $i > ctr$, return \perp if $b = 1$ $y \leftarrow F.\text{Eval}(k_i, x)$ else if $\mathcal{F}[i, x] = \perp$, $\mathcal{F}[i, x] \leftarrow_{\mathcal{S}} R$ $y \leftarrow \mathcal{F}[i, x]$ return y	proc. RKD($\phi \in \Phi_\lambda$): If $ctr > q$, return \perp $ctr \leftarrow ctr + 1$ $k_{ctr} \leftarrow \phi(k^*)$ return ctr
---	---	---

Fig. 6. Game defining (q, Φ) -RKA security of a PRF.

We will now show how we construct a (q, Φ) -RKA secure PRF. Let $\mathcal{H} = \{H : D \rightarrow R\}$ be a family of functions with domain $D = D_\lambda$ and range $R = R_\lambda$, and

let F be a PRF. We assume that the key space of F is R , and furthermore that $F.\text{KeyGen}(par)$ just samples a uniformly random element from R , and outputs this as a key, for any par output from $F.\text{Setup}(1^\lambda)$. Using these components, we construct another pseudorandom function \widehat{F} as in Fig. 7. Note that the key space of \widehat{F} (when set up with the security parameter λ) is D (which is equal to the domain of the hash function $H \in \mathcal{H}$).

$\begin{array}{l} \text{Alg. } \widehat{F}.\text{Setup}(1^\lambda): \\ par' \leftarrow F.\text{Setup}(1^\lambda) \\ H \leftarrow_{\S} \mathcal{H}_\lambda \\ par \leftarrow (par', H) \\ \text{return } par \end{array}$	$\begin{array}{l} \text{Alg. } \widehat{F}.\text{KeyGen}(par): \\ k \leftarrow_{\S} D_\lambda \\ \text{return } k \end{array}$	$\begin{array}{l} \text{Alg. } \widehat{F}.\text{Eval}(par, k, x): \\ (par', H) \leftarrow par \\ \tilde{k} \leftarrow H(k) \\ y \leftarrow F.\text{Eval}(par', \tilde{k}, x) \\ \text{return } y \end{array}$
--	--	--

Fig. 7. (q, Φ) -RKA-secure PRF \widehat{F} constructed from a standard PRF F and a (q, Φ) -CIPR-secure function family \mathcal{H} .

Theorem 7. *Let $q = q(\lambda)$ be any positive polynomial, let $\Phi = \{\phi : D \rightarrow D\}$ be a family of functions with domain and range $D = D_\lambda$, and let $\mathcal{H} = \{H : D \rightarrow R\}$ be a (q, Φ) -CIPR secure family of (hash) functions with domain D and range $R = R_\lambda$.⁸ Let F be a secure PRF with key space R (when set up with the security parameter λ), and with a key generation algorithm that outputs a uniformly random element from R . Then, the construction \widehat{F} shown in Fig. 7 is (q, Φ) -RKA secure. More precisely, for all PPT adversaries \mathcal{A} , there exist PPT adversaries \mathcal{B}_1 and \mathcal{B}_2 , such that*

$$\text{Adv}_{\widehat{F}, q, \mathcal{A}, \Phi}^{\text{RKAPRF}}(\lambda) \leq \text{Adv}_{\mathcal{H}, q, \mathcal{B}_1, \Phi}^{\text{CIPR}}(\lambda) + \text{Adv}_{F, \mathcal{B}_2}^{\text{PRF}}(\lambda). \tag{15}$$

The intuition behind the proof of this theorem is fairly simple. Recall that (q, Φ) -CIPR security of the underlying hash family \mathcal{H} essentially ensures the property that, for a randomly chosen function $H \leftarrow_{\S} \mathcal{H}$, and for any functions $\phi_1, \dots, \phi_q \in \Phi$, having access to the functions $\{F.\text{Eval}(H(\phi_i(k^*), \cdot))\}_{i \in [q]}$ is indistinguishable from having access to the functions $\{F.\text{Eval}(\tilde{k}_i, \cdot)\}_{i \in [q]}$, where $k^* \in D$ and each $\tilde{k}_i \in R$ are chosen uniformly at random. Then, the security of the PRF F ensures that the latter is indistinguishable from having access to q independently chosen random functions. The full proof of Theorem 7 can be found in the full version of the paper.

7 IND-RRR-CCA Security in the Standard Model

We will now show that, for any predetermined polynomial n , we can transform a PKE scheme PKE which is secure in the standard sense (without related-randomness security) into a scheme PRF-PKE that is (n, Φ, Ψ) -IND-RRR-CCA secure

⁸ In this statement, the requirements regarding output-unpredictability and collision resistance on Φ are implicitly posed by the requirement that \mathcal{H} is (q, Φ) -CIPR secure (c.f. Theorem 6 and Corollary 1).

<p>Alg. PRF-PKE.Setup(1^λ): $par' \leftarrow \text{PKE.Setup}(1^\lambda)$ $par'' \leftarrow \widehat{F}.\text{Setup}$ $par \leftarrow (par', par'')$ return par</p>	<p>Alg. PRF-PKE.Enc(pk, m): $r \leftarrow_{\mathcal{S}} \mathcal{R}$ $\tilde{r} \leftarrow \widehat{F}.\text{Eval}(par, r, pk m)$ $c \leftarrow \text{PKE.Enc}(pk, m; \tilde{r})$ return c</p>
<p>Alg. PRF-PKE.KeyGen(par): $(par', par'') \leftarrow par$ $(pk, sk) \leftarrow \text{PKE.KeyGen}(par')$ return (pk, sk)</p>	<p>Alg. PRF-PKE.Dec(sk, c): $m \leftarrow \text{PKE.Dec}(sk, c)$ return m</p>

Fig. 8. Scheme PRF-PKE constructed from a PKE scheme PKE and a PRF \widehat{F} .

in the standard model, by using a (n, Θ) -RKA secure PRF for an appropriate function class Θ . This approach is similar to that of [34, 43], but we obtain security for a much richer class of function families that captures non-algebraic functions, such as bit-flipping and bit-fixing functions.

More formally, the construction of PRF-PKE is as follows: let PKE be a PKE scheme for which the randomness space of PKE.Enc is $\{0, 1\}^\lambda$, let \widehat{F} be a PRF with key space \mathcal{R} and a key generation algorithm $\widehat{F}.\text{KeyGen}(par)$ returning a uniformly random element from \mathcal{R} as a key, for any par output by $\widehat{F}.\text{Setup}(1^\lambda)$. Using these components, we construct a PKE scheme PRF-PKE as in Fig. 8. Note that the randomness space of PRF-PKE.Enc is \mathcal{R} . The related-randomness security of PRF-PKE is guaranteed by the following theorem:

Theorem 8. *Let $n = n(\lambda)$ be an integer-valued positive polynomial. Let $\Phi = \{\phi : \mathcal{R} \rightarrow \mathcal{R}\}$ and $\Psi = \{\psi : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}\}$ be function families, where $\mathcal{R} = \mathcal{R}_\lambda$. Let Θ be the function family defined by using Φ and Ψ as follows:*

$$\Theta := \left\{ f(\cdot) := \phi(\psi(r, \cdot)) \mid \phi \in \Phi, \psi \in \Psi, r \in \mathcal{R} \right\} \cup \Phi.$$

Let \widehat{F} be a (n, Θ) -RKA secure PRF⁹, and let PKE be an IND-CCA secure PKE scheme. Then, the construction PRF-PKE shown in Fig. 8 is (n, Φ, Ψ) -IND-RRR-CCA secure. More precisely, for all PPT (n, Φ, Ψ) -restricted adversaries \mathcal{A} that make at most $q_r = q_r(\lambda)$ REFRESH queries, there exist PPT adversaries \mathcal{B}_1 and \mathcal{B}_2 such that

$$\text{Adv}_{\text{PRF-PKE}, \mathcal{A}}^{\text{IND-RRR-CCA}}(\lambda) \leq 2(q_r + 1) \text{Adv}_{\widehat{F}, n, \mathcal{B}_1, \Theta}^{\text{RKAPRF}}(\lambda) + \text{Adv}_{\text{PKE}, \mathcal{B}_2}^{\text{IND-CCA}}(\lambda). \quad (16)$$

The proof of Theorem 8 is based on a hybrid argument over the refresh epochs. More specifically, in each epoch, we use the (n, Θ) -RKA security of \widehat{F} to replace the output \tilde{r} with uniformly random values. This is possible since the

⁹ In this statement, the requirements regarding output unpredictability and collision resistance of Φ and Ψ are implicitly implied as \widehat{F} is a (n, Θ) -RKA secure PRF (c.f. Theorem 7). This will be made explicit in Corollary 2.

randomness r' used in the response to LR and ENC oracle queries will correspond to related keys of $\widehat{\mathbf{F}}$ computed by $f \in \Theta$. More precisely, it will be either of the form $r' = \phi(r_1)$ (in the first epoch) or $r' = \phi(\psi_{j-1}(r_{j-1}, s_{j-1}))$ (in the $j(\geq 2)$ -th epoch), where r_1 and s_{j-1} are chosen uniformly at random, and thus can be viewed as related keys of the initial key k^* in the RKA game by viewing r_1 or s_{j-1} as k^* . Note that the adversary is assumed to make in total at most n LR and ENC queries in each epoch, and thus (n, Θ) -RKA security will suffice. Then, in the last hybrid, the values \tilde{r} are all uniformly chosen, and we can rely on the IND-CCA security of the underlying PKE scheme PKE to conclude the proof. The full proof can be found in the full version of the paper.

Combining Theorem 8 with Corollary 1, we obtain the following corollary:

Corollary 2. *Let $t = t(\lambda)$, $p = p(\lambda)$, and $n = n(\lambda)$ be integer-valued positive polynomials such that t is always even and larger than 8. Let PKE be an IND-CCA secure PKE scheme, let \mathbf{F} be a PRF, and let \mathcal{H} be a t -wise independent hash family. Assume that the key space of \mathbf{F} and the output space of \mathcal{H} are $\{0, 1\}^\lambda$ when \mathbf{F} is set up with a security parameter λ . Let $\widehat{\mathbf{F}}$ be the PRF constructed from \mathbf{F} and \mathcal{H} as shown in Fig. 6, and let PKE' be the PKE scheme obtained from PKE and $\widehat{\mathbf{F}}$ as shown in Fig. 8. Let Φ and Ψ be function families such that $|\Phi| \leq 2^p$ and $|\Psi| \leq 2^{p'}$, respectively. Assume that*

$$t \geq n(p + p' + \log |\mathcal{R}| + 2\lambda + 2), \quad (17)$$

$$\max\{\text{UP}^\Phi(\lambda), \text{sUP}^{\Phi, \Psi}(\lambda)\} \leq 2^{-(3n\lambda + O(\log \lambda))}, \quad (18)$$

$$\max\{\text{CR}^\Phi(\lambda), \text{sCR}^{\Phi, \Psi}(\lambda)\} \leq \binom{n}{2}^{-1} \cdot 2^{-n\lambda}. \quad (19)$$

Then, PKE' is (n, Φ, Ψ) -IND-RRR-CCA secure. More precisely, for all PPT (n, Φ, Ψ) -restricted adversaries \mathcal{A} that make at most $q_r = q_r(\lambda)$ REFRESH queries, there exist PPT adversaries \mathcal{B} and \mathcal{B}' such that

$$\text{Adv}_{\text{PKE}', \mathcal{A}}^{\text{IND-RRR-CCA}}(\lambda) \leq 12(q_r + 1) \cdot 2^{-\lambda} + 2(q_r + 1)\text{Adv}_{\mathbf{F}, \mathcal{B}}^{\text{PRF}}(\lambda) + \text{Adv}_{\text{PKE}, \mathcal{B}'}^{\text{IND-CCA}}(\lambda).$$

Proof (of Corollary 2). Note that each function $f \in \Theta$ can be specified by (1) a bit indicating whether f is in the set $\{\phi(\psi(r, \cdot)) \mid \phi \in \Phi, \psi \in \Psi, r \in \mathcal{R}\}$ or in the set Φ , (2-1) a tuple $(\phi, \psi, r) \in \Phi \times \Psi \times \mathcal{R}$ in case f belongs to the former set, and (2-2) a function $\phi \in \Phi$ in case f belongs to the latter set. This implies that $|\Theta| \leq 2 \cdot (|\Phi| \cdot |\Psi| \cdot |\mathcal{R}| + |\Phi|) \leq 2^{p+p'+2} \cdot |\mathcal{R}| = 2^{p''}$, where $p'' = p + p' + 2 + \log |\mathcal{R}|$. Furthermore, by definition, the output unpredictability of Θ is at most the maximum of the output unpredictability of Φ and that of the seed-induced output-unpredictability of Φ with respect to Ψ , i.e. $\max\{\text{UP}^\Phi(\lambda), \text{sUP}^{\Phi, \Psi}(\lambda)\}$, and exactly the same relation holds for collision resistance. Recall also that the output space of \mathcal{H} is $\{0, 1\}^\lambda$.

Now, by using the definition of the function class Θ with the parameters described above in Corollary 1, we obtain the requirements in Eqs. (17), (18), and (19), and the upperbound $6 \cdot 2^{-\lambda}$ for the advantage of any (even computationally unbounded) adversary that attacks the (n, Θ) -CIPR security of \mathcal{H} . Then, using it in turn in Theorem 8, we obtain this corollary. \square (Corollary 2)

The reason the impossibility result from Sect. 4 is not applicable to the above construction, is that for each security parameter λ , with high probability over the choice of the t -wise independent hash function H , the function families Φ and Ψ are not capable of expressing H and thereby the encryption function of the scheme, due to requirement on t in Eq. (17). Note also that, as the size of the description of H must be linear in t , and this description is part of the parameters par'' in Fig. 8, the size of the parameters of the construction will grow linearly in the right hand side of Eq. (17) i.e. linearly in number of queries an adversary is allowed to make in an epoch, and logarithmically in the size of the function families Φ and Ψ .

Acknowledgement. A part of this work was supported by JST CREST grant number JPMJCR1688. Jacob Schuldt was supported by JSPS KAKENHI Grant Number 15K16006.

References

1. Abdalla, M., Benhamouda, F., Passelègue, A.: An algebraic framework for pseudorandom functions and applications to related-key security. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 388–409. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_19
2. Abdalla, M., Benhamouda, F., Passelègue, A.: Algebraic XOR-RKA-Secure Pseudorandom Functions from Post-Zeroizing Multilinear Maps. Cryptology ePrint Archive, Report 2017/500 (2017)
3. Abdalla, M., Benhamouda, F., Passelègue, A., Paterson, K.G.: Related-key security for pseudorandom functions beyond the linear barrier. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 77–94. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_5
4. Applebaum, B., Widder, E.: Related-Key Secure Pseudorandom Functions: The Case of Additive Attacks. Cryptology ePrint Archive, Report 2014/478 (2014)
5. Albrecht, M.R., Farshim, P., Paterson, K.G., Watson, G.J.: On cipher-dependent related-key attacks in the ideal-cipher model. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 128–145. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_8
6. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: how to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_14
7. Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_36
8. Bellare, M., Cash, D., Miller, R.: Cryptography secure against related-key attacks and tampering. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 486–503. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_26

9. 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
10. Bellare, M., Rompel, J.: Randomness-efficient oblivious sampling. In: FOCS 1994, pp. 276–287 (1994)
11. Bellare, M., Tackmann, B.: Nonce-based cryptography: retaining security when randomness fails. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 729–757. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_28
12. Bendel, M.: Hackers describe PS3 security as epic fail, gain unrestricted access (2011). <http://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/>
13. Bernstein, D.J., Chang, Y.-A., Cheng, C.-M., Chou, L.-P., Heninger, N., Lange, T., van Someren, N.: Factoring RSA keys from certified smart cards: Coppersmith in the wild. Cryptology ePrint Archive, Report 2013/599 (2013)
14. Birrell, E., Chung, K.-M., Pass, R., Telang, S.: Randomness-dependent message security. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 700–720. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_39
15. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. In: FOCS 1996, pp. 514–523 (1996)
16. Bitcoin.org: Android security vulnerability (2013). <http://bitcoin.org/en/alert/2013-08-11-android>
17. Chen, Y., Qin, B., Zhang, J., Deng, Y., Chow, S.S.M.: Non-malleable functions and their applications. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 386–416. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_15
18. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: Bounded tamper resilience: how to go beyond the algebraic barrier. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 140–160. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_8
19. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: The chaining lemma and its application. In: Lehmann, A., Wolf, S. (eds.) ICITS 2015. LNCS, vol. 9063, pp. 181–196. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17470-9_11
20. Debian: Debian Security Advisory DSA-1571-1: OpenSSL - predictable random number generator (2008). <http://www.debian.org/security/2008/dsa-1571>
21. Dodis, Y.: Exposure-resilient cryptography. Ph.D. thesis, Massachusetts Institute of Technology (2000)
22. Dodis, Y., Yu, Y.: Overcoming weak expectations. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 1–22. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_1
23. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the random number generator of the windows operating system. ACM Trans. Inf. Syst. Secur. **13**(1), 10 (2009)
24. Goldberg, I., Wagner, D.: Randomness and the Netscape browser (1996). <http://www.drdoobs.com/windows/184409807>
25. Goyal, V., O’Neill, A., Rao, V.: Correlated-input secure hash functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_12

26. Gutterman, Z., Malkhi, D.: Hold your sessions: an attack on java session-id generation. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 44–57. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_5
27. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: S&P, pp. 371–385 (2006)
28. Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: Construction of a pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)
29. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your Ps and Qs: detection of widespread weak keys in network devices. In: USENIX Security 2012, pp. 205–220 (2012)
30. Hoang, V.T., Katz, J., O’Neill, A., Zaheri, M.: Selective-opening security in the presence of randomness failures. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 278–306. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_10
31. Kamara, S., Katz, J.: How to encrypt with a malicious random number generator. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 303–315. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_19
32. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Public keys. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 626–642. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_37
33. National Institute of Standards and Technology (NIST): FIPS Publication 186: Digital Signature Standards (DSS) (1994)
34. Paterson, K.G., Schuldt, J.C.N., Sibborn, D.L.: Related randomness attacks for public key encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 465–482. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_27
35. Paterson, K.G., Schuldt, J.C.N., Sibborn, D.L., Wee, H.: Security against related randomness attacks via reconstructive extractors. In: Groth, J. (ed.) IMACC 2015. LNCS, vol. 9496, pp. 23–40. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27239-9_2
36. Pornin, T.: RFC6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) (2013). <https://tools.ietf.org/html/rfc6979>
37. Qin, B., Liu, S., Yuen, T.H., Deng, R.H., Chen, K.: Continuous non-malleable key derivation and its application to related-key security. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 557–578. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_25
38. Raghunathan, A., Segev, G., Vadhan, S.: Deterministic public-key encryption for adaptively chosen plaintext distributions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 93–110. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_6
39. Ristenpart, T., Yilek, S.: When good randomness goes bad: virtual machine reset vulnerabilities and hedging deployed cryptography. In: NDSS 2010 (2010)
40. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_23
41. Trevisan, L., Vadhan, S.P.: Extracting randomness from samplable distributions. In: FOCS 2000, pp. 32–42 (2000)

42. Vergnaud, D., Xiao, D.: Public-Key Encryption with Weak Randomness: Security Against Strong Chosen Distribution Attacks. *Cryptology ePrint Archive*, Report 2013/681 (2013)
43. Yilek, S.: Resettable public-key encryption: how to encrypt on a virtual machine. In: Pieprzyk, J. (ed.) *CT-RSA 2010*. LNCS, vol. 5985, pp. 41–56. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_4

Subversion Resistance



Subversion-Zero-Knowledge SNARKs

Georg Fuchsbauer^{1,2}(✉)

¹ Inria, Paris, France

² École normale supérieure, CNRS, PSL Research University, Paris, France
`georg.fuchsbauer@ens.fr`

Abstract. Subversion zero knowledge for non-interactive proof systems demands that zero knowledge (ZK) be maintained even when the common reference string (CRS) is chosen maliciously. SNARKs are proof systems with succinct proofs, which are at the core of the cryptocurrency Zcash, whose anonymity relies on ZK-SNARKs; they are also used for ZK contingent payments in Bitcoin.

We show that under a plausible hardness assumption, the most efficient SNARK schemes proposed in the literature, including the one underlying Zcash and contingent payments, satisfy subversion ZK or can be made to at very little cost. In particular, we prove subversion ZK of the original SNARKs by Gennaro et al. and the almost optimal construction by Groth; for the Pinocchio scheme implemented in libsnark we show that it suffices to add 4 group elements to the CRS. We also argue informally that Zcash is anonymous even if its parameters were set up maliciously.

Keywords: Zero knowledge · SNARKs · Parameter subversion
Zcash · Bitcoin contingent payments

1 Introduction

One of the primary motivations for succinct non-interactive arguments (SNARG) was verifiable computation. Consider a client that outsources resource-intensive computation to a powerful server, which attaches a *proof* to the result, so the client is convinced that it was computed correctly. For this to be meaningful, verification of such a proof must be considerably more efficient than performing the computation in the first place. SNARG systems provide such proofs and an impressive line of research has led to more and more efficient systems with proofs of size less than a kilobyte that can be verified in milliseconds. The reason why SNARGs are not used in outsourcing of computation is that computing a proof for complex computations is still not practical. (For example, a proof in Zcash, which is for a very simple statement, takes minutes to compute on a PC.)

Zero-knowledge (ZK) SNARGs are used when some inputs to the computation come from the prover (the server in our example), who wants to keep its inputs private. ZK systems guarantee that a proof does not reveal more about

private inputs than what can be inferred from the result of the computation. If the proofs prove knowledge of the private inputs, they are called SNARKs. ZK-SNARKs are already deployed, for example in Zcash [Zca], which is a cryptocurrency like Bitcoin [Nak09], based on the Zerocash protocol [BCG+14a]. As opposed to Bitcoin, where all transactions are public, Zcash payments are fully anonymous and protect the users' privacy. Zcash achieves this by using SNARK proofs that are zero-knowledge.

Zero-knowledge contingent payments use SNARKs for *fair exchange* of information against payments over the Bitcoin network, assuming that the information can be verified (in the sense that it can be formalized as the witness of an NP statement), e.g. solutions to a Sudoku puzzle. Bitcoin's scripting language defines *Pay-to-PubkeyHash* transactions, which are bound to a hash value y and can be redeemed by exhibiting a preimage, i.e., some x s.t. $H(x) = y$. In a contingent payment Alice, the seller, chooses a key k , encrypts the information she is offering as c under k and sends c together with $y := H(k)$ to Bob, the buyer. Bob makes a transaction to y . To redeem it, Alice must publish the preimage k , which then allows Bob to decrypt c and obtain the purchased information. To prevent Alice from cheating, she must prove that c encrypts the desired information under a preimage of y , for which she can use SNARKs. Zero-knowledge guarantees that no information is leaked before being paid.

The main drawback of SNARKs is that they require system parameters that must be generated in a trusted way. In particular, whoever knows the randomness used when setting them up can convince verifiers of false statements (violating *soundness* of the system), which for Zerocash translates to counterfeiting money. The authors of Zerocash write: “[D]ue to the zk-SNARK, our construction requires a one-time trusted setup of public parameters. The trust affects soundness of the proofs, though anonymity continues to hold even if the setup is corrupted by a malicious party.” [BCG+14a]. The last statement is then not elaborated any further.

For ZK contingent payments (ZKCP) the parameters are generated by the buyer, which prevents the seller from cheating. However, Campanelli et al. [CGGN17] recently showed that the *buyer* can cheat in the reference implementation of ZKCP, which allows for selling the solution to a Sudoku puzzle. By maliciously setting up the parameters, the buyer can learn information about the solution from the SNARK proof sent by the seller before paying. This shows that not only soundness but also zero knowledge of SNARKs breaks down in the face of parameter subversion.

In this work we look at whether zero knowledge can be salvaged when the parameters are set up in a malicious way and analyze the most efficient SNARK constructions in the literature, including the one [BCTV14] that underlies Zcash and ZKCP. We base our analyses on the theoretical framework introduced by Bellare et al. [BFS16], who formalized the notion of *subversion zero knowledge*.

ZERO-KNOWLEDGE PROOFS. A zero-knowledge proof [GMR89] is a protocol between a prover and a verifier that allows the former to convince the latter of the validity of a statement without revealing anything else. The three main

properties of a ZK proof system are that an honestly computed proof for a valid statement should convince a verifier (completeness); but there is no way that a malicious prover can convince a verifier of false statements (soundness); and nothing but the truth of the statement is revealed (zero knowledge).

In *non-interactive* ZK proofs [BFM88], the prover only sends one message (the proof) to the verifier. NIZK systems rely on a *common reference string* (CRS) to which both prover and verifier have access and which must be set up in a trusted way (for SNARKs the CRS is often called *parameters*). Without such a CRS, NIZK systems are not possible [GO94].

NIZK proof systems exist for every NP language [BFM88, BDMP91]. A language L is an NP language if it can be defined via a polynomial-time computable relation R : a statement x is in L iff there exists a *witness* w of length polynomial in the length of x such that $R(x, w) = \text{true}$. In verifiable computation a server's private input would be a witness. For ZK contingent payments, the ciphertext c , the hash value y and the Sudoku challenge are the statement. The witness is the plaintext of c (the Sudoku solution) and the encryption key k .

Zero knowledge is formalized via a *simulator* that generates a CRS in which it can embed a *trapdoor*. The trapdoor must allow the simulator to produce proofs without a witness for the proven statement. ZK requires that there exists a simulator whose simulated CRSs and proofs are computationally indistinguishable from real ones. If both types are distributed equivalently then we have *perfect* ZK. Groth et al. [GOS06b, GOS06a, Gro06, GS08] constructed NIZK proof systems based on groups equipped with a *pairing*, i.e., an efficiently computable bilinear map. They gave the first perfect ZK system for all NP languages and very efficient schemes for specific languages based on standard cryptographic hardness assumptions.

SNARKs. Another line of work considered the size of proofs from a theoretical point of view, leading to schemes with a proof size that is sublinear in the length of the proved statement [Mic00]. SNARGs are succinct non-interactive arguments, where *succinct* means that the proof length only depends (polynomially) on the security parameter. They are *arguments* (as opposed to proofs) because soundness only holds against efficient provers. This is the best achievable notion, since SNARGs are perfect-ZK, which implies that every CRS has a trapdoor. SNARKs are succinct non-interactive arguments *of knowledge*, for which a valid proofs implies that the prover knows the witness.

The first NIZK system with proofs whose size is independent of the proven statement (and its witness) was given by Groth [Gro10] using bilinear groups; it was later improved by Lipmaa [Lip12]. Gennaro et al. [GGPR13] introduced the notion of a quadratic span program (QSP), showed how to efficiently convert any boolean circuit into a QSP and then constructed a SNARK system for QSPs whose proofs consist of 8 elements of a bilinear group. They gave another construction based on quadratic arithmetic programs (QAP), which represent *arithmetic* circuits, whose inputs are elements from a finite field \mathbb{F} and whose gates add or multiply \mathbb{F} elements. QAPs are preferred in practice due to their greater efficiency. As circuit satisfiability is NP-complete, SNARKs exist for all NP languages.

Parno et al. [PHGR13] improved on [GGPR13], making the conversion from circuits to QAPs more efficient and reducing the proof size. They implemented their scheme and named it “Pinocchio”. Ben-Sasson et al. [BCG+13, BCTV14] improve the conversion of actual program code to QAPs, reduce the size of SNARK parameters and implement their results as *libsark* [BCG+14b]. The size of SNARK proofs for boolean circuits was then further reduced by Danezis et al. [DFGK14], who modified QSP to *square* span programs and built a system for them whose proofs consist of only 4 group elements.

Recently, Groth [Gro16] presented the most efficient SNARK construction to date, which is for arithmetic circuits and whose proofs consist of only 3 group elements (and require 3 pairings to verify). All previous bilinear-group-based SNARKs are proven under strong cryptographic assumptions (*knowledge* assumptions), for which there is evidence that they might be unavoidable [GW11, BCCT12]. Starting from Bitansky et al.’s [BCI+13] *linear interactive proof* framework, Groth [Gro16] achieves his result by proving security directly in the generic-group model [Sho97] (which implies all previously considered assumptions). He also shows that SNARKs over asymmetric bilinear groups must contain elements from both source groups, meaning that the proof size of his construction is only one element short of the optimal size. Recently, Fuchsbauer et al. [FKL17] proved Groth’s scheme secure under a “*q*-type” variant of the discrete log assumption in the *algebraic group model*, in which adversaries are restricted adversaries can only output group elements if they were obtained by applying the group operation to previously received group elements.

SUBVERSION-RESISTANCE. The Snowden revelations documented the NSA’s efforts to subvert standards, for which an illustrative example is the NSA-designed and ISO-standardized *Dual EC* random number generator. Its parameters include two elliptic-curve points, whose respective discrete logarithms can act as a backdoor that can be exploited to break TLS [CNE+14]. NIZK systems are particularly prone to parameter subversion, since their CRS must be subvertible *by design*: zero knowledge requires that an honest CRS is indistinguishable from a backdoored CRS, where the backdoor is the trapdoor used to simulate proofs. For SNARKs the parameters always contain a backdoor and anyone knowing it can simulate proofs for false statements, which means breaking soundness.

Motivated by this, Bellare et al. [BFS16] ask what security can be maintained for NIZKs when its trusted parameters are subverted. They formalize different notions of resistance to CRS subversion and investigate their achievability. They define *subversion soundness* (S-SND), meaning that no adversary can generate a (malicious) CRS together with a valid proof π for a false statement x .

They also give a subversion-resistant analogue for zero knowledge. Recall that ZK assumes that there exists a CRS simulator Sim.crs , which returns a simulated CRS crs' and an associated simulation trapdoor td , and a proof simulator Sim.pf that outputs proofs on input a valid instance x and td , such that no efficient adversary can distinguish the following: either being given crs' and an oracle implementing Sim.pf , or an honest crs and an oracle returning hon-

estly computed proofs. Subversion ZK (S-ZK) requires that for any adversary X creating a malicious CRS crs in any way it likes using randomness (coins) r , there exists a simulator $\text{Sim}_X.crs$ returning a simulated CRS crs' with trapdoor td together with simulated coins r' , as well as a proof simulator $\text{Sim}_X.pf$, such that no adversary can distinguish the following: being given crs' and r' and a $\text{Sim}_X.pf$ oracle, or a crs output by X , together with the used coins r and an honest proof oracle. The authors also define a subversion-resistant notion (S-WI) of witness-indistinguishability [FLS90] (see Sects. 2.3 and 2.4).

Following [GO94], Bellare et al. [BFS16] first show that S-SND cannot be achieved together with (standard) ZK for non-trivial languages (for trivial ones the verifier needs no proof to check validity of statements). This is because ZK allows breaking soundness by subverting the CRS. They then show that S-SND can be achieved together with S-WI. Their main result is a construction that achieves both S-ZK (and thus S-WI) and SND.

BFS's S-ZK SCHEME. To achieve S-ZK, a simulator must be able to simulate proofs under a CRS output by a subverter, so it cannot simply embed a trapdoor as in standard ZK. Bellare et al. [BFS16] base S-ZK on a knowledge assumption, which is the type of assumption on which security (in particular, knowledge soundness) of SNARKs relies. It states that an algorithm can only produce an output of a certain form if it knows some underlying information. This is formalized by requiring the existence of an extractor that extracts this information from the algorithm. In their scheme this information acts as the simulation trapdoor, which under their knowledge assumption can be obtained from a subverter outputting a CRS.

Concretely, they assume that for a bilinear group $(\mathbb{G}, +)$ with a generator P any algorithm that outputs a *Diffie-Hellman* tuple (P, s_1P, s_2P, s_1s_2P) for some s_1, s_2 , must know *either* s_1 or s_2 . They call their assumption *Diffie-Hellman knowledge-of-exponent assumption* (DH-KEA) and note that a tuple (P, S_1, S_2, S_3) of this form can be verified via a (symmetric) bilinear map e by checking $e(S_3, P) = e(S_1, S_2)$. A question that arises is: who chooses the group \mathbb{G} in their scheme? Bellare et al. address this by making the group \mathbb{G} part of the scheme specification. This begs the question whether the subversion risk has not simply been shifted from the CRS to the choice of the group. They argue that the group generation algorithm is deterministic and public, so users can create the group themselves, and it is thus *reproducible*, whereas the CRS is inherently not.

PARAMETER SETUP IN PRACTICE. A way to avoid the problem of generating a trusted CRS for NIZK systems is by proving its security in the *random-oracle model* (ROM) [BR93]. Instead of a CRS, all parties are assumed to have access to a truly random function (which is modeled as an oracle returning random values). In practice the random oracle is replaced by a cryptographic hash function and a proof in the ROM can be viewed as a security heuristic for the resulting scheme.

For NIZK systems whose CRS is a uniform random string, e.g. PCP-based constructions like [BSBC+17] recently, one can in practice set the CRS to a common random-looking public value such as the digits of π or the output of a

standardized hash function on a fixed input. This intuitively guarantees that no one has embedded a trapdoor. For the Groth-Sahai proof system [GS08] the CRS consists of random elements of an elliptic-curve group; they can be set up by hashing a common random string directly into the elliptic curve [BF01, BCI+10].

For practical SNARKs the situation is different: there are no CRS-less constructions in the random-oracle model and the CRS is highly structured. The parameters typically contain elements of the form $(P, \tau P, \tau^2 P)$, where P is a generator of a group \mathbb{G} and τ is a random value. Soundness completely breaks down if the value τ is known to anyone. Unfortunately, there is no known way of creating such a triple obliviously, that is, without knowing the value τ .

OUR TECHNIQUES. In order to show subversion zero knowledge of SNARK schemes, we assume that computing elements $(P, \tau P, \tau^2 P)$ cannot be done without knowing τ . (Looking ahead, we actually make a weaker assumption in asymmetric bilinear groups by requiring the adversary to return $(P_1, \tau P_1, \tau^2 P_1) \in \mathbb{G}_1^3$ as well as $(P_2, \tau P_2) \in \mathbb{G}_2^2$, which makes the structure of the triple verifiable using the bilinear map.) Under this assumption, which we call square knowledge of exponent (SKE) assumption (Definition 14), we then prove subversion ZK of five relevant SNARK constructions from the literature or slight variants of them.

As an additional sanity check, we prove that SKE holds in the generic group model (Theorem 16). Following Groth [Gro16], we assume that the bilinear group description is part of the specification of the language for which the proof system is defined (and not part of the CRS as in [BFS16]). Following his previous work [DFGK14], we let the CRS generation algorithm sample *random* group generators (in contrast to [BFS16], which assumes a fixed group generator). This intuitively leads to weaker assumptions required to prove soundness.

To show subversion zero knowledge of existing SNARK schemes, we proceed as follows. Standard zero knowledge holds because the randomness used to compute the CRS allows the simulator to produce proofs that are distributed equivalently to honestly generated proofs under the (honestly computed) CRS. However, for S-ZK this must hold even for a CRS that was computed in any arbitrary way. While we cannot guarantee that the CRS subverter used random values when computing the CRS, we first show how to verify that the *structure* of the CRS is as prescribed. (For the asymmetric Pinocchio scheme [BCTV14] this requires us to extend the CRS slightly.)

Another difference between standard and subversion ZK is that in the former the simulator creates the CRS and thus knows the simulation trapdoor, whereas for S-ZK the CRS is produced by the subverter, so it might not be clear how proofs can be simulated at all. Now if the CRS contains elements $(P, \tau P, \tau^2 P)$, whose correct structure can be verified via the pairing, then under our SKE assumption we can extract the value τ . SKE thus allows the simulator to obtain parts of the randomness even from a maliciously generated CRS. Unfortunately, the simulation trapdoor typically contains *other* values that the S-ZK simulator cannot extract.

Our next step is then to demonstrate that proofs can be simulated using τ only, or to show how under our assumption more values can be extracted that

then enable simulation. Our final step is to show that if a CRS passes the verification procedure we define, then proofs that were simulated using the partial trapdoor are distributed like real proofs. This shows that the analyzed scheme is S-ZK under our SKE assumption. While knowledge assumptions are strong assumptions, they seem unavoidable since S-ZK implies 2-move interactive ZK by letting the verifier create the CRS. And such schemes require extractability assumptions [BCPR14].

Since simulated proofs are by definition independent of a witness, our results imply that under a verified, but possibly malicious, CRS, proofs for different witnesses are equally distributed. As a corollary we thereby obtain that all SNARKs we consider satisfy subversion witness indistinguishability *unconditionally* (i.e., no assumptions required).

We note that Ben-Sasson et al. [BCG+15] also consider making a CRS verifiable. Their goal is to protect *soundness* against subversion by sampling the secret values underlying a CRS in a distributed way. Only if all participants in the CRS-creation protocol collude can they break soundness. To guarantee a correctly distributed CRS, the participant(s) must prove adherence to the protocol via NIZK proofs [Sch91, FS87] secure in the random-oracle model. The protocol thus returns *verifiable* SNARK parameters. The parameters used for Zcash were set up using this multiparty protocol, which was recently detailed by Bowe et al. [BGG17].

Our Results

As already discussed, SNARKs are not subversion-sound because their CRS contains the simulation trapdoor. In this work we look at subversion resistance of their zero-knowledge property and investigate several SNARK constructions from the literature that are based on bilinear groups. In particular,

1. the first QSP-based and 2. QAP-based constructions [GGPR13];
3. optimized Pinocchio [BCTV14] as implemented in libsnark [BCG+14b]; and
4. and 5. the two most efficient constructions by Groth et al. [DFGK14, Gro16].

We make the (reasonable) assumption that a privacy-conscious prover (whose protection is the goal of zero knowledge) first checks whether the CRS looks plausible (to whatever extent this is possible) before publishing a proof with respect to it. All of our results implicitly make this assumption.

We start with the first SNARK construction for QAPs by Gennaro et al. [GGPR13] and show how to verify that the CRS is correctly formed. We then show that under the square knowledge of exponent (SKE) assumption their construction satisfies subversion zero knowledge as defined in [BFS16]. The same holds for their QSP-based SNARK. (Due to space constraints, and since these results follow in much the same way as the next one, we defer our results on GGPR to the full version [Fuc17].)

We next turn to the optimized version of Pinocchio over asymmetric bilinear groups due to Ben-Sasson et al. [BCTV14]. For this construction we show that adding 4 group elements to the CRS makes it efficiently checkable. We then

prove that the scheme with this slightly extended CRS satisfies subversion zero knowledge under SKE, whereas the original scheme, which is implemented in libsnark [BCG+14b], succumbs to a parameter-subversion attack [CGGN17]. For the SNARK by Danezis, Fournet, Groth and Kohlweiss [DFGK14], we show that CRS well-formedness can be efficiently verified without modifying the CRS and that S-ZK holds analogously to Pinocchio.

Finally, we consider the most efficient SNARK scheme by Groth [Gro16] and again show that the scheme is *already* subversion-zero-knowledge under SKE. Proving this is more involved than for the previous schemes, since the value τ , for which $P, \tau P, \tau^2 P, \dots$ are contained in the CRS does not suffice to simulate proofs, as for the previous schemes. We show that, using SKE twice, another value can be extracted, which together with τ then enables proof simulation. As corollaries, we get that S-WI holds unconditionally for all considered schemes.

CONCURRENT WORK. Campanelli et al. [CGGN17] show that Pinocchio as implemented in libsnark [BCG+14b] is not subversion-zero-knowledge by exhibiting an attack. As countermeasures they propose to instead use one of the older SNARKs by Gennaro et al. [GGPR13], as they allow verification of CRS well-formedness, which yields witness indistinguishability. They admit that for applications for which there is only *one* witness, like selling a Sudoku solution, WI is vacuous (as any protocol satisfies WI).

They refer to Bellare et al.’s [BFS16] S-ZK system and conjecture that “the techniques extend to the original QSP/QAP protocol in [GGPR13]” (which we proved rigorously). Moreover, “[i]t is however not clear if those techniques extend to Pinocchio” and “it would require major changes in the current implementation of ZKCP protocols”. (We show that it suffices to add 4 group elements to the CRS and perform the checks of well-formedness.) They recommend following the Zcash approach [BCG+15, BGG17] and using an interactive protocol that lets the prover and verifier compute the CRS together.

In other concurrent work Abdolmaleki et al. [ABLZ17] present a S-ZK variant of Groth’s SNARK [Gro16]. They need to modify the scheme, thereby reducing efficiency, and they prove their result under a stronger assumption. In particular, they extend the CRS by $2d$ group elements (where d is the number of multiplication gates in the circuit representing the relation). Their assumption states that any adversary that for generators $P_1 \in \mathbb{G}_1^*$ and $P_2 \in \mathbb{G}_2^*$ outputs a pair of the form (sP_1, sP_2) must know s . As they note, their assumption is false in groups with a symmetric (“Type-1”) bilinear map as well as in asymmetric groups of Type 2, whereas our SKE assumption holds generically in all bilinear group settings. They claim security of their scheme under their own definition of S-ZK, which is a statistical notion, in contrast to original computational S-ZK notion [BFS16], which we consider¹.

¹ It is not clear how their scheme can achieve statistical S-ZK, considering that the success of the simulator relies on a *computational* assumption. They also claim that their notion is stronger because they let the subverter X pass “extra information” to the adversary A , whereas A “only” receives X ’s coins r in [BFS16]. But A can itself compute any such information from r .

PRACTICAL IMPLICATIONS OF OUR RESULTS. We show that for all analyzed schemes except asymmetric Pinocchio, it suffices to verify the parameters once in order to guarantee subversion zero knowledge. Any already deployed parameters can thus be continued to be used after verification. Subversion-ZK of Pinocchio can be obtained by adding 4 group elements to the CRS.

For Pinocchio-based ZK contingent payments this means that the scheme can be made secure by slightly augmenting the size of the parameters and having the seller verify them. No additional interaction between seller and buyer (as recommended by Campanelli et al. [CGGN17]) is thus required. Of course, admitting additional interaction could lead to more efficient schemes than using the (costly) CRS verification.

The SNARK parameters used in Zcash have been computed by running the multi-party protocol from [BCG+15, BGG17] and verifiability of this process is achieved via random-oracle NIZK proofs. Let us define a CRS subverter that runs this protocol, playing the roles of all parties, and outputs the resulting CRS, including the ROM proofs. Since the latter guarantee CRS well-formedness, under SKE there exists an efficient extractor that can extract the simulation trapdoor from the subverter. Using the trapdoor, proofs can be simulated (as specified in Sect. 4). We thus conclude that, assuming that users verify the CRS and that the SKE assumption holds in the used bilinear group, Zcash provides a subversion-resistant form of anonymity in the random oracle model. Thus, even if all parties involved in creating the parameters were malicious, Zcash is still anonymous.

We content ourselves with the above argument, as a formal proof would be beyond the scope of this paper. Bowe et al. [BGG17] subsequently proved that their protocol is S-ZK with a polynomially small (not negligible) simulation error in the random-oracle model without making knowledge assumptions.

2 Definitions

2.1 Notation

If x is a (binary) string then $|x|$ is its length. If S is a finite set then $|S|$ denotes its size and $s \leftarrow S$ denotes picking an element uniformly from S and assigning it to s . We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

Algorithms are randomized unless otherwise indicated. “PT” stands for “polynomial time”, whether for randomized or deterministic algorithms. By $y \leftarrow A(x_1, \dots; r)$ we denote the operation of running algorithm A on inputs x_1, \dots and coins r and letting y denote the output. By $y \leftarrow_s A(x_1, \dots)$, we denote letting $y \leftarrow A(x_1, \dots; r)$ for random r . We denote by $[A(x_1, \dots)]$ the set of points that have positive probability of being output by A on inputs x_1, \dots .

For our security definitions we use the code-based game framework [BR06]. A game G (e.g. Fig. 1) usually depends on a scheme and executes one or more adversaries. It defines oracles for the adversaries as procedures. The game eventually returns a boolean. We let $\Pr[G]$ denote the probability that G returns true.

We recall the standard notions of soundness, knowledge-soundness, witness-indistinguishability and zero knowledge for NIZKs, which assume the CRS is trusted and then give their subversion-resistant counterparts that were introduced in [BFS16]. We mainly follow their exposition and start with the syntax.

2.2 NP Relations and NI Systems

NP RELATIONS. Consider $R: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{\text{true}, \text{false}\}$. For $x \in \{0, 1\}^*$ we let $R(x) = \{w \mid R(x, w) = \text{true}\}$ be the *witness set* of x . R is an **NP** relation if it is PT and there is a polynomial P_R such that every w in $R(x)$ has length at most $P_R(|x|)$ for all x . We let $L(R) = \{x \mid R(x) \neq \emptyset\}$ be the *language* associated to R . We will consider relations output by a PT *relation generator* Rg (which may also output some auxiliary information z that is given to the adversary). We assume λ can be deduced from $R \in [\text{Rg}(1^\lambda)]$ and note that definitions from [BFS16], which are for one *fixed* relation R , are easily recovered by defining $\text{Rg}(1^\lambda) := (1^\lambda, R)$.

NI SYSTEMS. A non-interactive (NI) system Π for relation generator Rg specifies the following PT algorithms. Via $\text{crs} \leftarrow_s \Pi.\text{Pg}(R)$ one generates a common reference string crs . Via $\pi \leftarrow_s \Pi.\text{P}(R, \text{crs}, x, w)$ the honest prover, given x and $w \in R(x)$, generates a proof π that $x \in L(R)$. Via $d \leftarrow \Pi.\text{V}(R, \text{crs}, x, \pi)$ a verifier can produce a decision $d \in \{\text{true}, \text{false}\}$ indicating whether π is a valid proof that $x \in L(R)$. We require (perfect) completeness, that is, for all $\lambda \in \mathbb{N}$, all $R \in [\text{Rg}(1^\lambda)]$, all $\text{crs} \in [\Pi.\text{Pg}(R)]$, all $x \in L(R)$, all $w \in R(x)$ and all $\pi \in [\Pi.\text{P}(R, \text{crs}, x, w)]$ we have $\Pi.\text{V}(R, \text{crs}, x, \pi) = \text{true}$. We also assume that $\Pi.\text{V}$ returns false if any of its arguments is \perp .

2.3 Standard Notions: SND, KSND, WI and ZK

SOUNDNESS. Soundness means that it is hard to create a valid proof for any $x \notin L(R)$. We consider computational soundness as opposed to a statistical one, which is the notion achieved by SNARGs.

Definition 1 (SND). An NI system Π for relation generator Rg is sound if $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{snd}}(\cdot)$ is negligible for all PT adversaries A , where $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{snd}}(\lambda) = \Pr[\text{SND}_{\Pi, \text{Rg}, \text{A}}(\lambda)]$ and game SND is specified in Fig. 1.

KNOWLEDGE SOUNDNESS. This strengthening of soundness [BG93] means that a prover that outputs a valid proof must know the witness. Formally, there exists an extractor that can extract the witness from the prover. The notion implies soundness, since for a proof of a wrong statement there exists no witness.

Definition 2 (KSND). An NI system Π for relation generator Rg is knowledge-sound if for all PT adversaries A there exists a PT extractor E such that $\text{Adv}_{\Pi, \text{Rg}, \text{A}, \text{E}}^{\text{ksnd}}(\cdot)$ is negligible, where $\text{Adv}_{\Pi, \text{Rg}, \text{A}, \text{E}}^{\text{ksnd}}(\lambda) = \Pr[\text{KSND}_{\Pi, \text{Rg}, \text{A}, \text{E}}(\lambda)]$ and game KSND is specified in Fig. 1.

Note that (as for the following two notions) the output of game KSND is *efficiently computable*, which is not the case for SND, since membership in $L(R)$ may not be efficiently decidable. This can be an issue when proving security of more complex systems that use a system Π as a building block.

WI. Witness-indistinguishability [FLS90] requires that proofs for the same statement using different witnesses are indistinguishable. The adversary can adaptively request multiple proofs for statements x under one of two witnesses w_0, w_1 ; it receives proofs under w_b for a challenge bit b which it must guess.

Definition 3 (WI). An NI system Π for relation generator Rg is witness-indistinguishable if $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{wi}}(\cdot)$ is negligible for all PT adversaries A , where $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{wi}}(\lambda) = 2 \Pr[\text{WI}_{\Pi, \text{Rg}, \text{A}}(\lambda)] - 1$ and game WI is specified in Fig. 1.

ZK. Zero knowledge [GMR89] means that no information apart from the fact that $x \in L(R)$ is leaked by the proof. It is formalized by requiring that a simulator, who can create the CRS, can compute proofs without being given a witness, so that CRS and proofs are indistinguishable from real ones. In particular, the distinguisher A can adaptively request proofs by supplying an instance and a valid witness for it. The proof is produced either by the honest prover using the witness, or by the simulator. The adversary outputs a guess b' .

Definition 4 (ZK). An NI system Π for Rg is zero-knowledge if Π specifies additional PT algorithms $\Pi.\text{Sim.crs}$ and $\Pi.\text{Sim.pf}$ such that $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{zk}}(\cdot)$ is negligible for all PT adversaries A , where $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{zk}}(\lambda) = 2 \Pr[\text{ZK}_{\Pi, \text{Rg}, \text{A}}(\lambda)] - 1$ and game ZK is specified in Fig. 1.

An NI system Π is *statistical* zero-knowledge if the above holds for all (not necessarily PT) adversaries A . It is *perfect* zero-knowledge if $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{zk}}(\cdot) \equiv 0$.

2.4 Notions for Subverted CRS: S-SND, S-KSND, S-WI and S-ZK

For all notions considered in the previous section the CRS is assumed to be honestly generated. Bellare et al. [BFS16] ask what happens when the CRS is maliciously generated and define subversion-resistant analogues S-SND, S-WI and S-ZK, in which the adversary chooses the CRS. The following three definitions are from [BFS16].

SUBVERSION SOUNDNESS. This asks that if the adversary creates a CRS in any way it likes, it is still unable to prove false statements under it. We accordingly modify the soundness game SND by letting the adversary choose crs in addition to x and π .

Definition 5 (S-SND). An NI system Π for generator Rg is subversion-sound if $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{s-snd}}(\cdot)$ is negligible for all PT adversaries A , where $\text{Adv}_{\Pi, \text{Rg}, \text{A}}^{\text{s-snd}}(\lambda) = \Pr[\text{S-SND}_{\Pi, \text{Rg}, \text{A}}(\lambda)]$ and game S-SND is specified in Fig. 1.

<p><u>GAME SND$_{\Pi, \text{Rg}, \text{A}}(\lambda)$</u> $R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $\text{crs} \leftarrow_{\\$} \Pi.\text{Pg}(R)$ $(x, \pi) \leftarrow_{\\$} \text{A}(R, \text{crs})$ Return $(x \notin L(R) \text{ and } \Pi.\text{V}(R, \text{crs}, x, \pi))$</p>	<p><u>GAME S-SND$_{\Pi, \text{Rg}, \text{A}}(\lambda)$</u> $R \leftarrow_{\\$} \text{Rg}(R)$ $(\text{crs}, x, \pi) \leftarrow_{\\$} \text{A}(R)$ Return $(x \notin L(R) \text{ and } \Pi.\text{V}(R, \text{crs}, x, \pi))$</p>
<p><u>GAME KSND$_{\Pi, \text{Rg}, \text{A}, \text{E}}(\lambda)$</u> $R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $\text{crs} \leftarrow_{\\$} \Pi.\text{Pg}(R) ; r \leftarrow_{\\$} \{0, 1\}^{\text{A.n}(\lambda)}$ $(x, \pi) \leftarrow \text{A}(R, \text{crs}; r)$ $w \leftarrow_{\\$} \text{E}(R, \text{crs}, r)$ Return $(R(x, w) = \text{false and } \Pi.\text{V}(R, \text{crs}, x, \pi))$</p>	<p><u>GAME S-KSND$_{\Pi, \text{Rg}, \text{A}, \text{E}}(\lambda)$</u> $R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $r \leftarrow_{\\$} \{0, 1\}^{\text{A.n}(\lambda)}$ $(\text{crs}, x, \pi) \leftarrow \text{A}(R; r)$ $w \leftarrow_{\\$} \text{E}(R, r)$ Return $(R(x, w) = \text{false and } \Pi.\text{V}(R, \text{crs}, x, \pi))$</p>
<p><u>GAME WI$_{\Pi, \text{Rg}, \text{A}}(\lambda)$</u> $b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $\text{crs} \leftarrow_{\\$} \Pi.\text{Pg}(R)$ $b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs})$ Return $(b = b')$</p> <p><u>PROVE(x, w_0, w_1)</u> If $R(x, w_0) = \text{false or } R(x, w_1) = \text{false}$ then return \perp $\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}, x, w_b)$ Return π</p>	<p><u>GAME S-WI$_{\Pi, \text{Rg}, \text{A}}(\lambda)$</u> $b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $(\text{crs}, st) \leftarrow_{\\$} \text{A}(R)$ $b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}, st)$ Return $(b = b')$</p> <p><u>PROVE(x, w_0, w_1)</u> If $R(x, w_0) = \text{false or } R(x, w_1) = \text{false}$ then return \perp $\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}, x, w_b)$ Return π</p>
<p><u>GAME ZK$_{\Pi, \text{Rg}, \text{A}}(\lambda)$</u> $b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $\text{crs}_1 \leftarrow_{\\$} \Pi.\text{Pg}(R)$ $(\text{crs}_0, td) \leftarrow_{\\$} \Pi.\text{Sim.crs}(R)$ $b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}_b)$ Return $(b = b')$</p> <p><u>PROVE(x, w)</u> If $R(x, w) = \text{false}$ then return \perp If $b = 1$ then $\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}_1, x, w)$ Else $\pi \leftarrow_{\\$} \Pi.\text{Sim.pf}(R, \text{crs}_0, td, x)$ Return π</p>	<p><u>GAME S-ZK$_{\Pi, \text{Rg}, \text{X}, \text{S}, \text{A}}(\lambda)$</u> $b \leftarrow_{\\$} \{0, 1\} ; R \leftarrow_{\\$} \text{Rg}(1^\lambda)$ $r_1 \leftarrow_{\\$} \{0, 1\}^{\text{X.n}(\lambda)} ; \text{crs}_1 \leftarrow \text{X}(R; r_1)$ $(\text{crs}_0, r_0, td) \leftarrow_{\\$} \text{S.crs}(R)$ $b' \leftarrow_{\\$} \text{A}^{\text{PROVE}}(R, \text{crs}_b, r_b)$ Return $(b = b')$</p> <p><u>PROVE(x, w)</u> If $R(x, w) = \text{false}$ then return \perp If $b = 1$ then $\pi \leftarrow_{\\$} \Pi.\text{P}(R, \text{crs}_1, x, w)$ Else $\pi \leftarrow_{\\$} \text{S.pf}(R, \text{crs}_0, td, x)$ Return π</p>

Fig. 1. Games defining soundness, knowledge-soundness, witness-indistinguishability and zero knowledge (left) and their subversion-resistant counterparts (right) for an NI system Π .

SUBVERSION WI. This notion demands that even when the subverter creates a CRS in any way it likes, it can still not decide which of two witnesses of its choice were used to create a proof. The adversary is modeled as a two-stage algorithm: it first outputs a CRS crs along with state information (which could e.g. contain a trapdoor associated to crs) passed to the second stage. The second stage is then defined like for the honest-CRS game WI, where via its PROVE oracle, the adversary can adaptively query proofs for instances under one of two witnesses.

Definition 6 (S-WI). *An NI system Π for generator Rg is subversion-witness-indistinguishable if $\text{Adv}_{\Pi, Rg, A}^{s-wi}(\cdot)$ is negligible for all PT adversaries A , where $\text{Adv}_{\Pi, Rg, A}^{s-wi}(\lambda) = 2 \Pr[S - WI_{\Pi, Rg, A}(\lambda)] - 1$ and game S - WI is specified in Fig. 1. An NI system Π is perfect S-WI if $\text{Adv}_{\Pi, Rg, A}^{s-wi}(\cdot) \equiv 0$.*

SUBVERSION ZK. This notion considers a CRS subverter X that returns an arbitrarily formed CRS. Subversion ZK now asks that for any such X there exists a simulator that is able to simulate (1) the full view of the CRS subverter, *including its coins*, and (2) proofs for adaptively chosen instances without knowing the witnesses. The simulator consists of $S.crs$, which returns a CRS, coins for X and a trapdoor which is then used by its second stage $S.pf$ to simulate proofs. The adversary's task is to decide whether it is given a real CRS and the coins used to produce it, and real proofs (case $b = 1$); or whether it is given a simulated CRS and coins, and simulated proofs (case $b = 0$).

Definition 7 (S-ZK). *An NI system Π for Rg is subversion-zero-knowledge if for all PT CRS subvertors X there exists a PT simulator $S = (S.crs, S.pf)$ such that for all PT adversaries A the function $\text{Adv}_{\Pi, Rg, X, S, A}^{s-zk}(\cdot)$ is negligible, where $\text{Adv}_{\Pi, Rg, X, S, A}^{s-zk}(\lambda) = 2 \Pr[S - ZK_{\Pi, Rg, X, S, A}(\lambda)] - 1$ and game S - ZK is specified in Fig. 1.*

The definition is akin to ZK for interactive proof systems [GMR89], when interpreting the CRS as the verifier's first message. The simulator must produce a full view of the verifier (including coins and a transcript of its interaction with the PROVE oracle). On the other hand, to imply ZK of NI systems, the simulator needs to produce the CRS *before* learning the statements for which it must simulate proofs. Moreover, the simulator can depend on X but not on A .

SUBVERSION KSND. For completeness we give a subversion-resistant analogue for knowledge soundness (not considered in [BFS16]), as this is the relevant notion for SNARKs. We modify game KSND and let the adversary choose crs in addition to x and π . We are not aware of any construction that achieves S-KSND and some form of WI.

Definition 8 (S-KSND). *An NI system Π for generator Rg is subversion-knowledge-sound if for all PT adversaries A there exists a PT extractor E such that $\text{Adv}_{\Pi, Rg, A, E}^{s-ksnd}(\cdot)$ is negligible, where $\text{Adv}_{\Pi, Rg, A, E}^{s-ksnd}(\lambda) = \Pr[S - KSND_{\Pi, Rg, A, E}(\lambda)]$ and game S - KSND is specified in Fig. 1.*

2.5 Bilinear Groups and Assumptions

BILINEAR GROUPS. The SNARK constructions we consider are based on bilinear groups, for which we introduce a new type of knowledge-of-exponent assumption. We distinguish between asymmetric and symmetric groups.

Definition 9. *An asymmetric-bilinear-group generator \mathbf{aGen} is a PT algorithm that takes input a security parameter 1^λ and outputs a description of a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e})$ with the following properties:*

- p is a prime of length λ ;
- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and (\mathbb{G}_T, \cdot) are groups of order p ;
- $\mathbf{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, that is, for all $a, b \in \mathbb{Z}_p$ and $S \in \mathbb{G}_1$, $T \in \mathbb{G}_2$ we have: $\mathbf{e}(aS, bT) = \mathbf{e}(S, T)^{ab}$;
- \mathbf{e} is non-degenerate, that is, for $P_1 \in \mathbb{G}_1^*$ and $P_2 \in \mathbb{G}_2^*$ (i.e., P_1 and P_2 are generators) $\mathbf{e}(P_1, P_2)$ generates \mathbb{G}_T .

Moreover, we assume that group operations and the bilinear map can be computed efficiently, membership of the groups and equality of group elements can be decided efficiently, and group generators can be sampled efficiently.

A symmetric-bilinear-group generator \mathbf{sGen} returns a bilinear group with $\mathbb{G}_1 = \mathbb{G}_2$, which we denote by \mathbb{G} , and with a symmetric non-degenerate bilinear map $\mathbf{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

ASSUMPTIONS. We recall the assumptions under which SNARKs in the literature were proven sound. The following assumptions are from [DFGK14], who adapted PDH from [Gro10] to asymmetric groups, and TSDH from [BB04, Gen04].

Definition 10 (q -PDH). *The $q(\lambda)$ -power Diffie-Hellman assumption holds for an asymmetric group generator \mathbf{aGen} if $\mathbf{Adv}_{q, \mathbf{aGen}, \mathbf{A}}^{\text{pdh}}(\cdot)$ is negligible for all PT \mathbf{A} , where $\mathbf{Adv}_{q, \mathbf{aGen}, \mathbf{A}}^{\text{pdh}}(\lambda) = \Pr[\text{PDH}_{q, \mathbf{aGen}, \mathbf{A}}(\lambda)]$ and PDH is defined in Fig. 2.*

The q -PDH assumption for symmetric group generators \mathbf{sGen} is defined analogously by letting $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ (\mathbf{A} thus only receives $2q$ group elements).

Definition 11 (q -TSDH). *The $q(\lambda)$ -target-group strong Diffie-Hellman assumption holds for an asymmetric group generator \mathbf{aGen} if $\mathbf{Adv}_{q, \mathbf{aGen}, \mathbf{A}}^{\text{tsdh}}(\cdot)$ is negligible for all PT adversaries \mathbf{A} , where $\mathbf{Adv}_{q, \mathbf{aGen}, \mathbf{A}}^{\text{tsdh}}(\lambda) = \Pr[\text{TSDH}_{q, \mathbf{aGen}, \mathbf{A}}(\lambda)]$ and TSDH is defined in Fig. 2.*

The q -TSDH assumption for symmetric group generators \mathbf{sGen} is defined analogously by letting $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ (\mathbf{A} thus only receives $q + 1$ group elements).

KEA. The knowledge-of-exponent assumption [Dam92, HT98, BP04] in a group \mathbb{G} states that an algorithm \mathbf{A} that is given two random generators $P, Q \in \mathbb{G}^*$ and outputs (cP, cQ) must know c . This is formalized by requiring that there exists

<p><u>GAME PDH_{q,aGen,A}(λ)</u></p> <p>$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathfrak{s} \mathbf{aGen}(1^\lambda) ; P_1 \leftarrow \mathfrak{s} \mathbb{G}_1^* ; P_2 \leftarrow \mathfrak{s} \mathbb{G}_2^* ; s \leftarrow \mathfrak{s} \mathbb{Z}_p$</p> <p>$Y \leftarrow \mathfrak{s} \mathbf{A}(Gr, P_1, P_2, sP_1, sP_2, \dots, s^q P_1, s^q P_2, s^{q+2} P_1, s^{q+2} P_2, \dots, s^{2q} P_1, s^{2q} P_2)$</p> <p>Return $(Y = s^{q+1} P_1)$</p>
<p><u>GAME TSDH_{q,aGen,A}(λ)</u></p> <p>$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathfrak{s} \mathbf{aGen}(1^\lambda) ; P_1 \leftarrow \mathfrak{s} \mathbb{G}_1^* ; P_2 \leftarrow \mathfrak{s} \mathbb{G}_2^* ; s \leftarrow \mathfrak{s} \mathbb{Z}_p$</p> <p>$(r, Y) \leftarrow \mathfrak{s} \mathbf{A}(Gr, P_1, P_2, sP_1, sP_2, \dots, s^q P_1, s^q P_2,)$</p> <p>Return $(r \in \mathbb{Z}_p \setminus \{s\} \text{ and } Y = \mathbf{e}(P_1, P_2)^{1/(s-r)})$</p>
<p><u>GAME PKE_{q,aGen,Z,A,E}(λ)</u></p> <p>$Gr = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathfrak{s} \mathbf{aGen}(1^\lambda) ; P_1 \leftarrow \mathfrak{s} \mathbb{G}_1^* ; P_2 \leftarrow \mathfrak{s} \mathbb{G}_2^* ; s \leftarrow \mathfrak{s} \mathbb{Z}_p$</p> <p>$r \leftarrow \mathfrak{s} \{0, 1\}^{\mathbf{A}.t(\lambda)}$</p> <p>$z \leftarrow \mathfrak{s} \mathbf{Z}(Gr, P_1, sP_1, \dots, s^q P_1)$</p> <p>$(V, W) \leftarrow \mathbf{A}(Gr, P_1, P_2, sP_1, sP_2, \dots, s^q P_1, s^q P_2, z; r)$</p> <p>$(a_0, \dots, a_q) \leftarrow \mathfrak{s} \mathbf{E}(Gr, P_1, P_2, sP_1, sP_2, \dots, s^q P_1, s^q P_2, z, r)$</p> <p>Return $(\mathbf{e}(V, P_2) = \mathbf{e}(P_1, W) \text{ and } V \neq (\sum_{i=0}^q a_i s^i) P_1)$</p>

Fig. 2. Games defining assumptions q -PDH, q -TSDH and q -PKE

an extractor for \mathbf{A} which given \mathbf{A} 's coins outputs c . This has been considered in the bilinear-group setting [AF07] where \mathbf{A} 's output (cP, cQ) can be verified by using the bilinear map. Generalizations of KEA were made by Groth [Gro10], who assumes that for every \mathbf{A} that on input $(P, Q, sP, sQ, s^2P, s^2Q, \dots, s^qP, s^qQ)$ returns (cP, cQ) an extractor can extract (a_0, \dots, a_q) such that $c = \sum_{i=0}^q a_i s^i$. Danezis et al. [DFGK14] port Groth's assumption to asymmetric groups as follows.

Definition 12 (q -PKE). *The $q(\lambda)$ -power knowledge of exponent assumption holds for \mathbf{aGen} w.r.t. the class \mathcal{Aux} of auxiliary input generators if for every PT $Z \in \mathcal{Aux}$ and PT adversary \mathbf{A} there exists a PT extractor \mathbf{E} s.t. $\mathbf{Adv}_{q,\mathbf{aGen},Z,\mathbf{A},\mathbf{E}}^{\text{pke}}(\cdot)$ is negligible, where $\mathbf{Adv}_{q,\mathbf{aGen},Z,\mathbf{A},\mathbf{E}}^{\text{pke}}(\lambda) = \Pr[\text{PKE}_{q,\mathbf{aGen},Z,\mathbf{A},\mathbf{E}}(\lambda)]$ and PKE is defined in Fig. 2.*

The q -PKE assumption for symmetric generators \mathbf{sGen} is defined by letting $\mathbb{G}_1 = \mathbb{G}_2$ but again choosing $P_1, P_2 \leftarrow \mathfrak{s} \mathbb{G}^*$ (\mathbf{A} thus again receives $2q + 2$ group elements).

Bellare et al. [BFS16] consider deterministically generated groups (whereas for SNARK systems the group will be part of the relation R output by a relation generator \mathbf{Rg}). They therefore need to define all other assumptions, such as DLin [BBS04], with respect to this fixed group. BFS introduce a new type of KEA, called DH-KEA, which assumes that if \mathbf{A} outputs a Diffie-Hellman (DH) tuple (sP, tP, stP) w.r.t. the fixed P , then \mathbf{A} must know either s or t . The auxiliary input given to \mathbf{A} are two additional random generators H_0, H_1 . The intuition is

that while an adversary may produce one group element without knowing its discrete logarithm by hashing into the elliptic curve [BF01, SvdW06, BCI+10], it seems hard to produce a DH tuple without knowing at least one of the logarithms.

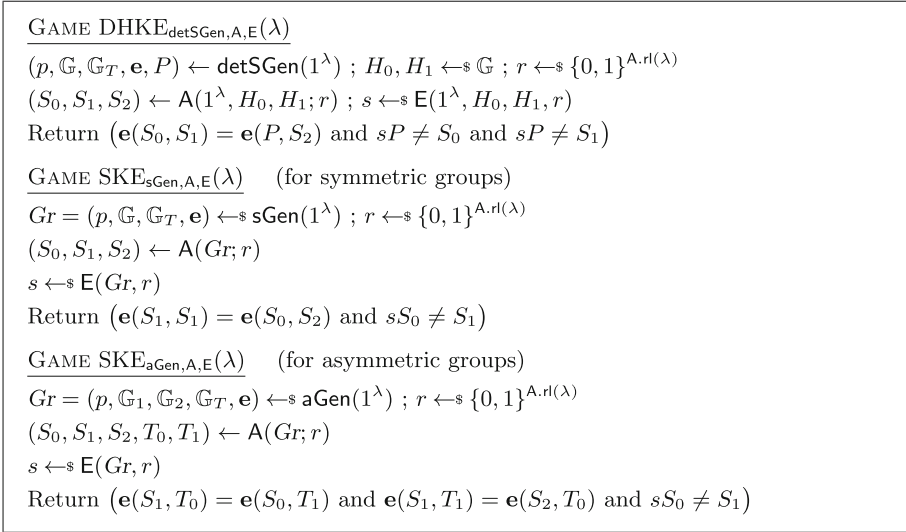


Fig. 3. Games defining knowledge-of-exponent assumptions

Definition 13 (DH-KEA). Let detSGen be a deterministic group generator; let $\text{Adv}_{\text{detSGen,A,E}}^{\text{dhke}}(\lambda) = \Pr[\text{DHKE}_{\text{detSGen,A,E}}(\lambda)]$, with game DHKE defined in Fig. 3. The Diffie-Hellman knowledge of exponent assumption holds for detSGen if for every PT A there exists a PT E s.t. $\text{Adv}_{\text{detSGen,A,E}}^{\text{dhke}}(\cdot)$ is negligible.

SKE. We now consider a weakening of DH-KEA where we prescribe $s = t$; that is, if A on input P outputs a pair (sP, s^2P) then E extracts s . This assumption is weaker than (i.e., implied by) DH-KEA. As we consider groups with randomly sampled generators, we let A choose the generator P itself and assume that there exists an extractor that extracts s when A outputs a tuple (P, sP, s^2P) . This allows us to choose a random generator when setting up parameters of a scheme. The security of such schemes then follows from assumptions such as PDH, as defined above, where the generators are chosen randomly.

Definition 14 (SKE). Let sGen be a symmetric-group generator and define $\text{Adv}_{\text{sGen,A,E}}^{\text{ske}}(\lambda) = \Pr[\text{SKE}_{\text{sGen,A,E}}(\lambda)]$, where game SKE is defined in Fig. 3. The square knowledge of exponent assumption holds for sGen if for every PT A there exists a PT E s.t. $\text{Adv}_{\text{sGen,A,E}}^{\text{ske}}(\cdot)$ is negligible.

SKE FOR ASYMMETRIC GROUPS. For asymmetric bilinear-group generators, we make assumption SKE in the first source group \mathbb{G}_1 . Unlike for symmetric groups,

a tuple $(S_0, sS_0, s^2S_0) \in \mathbb{G}_1^3$ is not verifiable via an asymmetric pairing. To make it verifiable, we *weaken* the assumption and require \mathbf{A} to additionally output a \mathbb{G}_2 -element T_0 as well as $T_1 = sT_0$, which enables verification (as done in game SKE_{aGen}).

Definition 15 (SKE). *Let aGen be an asymmetric-group generator and define $\text{Adv}_{\text{aGen}, \mathbf{A}, \mathbf{E}}^{\text{ske}}(\lambda) = \Pr[\text{SKE}_{\text{aGen}, \mathbf{A}, \mathbf{E}}(\lambda)]$, where game SKE is defined in Fig. 3. The SKE assumption holds for aGen in the first source group if for every PT \mathbf{A} there exists a PT \mathbf{E} s.t. $\text{Adv}_{\text{aGen}, \mathbf{A}, \mathbf{E}}^{\text{ske}}(\cdot)$ is negligible.*

We note that in addition to verifiability these additional elements T_0 and T_1 actually add to the plausibility of the assumption for asymmetric groups. Even if outputting S_2 was not required, one could argue that the following *stronger* assumption holds in Type-3 bilinear groups, in which DDH holds in \mathbb{G}_1 and in \mathbb{G}_2 : it is hard to compute $(S_0, S_1, T_0, T_1) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ with $\mathbf{e}(S_1, T_0) = \mathbf{e}(S_0, T_1)$ without knowing the logarithms of S_1 to base S_0 (or equivalently T_1 to base T_0):² an adversary might choose S_0 and S_1 obliviously by hashing into the group; but if it was able to compute from them the respective T_0 and T_1 then this would break DDH in \mathbb{G}_1 . (Given a DDH challenge $(S_0, S_1 = s_1S_0, S_2 = s_2S_0, R)$, compute T_0 and T_1 as above; then we have $R = s_1s_2S_0$ iff $\mathbf{e}(R, T_0) = \mathbf{e}(S_2, T_1)$.) Of course, this argument breaks down if there is an efficiently computable homomorphism from \mathbb{G}_1 to \mathbb{G}_2 or vice versa.

Finally, we note that q -PKE with $q = 0$ does not imply SKE, since a PKE adversary must return (V, W) which is a multiple of the received (P_1, P_2) , while an SKE adversary can choose the “basis” (S_0, T_0) itself. The converse does not hold either (SKE $\not\Rightarrow$ PKE), since an SKE adversary must return $S_2 = s^2S_0$.

2.6 SKE in the Generic-Group Model

We show that SKE holds in the generic-group model. We show it for symmetric generic groups, which implies the result for asymmetric groups (where the adversary has less power). As [BFS16] did for DH-KEA, we reflect hashing into elliptic curves by providing the adversary with an additional generic operation: it can create new group elements without knowing their discrete logarithms (which are not known to the extractor either).

Theorem 16. *SKE, as defined in Definition 14, holds in the generic-group model with hashing into the group.*

In the proof we will use the following lemma, which we prove first.

Lemma 17. *Let \mathbb{F} be a field and let $A, B, C \in \mathbb{F}[X_1, \dots, X_k]$, with degree of A, B and C at most 1. If $A \cdot C = B^2$ then for some $s \in \mathbb{F}$: $B = s \cdot A$.*

² When fixing the generators S_0 and T_0 , this corresponds to the assumption made by Abdolmaleki et al. [ABLZ17] to show S-ZK of their SNARK.

Proof. Let $\alpha_i, \beta_i, \gamma_i$, for $0 \leq i \leq k$, denote the coefficients of X_i (where $X_0 := 1$) in A, B, C , respectively. If $A = 0$ then $B = 0$ and the theorem follows. Assume thus $A \neq 0$; Define $j := \min\{i \in [0, k] : \alpha_j \neq 0\}$ and $s := \beta_j \cdot \alpha_j^{-1}$.

To prove the lemma, we will now show that for all $i \in [0, k]$:

$$\beta_i = s \cdot \alpha_i. \tag{1}$$

From $A \cdot C = B^2$ we have

$$L(\vec{X}) := (\beta_0 + \sum_{i=1}^k \beta_i X_i)^2 - (\alpha_0 + \sum_{i=1}^k \alpha_i X_i)(\gamma_0 + \sum_{i=1}^k \gamma_i X_i) = 0. \tag{2}$$

From $L(0, \dots, 0) = 0$, we get: (I) $\beta_0^2 = \alpha_0 \gamma_0$, which implies that Eq. (1) holds for $i = 0$: either $\alpha_0 = 0$, then from (I): $\beta_0 = 0$; or $\alpha_0 \neq 0$, then $j = 0$ and Eq. (1) holds as well.

Let now $i \in [1, k]$ be arbitrarily fixed and let e_i denote the vector $(0, \dots, 0, 1, 0, \dots, 0)$ with 1 at position i . Consider $L(e_i) = 0$, which together with (I) yields

$$2\beta_0\beta_i + \beta_i^2 - \alpha_0\gamma_i - \alpha_i\gamma_0 - \alpha_i\gamma_i = 0. \tag{3}$$

Similarly, from $L(2e_i) = 0$, we have $4\beta_0\beta_i + 4\beta_i^2 - 2\alpha_0\gamma_i - 2\alpha_i\gamma_0 - 4\alpha_i\gamma_i = 0$, which after subtracting Eq. (3) twice yields: (II) $\beta_i^2 = \alpha_i\gamma_i$. If $\alpha_i = 0$ then $\beta_i = 0$, which shows Eq. (1). For the remainder let us assume $\alpha_i \neq 0$.

Plugging (II) into Eq. (3) yields: (III) $2\beta_0\beta_i = \alpha_0\gamma_i - \alpha_i\gamma_0$.

If $\alpha_0 \neq 0$ then $j = 0$ and plugging (I) and (II) into (III) yields

$$2\beta_0\beta_i - \alpha_0\alpha_i^{-1}\beta_i^2 - \alpha_i\alpha_0^{-1}\beta_0^2 = 0.$$

Solving for β_i yields the unique solution $\beta_i = \beta_0\alpha_0^{-1}\alpha_i$, which shows Eq. (1) for the case $\alpha_0 \neq 0$.

Let us now assume $\alpha_0 = 0$. By (I) we have $\beta_0 = 0$. If $i = j$ then Eq. (1) holds by definition of s . Assume $i \neq j$. From $L(e_i + e_j)$ we have (since $\alpha_0 = \beta_0 = 0$):

$$0 = \beta_i^2 + \beta_j^2 + 2\beta_i\beta_j - \alpha_i\gamma_0 - \alpha_i\gamma_i - \alpha_i\gamma_j - \alpha_j\gamma_0 - \alpha_j\gamma_i - \alpha_j\gamma_j = 2\beta_i\beta_j - \alpha_i\gamma_j - \alpha_j\gamma_i,$$

where we used (II) and $\alpha_i\gamma_0 = \alpha_j\gamma_0 = 0$ (which follows from (III) and $\alpha_0 = \beta_0 = 0$). Together with (II) the latter yields

$$2\beta_i\beta_j - \alpha_i\alpha_j^{-1}\beta_j^2 - \alpha_j\alpha_i^{-1}\beta_i^2 = 0.$$

Solving for β_i yields the unique solution $\beta_i = \beta_j\alpha_j^{-1}\alpha_i$, which concludes the proof. □

Proof (of Theorem 16). In the “traditional” generic-group model, group elements are represented by random strings and an adversary A only has access to operations on them (addition of elements in \mathbb{G} , multiplication of elements in \mathbb{G}_T and pairing of elements in \mathbb{G}) via oracles. In particular, A can only produce new \mathbb{G} elements by adding received elements.

We also need to reflect the fact that by “hashing into the group”, A can create a new group element *without knowing its discrete logarithm w.r.t. one of the received elements*. We extend the generic-group model and provide the adversary with an additional operation, namely to request a new group element “independently of the received ones”. (And neither the adversary nor the extractor we construct knows its discrete logarithm.)

For SKE the adversary A receives the group element P and needs to output (S_0, S_1, S_2) where for some s, t : $S_0 = tP$, $S_1 = sS_0 = stP$ and $S_2 = s^2S_0 = s^2tP$. The adversary can produce these group elements by combining the received generator P with newly generated (“hashed”) group elements that it has requested. We represent the latter as x_iP , for $i = 1, \dots, k$, for some k . The extractor keeps track of the group operations performed by A and thus knows

$$\alpha_0, \dots, \alpha_k, \beta_0, \dots, \beta_k, \gamma_0, \dots, \gamma_k \in \mathbb{Z}_p \tag{4}$$

such that A 's output (S_0, S_1, S_2) is of the form

$$\begin{aligned} S_0 &= \alpha_0P + \sum_{i=1}^k \alpha_i(x_iP) & S_1 &= \beta_0P + \sum_{i=1}^k \beta_i(x_iP) \\ S_2 &= \gamma_0P + \sum_{i=1}^k \gamma_i(x_iP) \end{aligned}$$

Note that the extractor does however not know $x := (x_1, \dots, x_k)$.

Assume the adversary wins and $e(S_1, S_1) = e(S_0, S_2)$. Taking the logarithms of the latter yields

$$\left(\beta_0 + \sum_{i=1}^k \beta_i x_i\right)^2 - \left(\alpha_0 + \sum_{i=1}^k \alpha_i x_i\right)\left(\gamma_0 + \sum_{i=1}^k \gamma_i x_i\right) = 0. \tag{5}$$

Since the adversary has no information about x_1, \dots, x_k (except for a negligible information leak by comparing group elements, which we ignore), the values in Eq. (4) are generated independently of x_1, \dots, x_k . By the Schwartz-Zippel lemma the probability that Eq. (5) holds when x_1, \dots, x_k are randomly chosen is negligible, except if the left-hand side corresponds to the zero polynomial. With overwhelming probability we thus have

$$B(\vec{X})^2 - A(\vec{X}) \cdot C(\vec{X}) = 0 \quad \text{with}$$

$$A(\vec{X}) = \alpha_0 + \sum_{i=1}^k \alpha_i X_i \quad B(\vec{X}) = \beta_0 + \sum_{i=1}^k \beta_i X_i \quad C(\vec{X}) = \gamma_0 + \sum_{i=1}^k \gamma_i X_i$$

By Lemma 17 we have that $B = sA$ for some $s \in \mathbb{F}$. The extractor computes and returns s , which is correct since $S_1 = B(\vec{x})P = sA(\vec{x})P = sS_0$. \square

3 SNARKs

We start with a formal definition of SNARGs and SNARKs.

Definition 18 (SNARG). *An NI system $\Pi = (\Pi.Pg, \Pi.P, \Pi.V)$ is a succinct non-interactive argument for relation generator Rg if it is complete and sound, as in Definition 1, and moreover succinct, meaning that for all $\lambda \in \mathbb{N}$,*

all $R \in [\text{Rg}(1^\lambda)]$, all $\text{crs} \in [\Pi.\text{Pg}(R)]$, all $x \in L(R)$, all $w \in R(x)$ and all $\pi \in [\Pi.\text{P}(1^\lambda, \text{crs}, x, w)]$ we have $|\pi| = \text{poly}(\lambda)$ and $\Pi.\text{V}(1^\lambda, \text{crs}, x, \pi)$ runs in time $\text{poly}(\lambda + |x|)$.

Definition 19 (SNARK). A SNARG Π is a succinct non-interactive argument of knowledge if it satisfies knowledge soundness, as in Definition 2.

Gennaro et al. [GGPR13] base their SNARK constructions on quadratic programs. In particular, they show how to convert any boolean circuit into a quadratic span program and any arithmetic circuit into a quadratic arithmetic program (QAP).

Definition 20 (QAP). A quadratic arithmetic program over a field \mathbb{F} is a tuple

$$(\mathbb{F}, n, \{A_i(X), B_i(X), C_i(X)\}_{i=0}^m, Z(X)),$$

where $A_i(X), B_i(X), C_i(X), Z(X) \in \mathbb{F}[X]$, which define a language of statements $(s_1, \dots, s_n) \in \mathbb{F}^n$ and witnesses $(s_{n+1}, \dots, s_m) \in \mathbb{F}^{m-n}$ such that

$$\begin{aligned} & \left(A_0(X) + \sum_{i=1}^m s_i A_i(X) \right) \cdot \left(B_0(X) + \sum_{i=1}^m s_i B_i(X) \right) \\ & = C_0(X) + \sum_{i=1}^m s_i C_i(X) + H(X) \cdot Z(X), \end{aligned} \quad (6)$$

for some degree- $(d-2)$ quotient polynomial $H(X)$, where d is the degree of $Z(X)$ (we assume the degrees of all $A_i(X), B_i(X), C_i(X)$ are at most $d-1$).

All of the discussed SNARK constructions are for QAPs defined over a bilinear group. We will thus consider relation generators Rg of the following form:

Definition 21 (QAP relation). A QAP relation generator Rg is a PT algorithm that on input 1^λ returns a relation description of the following form:

$$\begin{aligned} R = (\text{Gr}, n, \vec{A}, \vec{B}, \vec{C}, Z) \quad & \text{where Gr is a bilinear group whose order } p \\ & \text{defines } \mathbb{F} := \mathbb{Z}_p \text{ and} \\ & \vec{A}, \vec{B}, \vec{C} \in (\mathbb{F}^{(d-1)}[X])^{(m+1)}, \quad Z \in \mathbb{F}^{(d)}[X], \quad n \leq m. \end{aligned} \quad (7)$$

For $x \in \mathbb{F}^n$ and $w \in \mathbb{F}^{m-n}$ we define $R(x, w) = \text{true}$ iff there exists $H(X) \in \mathbb{F}[X]$ so that Eq. (6) holds for $s := x \parallel w$ (where “ \parallel ” denotes concatenation).

4 Asymmetric Pinocchio

The CRS of SNARKs systems is usually split into a (long) part pk , used to compute proofs, and a (short) part vk , used to verify them. Pinocchio [PHGR13] is one of the central SNARK systems. Ben-Sasson et al. [BCTV14] proposed a

variant in asymmetric groups for which they also shorten the verification key. Their system is implemented in libsnark [BCG+14b] and underlies Zcash.

Campanelli et al. [CGGN17] show that the protocol is not subversion-zero-knowledge and expect major changes to make it secure. In the following we show that by adding merely 4 group elements to the CRS (which we denote by ck for “checking key”), we can enable verification of well-formedness of (vk, pk) by using the pairing available in the bilinear group. We then show that under SKE (Definition 15), our modification of the scheme from [BCTV14] achieves subversion zero knowledge. The protocol is given in Fig. 4, where we underlined our modifications. Below we define procedure CRS VERIFICATION, which a prover runs on a CRS before using it the first time.

Theorem 22 ([PHGR13, BCTV14]). *Let Rg be a relation generator that on input 1^λ returns a QAP of degree at most $d(\lambda)$ over Gr . Define $aGen$ that returns the first output Gr of Rg and let $q := 4d + 4$. If the q -PDH, the q -PKE and the $2q$ -SDH assumptions hold for $aGen$ then the scheme in Fig. 4 without including ck in the CRS is knowledge-sound. Moreover, it is statistical zero-knowledge.*

Inspecting the proof of the theorem in [PHGR13], it is easily seen that the additional elements contained in ck can be produced by the reduction. Moreover, knowledge soundness is independent of the prove algorithm $\Pi.P$, and a correctly generated CRS passes CRS verification. This yields the following.

Corollary 23 (to Theorem 22). *Let Rg and $aGen$ be as in Theorem 22. If the q -PDH, the q -PKE and the $2q$ -SDH assumptions hold for $aGen$ for $q := 4d + 4$ then the scheme in Fig. 4 is knowledge-sound statistical zero-knowledge.*

CRS VERIFICATION. On input (R, vk, pk, ck) , let $\{a_{i,j}\}, \{b_{i,j}\}, \{c_{i,j}\}, \{z_k\}$ denote the coefficients of polynomials $A_i(X), B_i(X), C_i(X)$ and $Z(X)$, respectively, for $0 \leq i \leq m$ and $0 \leq j \leq d - 1$ and $0 \leq k \leq d$.

1. Check $P_1 \neq 0_{G_1}$ and $P_2 \neq 0_{G_2}$.
2. Check correct choice of secret values: $ck_A \neq 0_{G_2}, ck_B \neq 0_{G_2}, vk_\gamma \neq 0_{G_2}, vk_{\beta\gamma} \neq 0_{G_1}$ and $vk_Z \neq 0_{G_2}$.
3. Check consistency of pk_H : Check $pk_{H,0} = P_1$; and for $i = 1, \dots, d$:

$$e(pk_{H,i}, P_2) = e(pk_{H,i-1}, ck_H)$$

4. Check consistency of pk_A, pk'_A, pk_B, pk'_B : for $i = 0, \dots, m + 3$:

$$e(pk_{A,i}, P_2) = e\left(\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, ck_A\right) \quad e(pk'_{A,i}, P_2) = e(pk_{A,i}, vk_A)$$

$$e(P_1, pk_{B,i}) = e\left(\sum_{j=0}^{d-1} b_{i,j} pk_{H,j}, ck_B\right) \quad e(pk'_{B,i}, P_2) = e(vk_B, pk_{B,i})$$

5. Check consistency of ck_C : $e(pk_{A,m+1}, ck_B) = e\left(\sum_{j=0}^d z_j pk_{H,j}, ck_C\right)$ (Note that for an honest CRS we have $pk_{A,m+1} = Z(\tau)\rho_A P_1 \neq 0$.)

KEY GENERATION. On input R as in Eq. (7) representing a QAP for an asymmetric group Gr do the following:

1. Sample $P_1 \leftarrow \mathbb{G}_1^*$ and $P_2 \leftarrow \mathbb{G}_2^*$
2. Set $\begin{bmatrix} A_{m+1} & B_{m+1} & C_{m+1} \\ A_{m+2} & B_{m+2} & C_{m+2} \\ A_{m+3} & B_{m+3} & C_{m+3} \end{bmatrix} := \begin{bmatrix} Z & 0 & 0 \\ 0 & Z & 0 \\ 0 & 0 & Z \end{bmatrix}$
3. Sample $\rho_A, \rho_B, \beta, \gamma \leftarrow \mathbb{F}^*$ and $\tau, \alpha_A, \alpha_B, \alpha_C \leftarrow \mathbb{F}$, conditioned on $Z(\tau) \neq 0$.
4. Set $vk = (P_1, P_2, vk_A, vk_B, vk_C, vk_\gamma, \widehat{vk}_{\beta\gamma}, vk_Z, vk_{IC})$, where

$$\begin{aligned} vk_A &:= \alpha_A P_2 & vk_B &:= \alpha_B P_1 & vk_C &:= \alpha_C P_2 \\ vk_\gamma &:= \gamma P_2 & vk_{\beta\gamma} &:= \gamma\beta P_1 & \widehat{vk}_{\beta\gamma} &:= \gamma\beta P_2 \\ vk_Z &:= Z(\tau)\rho_A\rho_B P_2 & \{vk_{IC,i}\}_{i=0}^n &:= \{A_i(\tau)\rho_A P_1\}_{i=0}^n \end{aligned}$$
5. Set $pk = (pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K, pk_H)$ where

$$\begin{aligned} \text{for } i = 0, \dots, m+3: \quad & pk_{A,i} := A_i(\tau)\rho_A P_1 & pk'_{A,i} &:= A_i(\tau)\alpha_A\rho_A P_1 \\ & pk_{B,i} := B_i(\tau)\rho_B P_2 & pk'_{B,i} &:= B_i(\tau)\alpha_B\rho_B P_1 \\ & pk_{C,i} := C_i(\tau)\rho_A\rho_B P_1 & pk'_{C,i} &:= C_i(\tau)\alpha_C\rho_A\rho_B P_1 \\ & pk_{K,i} := \beta(A_i(\tau)\rho_A + B_i(\tau)\rho_B + C_i(\tau)\rho_A\rho_B)P_1 \end{aligned}$$

$$\text{for } i = 0, \dots, d: \quad pk_{H,i} := \tau^i P_1$$
6. **Set** $ck := (ck_A, ck_B, ck_C, ck_H)$ **where**

$$ck_A := \rho_A P_2 \quad ck_B := \rho_B P_2 \quad ck_C := \rho_A\rho_B P_2 \quad ck_H := \tau P_2$$
7. Return $crs := (vk, pk, ck)$.

PROVE. On input $R, (vk, pk, ck)$ and $\vec{s} \in \mathbb{F}^m$ s.t. Eq. (6) is satisfied for some $H' \in \mathbb{F}[X]$.

1. **If** (R, vk, pk, ck) **does not pass CRS VERIFICATION then return** \perp .
2. Sample $\delta_A, \delta_B, \delta_C \leftarrow \mathbb{F}$ and define

$$\begin{aligned} A(X) &:= A_0(X) + \sum_{i=1}^m s_i A_i(X) + \delta_A Z(X) \\ B(X) &:= B_0(X) + \sum_{i=1}^m s_i B_i(X) + \delta_B Z(X) \\ C(X) &:= C_0(X) + \sum_{i=1}^m s_i C_i(X) + \delta_C Z(X) \end{aligned}$$
3. Compute $H(X)$ with coefficients (h_0, \dots, h_d) s.t. $A(X)B(X) - C(X) = H(X)Z(X)$.
4. Define $\widetilde{pk}_{A,i} := \mathbb{1}_{i>n} pk_{A,i}$ (that is, $\widetilde{pk}_{A,i} = 0$ for $0 \leq i \leq n$ and $= pk_{A,i}$ otherwise)
 Define $\widetilde{pk}'_{A,i} := \mathbb{1}_{i>n} pk'_{A,i}$
5. Let $\vec{c} := \mathbb{1} \parallel \vec{s} \parallel \delta_A \parallel \delta_B \parallel \delta_C \in \mathbb{F}^{m+4}$ and compute (" $\langle \cdot, \cdot \rangle$ " denotes the scalar product)

$$\begin{aligned} \pi_A &:= \langle \vec{c}, \widetilde{pk}_A \rangle & \pi'_A &:= \langle \vec{c}, \widetilde{pk}'_A \rangle & \pi_B &:= \langle \vec{c}, pk_B \rangle & \pi'_B &:= \langle \vec{c}, pk'_B \rangle \\ \pi_C &:= \langle \vec{c}, pk_C \rangle & \pi'_C &:= \langle \vec{c}, pk'_C \rangle & \pi_K &:= \langle \vec{c}, pk_K \rangle & \pi_H &:= \langle \vec{h}, pk_H \rangle \end{aligned}$$
6. Return $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$.

VERIFY. On input $R, vk, \vec{x} \in \mathbb{F}^n$ and proof $\pi \in \mathbb{G}_1^7 \times \mathbb{G}_2$.

1. Compute $vk_x := vk_{IC,0} + \sum_{i=1}^n x_i vk_{IC,i}$.
2. Check validity of π'_A, π'_B , and π'_C :

$$\mathbf{e}(\pi'_A, P_2) = \mathbf{e}(\pi_A, vk_A) \quad \mathbf{e}(\pi'_B, P_2) = \mathbf{e}(vk_B, \pi_B) \quad \mathbf{e}(\pi'_C, P_2) = \mathbf{e}(\pi_C, vk_C)$$
3. Check same coefficients were used via π_K :

$$\mathbf{e}(\pi_K, vk_\gamma) = \mathbf{e}(vk_x + \pi_A + \pi_C, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, \pi_B)$$
4. Check QAP is satisfied via π_H : $\mathbf{e}(vk_x + \pi_A, \pi_B) = \mathbf{e}(\pi_H, vk_Z) \cdot \mathbf{e}(\pi_C, P_2)$
5. If all checks in 2–4. succeeded then return **true** and otherwise **false**.

Fig. 4. S-ZK Asymmetric Pinocchio, adapted from [BCTV14].

6. Check consistency of vk : for $i = 0, \dots, n$: $vk_{IC,i} = pk_{A,i}$ and

$$\mathbf{e}(vk_{\beta\gamma}, P_2) = \mathbf{e}(P_1, \widehat{vk}_{\beta\gamma}) \quad \mathbf{e}(P_1, vk_Z) = \mathbf{e}\left(\sum_{j=0}^d z_j pk_{H,j}, ck_C\right)$$

7. Check consistency of pk_C, pk'_C, pk_K : for $i = 0, \dots, m + 3$:

$$\begin{aligned} \mathbf{e}(pk_{C,i}, P_2) &= \mathbf{e}\left(\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, ck_C\right) & \mathbf{e}(pk'_{C,i}, P_2) &= \mathbf{e}(pk_{C,i}, vk_C) \\ \mathbf{e}(pk_{K,i}, vk_\gamma) &= \mathbf{e}(pk_{A,i} + pk_{C,i}, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, pk_{B,i}) \end{aligned}$$

8. If all checks in 1.-7. succeeded then return **true** and otherwise **false**.

Remark 24. The condition that in **KEY GENERATION** $\rho_A, \rho_B, \beta, \gamma$ and $Z(\tau)$ must be non-zero is not made explicit in [BCTV14]. However if $\gamma = 0$ then any π_K satisfies the verification equation in 3; and if $\beta = 0$ and $\gamma \neq 0$ then no π_K satisfies it. If $Z(\tau) = 0$ or $\rho_A = 0$ or $\rho_B = 0$ then $vk_Z = 0_{\mathbb{G}_2}$ and setting π_B and π_C to zero always satisfies the equation in 4 in verification.

CRS Verifiability. We show that a CRS (vk, pk, ck) that passes verification is constructed as in **KEY GENERATION**; in particular, there exist $\tau, \alpha_A, \alpha_B, \alpha_C \in \mathbb{F}$ and $\rho_A, \rho_B, \beta, \gamma, \in \mathbb{F}^*$ such that (vk, pk, ck) is computed as in **KEY GENERATION**. Let $\tau, \alpha_A, \alpha_B, \alpha_C, \rho_A, \rho_B, \gamma, \xi \in \mathbb{F}$ be the values defined by the logarithms of the elements $ck_H, vk_A, vk_B, vk_C, ck_A, ck_B, vk_\gamma$ and $vk_{\beta\gamma}$, respectively. Check 2. ensures that $\rho_A, \rho_B, \gamma, \xi$ and $Z(\tau)$ are all non-zero. Set $\beta := \xi\gamma^{-1} \neq 0$.

Check 3. ensures that pk_H is correctly computed w.r.t. τ . Check 4. ensures that pk_A, pk'_A, pk_B and pk'_B are correctly computed w.r.t. $\tau, \rho_A, \rho_B, \alpha_A$ and α_B . Check 5. ensures that pk_C is correctly computed: since by 4., $pk_{A,m+1} = Z(\tau)\rho_A P_1$ and $Z(\tau) \neq 0$, we have $ck_C = \rho_A \rho_B P_2$. Check 6. ensures that $\widehat{vk}_{\beta\gamma}$ and vk_Z are correctly computed and Check 7. does the same for pk_C, pk'_C and pk_K .

This shows that with respect to $ck_H, vk_A, vk_B, vk_C, ck_A, ck_B, vk_\gamma$ and $vk_{\beta\gamma}$ (which implicitly define the randomness used in a CRS), all remaining elements $pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K, pk_H$, as well as $\widehat{vk}_{\beta\gamma}, vk_Z, vk_{IC}$ and ck_C are defined as prescribed by **KEY GENERATION**.

Trapdoor Extraction. In order to prove subversion zero knowledge, we now show how to construct a simulator $(\Pi.\text{Sim.crs}, \Pi.\text{Sim.pf})$ for a CRS subverter X . Let X be a CRS subverter that outputs (vk, pk, ck) . Define $\mathsf{X}'(1^\lambda; r)$ that runs $(vk, pk, ck) \leftarrow \mathsf{X}(1^\lambda; r)$, parses the output as above and returns $(pk_{H,0}, pk_{H,1}, pk_{H,2}, P_2, ck_H)$. By **SKE for aGen** (Definition 15) there exists a PT algorithm $\mathsf{E}_{\mathsf{X}'}$ such that if for some $\tau \in \mathbb{F}$: $pk_{H,1} = \tau pk_{H,0}, pk_{H,2} = \tau^2 pk_{H,0}$ and $ck_H = \tau P_2$ then with overwhelming probability $\mathsf{E}_{\mathsf{X}'}$ extracts τ . Using $\mathsf{E}_{\mathsf{X}'}$ we define the CRS simulator $\mathsf{S.crs}$ which computes (crs, r, td) as follows: On input 1^λ :

1. Sample randomness for X : $r \leftarrow_{\mathfrak{s}} \{0, 1\}^{X.r(\lambda)}$.
2. Run $(vk, pk, ck) \leftarrow X(1^\lambda; r)$.
3. If (R, vk, pk, ck) passes CRS VERIFICATION then $\tau \leftarrow_{\mathfrak{s}} \text{Ex}'(1^\lambda, r)$; else $\tau \leftarrow \perp$.
4. Return $((vk, pk, ck), r, \tau)$.

Proof Simulation. Given (vk, pk, ck) , trapdoor τ and a statement $x \in \mathbb{F}^n$, the proof simulator S_{pf} is defined as follows:

1. If $\tau = \perp$ then return \perp .
2. Use τ to compute $Z(\tau)$ (which in a verified CRS is non-zero). Compute the following “simulation keys”:

$$\begin{aligned}
 sk_A &:= Z(\tau)^{-1}pk_{A,m+1} = \rho_A P_1 & sk'_A &:= Z(\tau)^{-1}pk'_{A,m+1} = \alpha_A \rho_A P_1 \\
 sk_B &:= Z(\tau)^{-1}pk_{B,m+2} = \rho_B P_2 & sk'_B &:= Z(\tau)^{-1}pk'_{B,m+2} = \alpha_B \rho_B P_1 \\
 sk_C &:= Z(\tau)^{-1}pk_{C,m+3} = \rho_A \rho_B P_1 & sk'_C &:= Z(\tau)^{-1}pk'_{C,m+3} = \alpha_C \rho_A \rho_B P_1 \\
 sk''_A &:= Z(\tau)^{-1}pk_{K,m+1} = \beta \rho_A P_1 \\
 sk''_B &:= Z(\tau)^{-1}pk_{K,m+2} = \beta \rho_B P_1 & sk''_C &:= Z(\tau)^{-1}pk_{K,m+3} = \beta \rho_A \rho_B P_1
 \end{aligned}$$

3. Compute $vk_x := pk_{A,0} + \sum_{i=1}^n x_i pk_{A,i}$ and $vk'_x := pk'_{A,0} + \sum_{i=1}^n x_i pk'_{A,i}$
4. Choose $a, b, c \leftarrow_{\mathfrak{s}} \mathbb{F}$ and define $\pi := (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$ with:

$$\begin{aligned}
 \pi_A &:= a sk_A - vk_x = a \rho_A P_1 - vk_x & \pi'_A &:= a sk'_A - vk'_x = a \alpha_A \rho_A P_1 - \alpha_A vk_x \\
 \pi_B &:= b sk_B = b \rho_B P_2 & \pi'_B &:= b sk'_B = b \alpha_B \rho_B P_1 \\
 \pi_C &:= c sk_C = c \rho_A \rho_B P_1 & \pi'_C &:= c sk'_C = c \alpha_C \rho_A \rho_B P_1 \\
 \pi_K &:= a sk''_A + b sk''_B + c sk''_C & \pi_H &:= Z(\tau)^{-1}(ab - c) P_1
 \end{aligned}$$

Theorem 25. *Let Rg be a QAP generator defining a bilinear-group generator $aGen$ for which SKE holds. Then the scheme in Fig. 4 for Rg satisfies subversion zero knowledge.*

Proof. Consider $(vk, pk, ck) \leftarrow X(1^\lambda; r)$ and let E denote the event that (R, vk, pk, ck) passes CRS VERIFICATION but Ex' fails to extract τ . From Check 3 in CRS VERIFICATION, we have $\mathbf{e}(pk_{H,1}, P_2) = \mathbf{e}(pk_{H,0}, ck_H)$ and $\mathbf{e}(pk_{H,2}, P_2) = \mathbf{e}(pk_{H,1}, ck_H)$; thus $(pk_{H,0}, pk_{H,1}, pk_{H,2}, P_2, ck_H)$ is a valid SKE tuple. By the SKE assumption the probability of E is thus negligible. It now suffices to show that, conditioned on E not happening, the probability that A outputs 1 in game S-ZK when $b = 0$ is the same as when $b = 1$.

If (vk, pk, ck) fails CRS VERIFICATION then $\tau = \perp$ and both prover and proof simulator return \perp . If (vk, pk, ck) verifies then (because of $\neg E$) Ex' extracts τ .

We show that the outputs of the prover and the proof simulator are distributed equivalently. Above we showed that for a valid CRS there exist $\tau, \rho_A, \rho_B, \beta, \gamma, \alpha_A, \alpha_B, \alpha_C \in \mathbb{F}$ with $\rho_A \neq 0, \rho_B \neq 0, \beta \neq 0, \gamma \neq 0$ and $Z(\tau) \neq 0$ such that vk and pk are defined as in Items 4. and 5. in KEY GENERATION.

Because of this, $\delta_A Z(\tau)\rho_A P_1$, the $(m+2)$ -th summand in π_A is uniformly random. And so are the $(m+3)$ -th summand $\delta_B Z(\tau)\rho_B P_1$ of π_B and the $(m+4)$ -th summand $\delta_C Z(\tau)\rho_A \rho_B P_1$ in π_C . But this means that π_A , π_B and π_C are uniformly random group elements. For fixed vk , π_A , π_B and π_C the Eq. (2). of VERIFY uniquely determine π'_A , π'_B and π'_C , while the Eqs. 3. and 4. uniquely determine π_K and π_H (since $vk_\gamma \neq 0_{\mathbb{G}_2}$ and $vk_Z \neq 0_{\mathbb{G}_2}$).

Since for a valid CRS the values ρ_A and ρ_B are non-zero, the simulated proof elements π_A , π_B and π_C are also uniformly random. Thus, it suffices to show that the remaining proof elements satisfy the verification equations:

$$\begin{aligned} \mathbf{e}(\pi'_A, P_2) &= \mathbf{e}(a\alpha_A \rho_A P_1 - \alpha_A vk_x, P_2) = \mathbf{e}(\pi_A, vk_A) \\ \mathbf{e}(\pi'_B, P_2) &= \mathbf{e}(b\alpha_B \rho_B P_1, P_2) = \mathbf{e}(vk_B, \pi_B) \\ \mathbf{e}(\pi'_C, P_2) &= \mathbf{e}(c\alpha_C \rho_A \rho_B P_1, P_2) = \mathbf{e}(\pi_C, vk_C) \\ \mathbf{e}(\pi_K, vk_\gamma) &= \mathbf{e}(\beta(a\rho_A P_1 + b\rho_B P_1 + c\rho_A \rho_B P_1), \gamma P_2) \\ &= \mathbf{e}(vk_x + \pi_A + \pi_C, \widehat{vk}_{\beta\gamma}) \cdot \mathbf{e}(vk_{\beta\gamma}, \pi_B) \\ \mathbf{e}(\pi_H, vk_Z) &= \mathbf{e}(Z(\tau)^{-1}(ab - c)P_1, Z(\tau)\rho_A \rho_B P_2) \\ &= \mathbf{e}(a\rho_A P_1, b\rho_B P_2) \cdot \mathbf{e}(c\rho_A \rho_B P_1, P_2)^{-1} = \mathbf{e}(vk_x + \pi_A, \pi_B) \cdot \mathbf{e}(\pi_C, P_2)^{-1} \end{aligned}$$

This concludes the proof. \square

Corollary 26. *The scheme in Fig. 4 for a QAP generator Rg satisfies perfect subversion witness indistinguishability.*

Proof. In Theorem 25 we showed that proofs under a (possibly maliciously generated but) valid CRS are uniform group elements subject to satisfying the verification equation. Proofs using different witnesses are thus equally distributed. \square

DFGK's SNARK. Danezis et al. [DFGK14] define *square* span programs, which are described by only one set $\{A_i(X)\}_i$ of polynomials (cf. Definition 20). They show how to convert any boolean circuit into an SSP. They construct a zk-SNARK for SSPs with proofs only consisting of 4 elements of an asymmetric bilinear group.

Analogously to the SNARK from [BCTV14], their scheme is shown to satisfy subversion ZK by observing that (1) the structure of a CRS can be verified via the bilinear map; (2) the trapdoor τ (s in their notation) can be extracted analogously to the SNARK analyzed above; and (3) proofs can be simulated using s by simply following the simulation procedure described in [DFGK14]. (When s is known, the element G^β (in their multiplicative notation) can be obtained from the CRS element $G^{\beta t(s)}$ since $t(s) \neq 0$.)

5 Groth's Near-Optimal SNARK

Groth [Gro16] proposed the most efficient zk-SNARK system to date. He drastically reduced the proof size for QAP-based SNARKs to 3 group elements and verification to one equation using 3 pairings. He achieves this by proving soundness directly in the generic-group model. His system is given in Fig. 5, to which we added a procedure CRS VERIFICATION defined below.

Theorem 27. ([Gro16]). *The scheme in Fig. 5 is knowledge-sound against adversaries that only use a polynomial number of generic bilinear group operations. Moreover, it has perfect zero knowledge.*

KEY GENERATION. On input R as in Eq. (7) representing a QAP for an asymmetric group Gr do the following:

1. Sample random group generators $P_1 \leftarrow \mathbb{G}_1^*$ and $P_2 \leftarrow \mathbb{G}_2^*$.
2. Sample random $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{F}^*$ and $\tau \leftarrow \mathbb{F}$ conditioned on $Z(\tau) \neq 0$.
3. Set $vk = (P_1, P_2, vk_T, vk'_\gamma, vk'_\delta, vk_L)$, where

$$\begin{aligned} vk_T &:= \mathbf{e}(P_1, P_2)^{\alpha\beta} & vk'_\gamma &:= \gamma P_2 & vk'_\delta &:= \delta P_2 \\ \text{for } i = 0, \dots, n: & vk_{L,i} &:= \gamma^{-1}(\beta A_i(\tau) + \alpha B_i(\tau) + C_i(\tau)) P_1 \end{aligned}$$

4. Set $pk = (pk_\alpha, pk_\beta, pk'_\beta, pk_\delta, pk'_\delta, pk_H, pk'_H, pk_K, pk_Z)$, where

$$\begin{aligned} pk_\alpha &:= \alpha P_1 & pk_\beta &:= \beta P_1 & pk'_\beta &:= \beta P_2 & pk_\delta &:= \delta P_1 & pk'_\delta &:= \delta P_2 \\ \text{for } i = 0, \dots, d-1: & pk_{H,i} &:= \tau^i P_1 & pk'_{H,i} &:= \tau^i P_2 \\ \text{for } i = n+1, \dots, m: & pk_{K,i} &:= \delta^{-1}(\beta A_i(\tau) + \alpha B_i(\tau) + C_i(\tau)) P_1 \\ \text{for } i = 0, \dots, d-2: & pk_{Z,i} &:= \delta^{-1} \tau^i Z(\tau) P_1 \end{aligned}$$

5. Return $crs := (vk, pk)$.

PROVE. On input $R, (vk, pk)$ and $\vec{s} \in \mathbb{F}^m$ s.t. Eq. (6) is satisfied for some $H(X) \in \mathbb{F}[X]$:

1. **If (R, vk, pk) does not pass CRS VERIFICATION then return \perp .**
2. Compute $H(X)$ such that Eq. (6) is satisfied and let (h_0, \dots, h_{d-2}) be its coefficients.
3. Sample $r, s \leftarrow \mathbb{F}$ and define

$$\begin{aligned} \pi_A &:= pk_\alpha + \sum_{j=0}^{d-1} (a_{0,j} + s_i \sum_{i=1}^m a_{i,j}) pk_{H,j} + r pk_\delta \\ \pi'_B &:= pk'_\beta + \sum_{j=0}^{d-1} (b_{0,j} + s_i \sum_{i=1}^m b_{i,j}) pk'_{H,j} + s pk'_\delta \\ \pi_{B,\text{aux}} &:= pk_\beta + \sum_{j=0}^{d-1} (b_{0,j} + s_i \sum_{i=1}^m b_{i,j}) pk_{H,j} + s pk_\delta \\ \pi_C &:= \sum_{i=n+1}^m s_i pk_{K,i} + \sum_{j=0}^{d-2} h_j pk_{Z,i} + s \pi_A + r \pi_{B,\text{aux}} - r s pk_\delta \end{aligned}$$

4. Return $\pi := (\pi_A, \pi'_B, \pi_C)$.

VERIFY. On input $R, vk, \vec{x} \in \mathbb{F}^n$ and proof $\pi \in \mathbb{G}_1^2 \times \mathbb{G}_2$:

1. Compute $vk_x := vk_{L,0} + \sum_{i=1}^n x_i vk_{L,i}$.
2. Return true if and only if the following holds:

$$\mathbf{e}(\pi_A, \pi'_B) = vk_T + \mathbf{e}(vk_x, vk'_\gamma) + \mathbf{e}(\pi_C, vk'_\delta)$$

Fig. 5. Groth's SNARK [Gro16] with CRS verification (in bold)

CRS VERIFICATION. On input (R, vk, pk) , let $\{a_{i,j}\}$, $\{b_{i,j}\}$, $\{c_{i,j}\}$, $\{z_k\}$ denote the coefficients of $A_i(X)$, $B_i(X)$, $C_i(X)$ and $Z(X)$, respectively.

1. Check $P_1 \neq 0_{\mathbb{G}_1}$ and $P_2 \neq 0_{\mathbb{G}_2}$.
2. Check $\alpha, \beta, \gamma, \delta$ and $Z(\tau)$ are non-zero: $pk_\alpha \neq 0_{\mathbb{G}_1}, pk_\beta \neq 0_{\mathbb{G}_1}, vk'_\gamma \neq 0_{\mathbb{G}_2}, pk_\delta \neq 0_{\mathbb{G}_1}, pk_{Z,0} \neq 0_{\mathbb{G}_1}$
3. Check consistency of pk_H and pk'_H : check $pk_{H,0} = P_1$ and $pk'_{H,0} = P_2$.
For $i = 1, \dots, d$:

$$\mathbf{e}(pk_{H,i}, P) = \mathbf{e}(pk_{H,i-1}, pk'_{H,1}) \quad \mathbf{e}(P_1, pk'_{H,i}) = \mathbf{e}(pk_{H,i}, P_2)$$

4. Check consistency of the remaining pk elements:

$$\mathbf{e}(P_1, pk'_\beta) = \mathbf{e}(pk_\beta, P_2) \quad \mathbf{e}(P_1, pk'_\delta) = \mathbf{e}(pk_\delta, P_2)$$

$$\text{for } i = n+1, \dots, m: \quad \mathbf{e}(pk_{K,i}, pk'_\delta) = \mathbf{e}\left(\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, pk'_\beta\right) \cdot \mathbf{e}\left(pk_\alpha, \sum_{j=0}^{d-1} b_{i,j} pk'_{H,j}\right) \cdot \mathbf{e}\left(\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, P_2\right)$$

$$\text{for } i = 0, \dots, d-2: \quad \mathbf{e}(pk_{Z,i}, pk'_\delta) = \mathbf{e}\left(\sum_{j=0}^{d-1} z_j pk_{H,j}, pk'_{H,i}\right)$$

5. Check consistency of the remaining vk elements: check $vk_T = \mathbf{e}(pk_\alpha, pk'_\beta)$ and $vk'_\delta = pk'_\delta$. For $i = 0, \dots, n$:

$$\mathbf{e}(pk_{L,i}, pk'_\gamma) = \mathbf{e}\left(\sum_{j=0}^{d-1} a_{i,j} pk_{H,j}, pk'_\beta\right) \cdot \mathbf{e}\left(pk_\alpha, \sum_{j=0}^{d-1} b_{i,j} pk'_{H,j}\right) \cdot \mathbf{e}\left(\sum_{j=0}^{d-1} c_{i,j} pk_{H,j}, P_2\right)$$

6. If all checks in 1.–5. succeeded then return true and otherwise false.

CRS Verifiability. Let $\tau, \alpha, \beta, \gamma, \delta$ denote the logarithms of $pk_{H,1}, pk_\alpha, pk_\beta, vk'_\gamma, pk_\delta$. By Check 2. in CRS VERIFICATION, $\alpha, \beta, \gamma, \delta, Z(\tau)$ are non-zero. It follows by inspection that if all checks in 3.–5. pass then the remaining elements of pk and vk are correctly computed.

Trapdoor Extraction. Let X be a CRS subverter that outputs (vk, pk) . Define $\mathsf{X}'(1^\lambda; r)$ that runs $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$, parses the output as above and returns $(P_1, pk_{H,1}, pk_{H,2}, P_2, pk'_{H,1})$. For a valid CRS this corresponds to $(P_1, \tau P_1, \tau^2 P_1, P_2, \tau P_2)$ for some $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ and $\tau \in \mathbb{F}$. By SKE there exists a PT algorithm $\mathsf{E}_{\mathsf{X}'}$ which from a valid tuple extracts τ with overwhelming probability.

Define another algorithm $\mathsf{X}''(1^\lambda; (r, r'))$ that runs $(vk, pk) \leftarrow \mathsf{X}(1^\lambda; r)$ and extracts $\tau \leftarrow \mathsf{E}_{\mathsf{X}'}(1^\lambda; r; r')$, computes $Z(\tau)$ (which is non-zero in a valid CRS) and sets $P'_1 := Z(\tau)^{-1} pk_{Z,0}$ (which for a valid CRS yields $P'_1 = \delta^{-1} P_1$). Finally, X'' returns $(P'_1, P_1, pk_\delta, P_2, pk'_\delta)$. For a valid CRS this corresponds to $(P'_1, \delta P'_1, \delta^2 P'_1, P_2, \delta P_2)$. By SKE there exist a PT algorithm $\mathsf{E}_{\mathsf{X}''}$ that on input $r'' = (r, r')$ returns δ with overwhelming probability.

Using $\mathsf{E}_{\mathsf{X}'}$ and $\mathsf{E}_{\mathsf{X}''}$, we define the CRS simulator $\mathsf{S.crs}$ as follows: On input 1^λ do the following:

- Sample randomness for X and $E_{X'}$: $r \leftarrow_s \{0, 1\}^{X.r(\lambda)}$; $r' \leftarrow_s \{0, 1\}^{E_{X'}.r(\lambda)}$
- Run $(vk, pk) \leftarrow X(1^\lambda; r)$
- If (R, vk, pk) verifies then $\tau \leftarrow E_{X'}(1^\lambda, r; r')$ and $\delta \leftarrow_s E_{X''}(1^\lambda, (r, r'))$, else $(\tau, \delta) \leftarrow (\perp, \perp)$
- Return $((vk, pk), r, (\tau, \delta))$

Remark 28. Proof simulation is defined in [Gro16] using the full randomness of the CRS and does not work with the trapdoor (τ, δ) , as the simulator requires α and β , which SKE does not allow to extract. Note that it is impossible to extract α , since a valid CRS can be computed without knowing α : obviously sample a random generator $pk_\alpha \leftarrow_s \mathbb{G}_1^*$ and then compute vk_T and, for all i , $vk_{L,i}$ and $pk_{K,i}$ from pk_α . In the following we show how to simulate a proof without knowledge of α and β .

Proof Simulation. Given (vk, pk) , trapdoor (τ, δ) and a statement $x \in \mathbb{F}^n$, the proof simulator $S.pf$ does the following:

1. If $(\tau, \delta) = (\perp, \perp)$ then return \perp .
2. Choose $a, b \leftarrow_s \mathbb{F}$ and define the proof $\pi := (\pi_A, \pi'_B, \pi_C)$ as follows

$$\begin{aligned} \pi_A &:= aP_1 + pk_\alpha & \pi'_B &:= bP_2 + pk'_\beta \\ \pi_C &:= \delta^{-1}(ab - C_0(\tau) - \sum_{i=1}^n x_i C_i(\tau))P_1 + \delta^{-1}(b - B_0(\tau) - \sum_{i=1}^n x_i B_i(\tau))pk_\alpha \\ & \quad + \delta^{-1}(a - A_0(\tau) - \sum_{i=1}^n x_i A_i(\tau))pk_\beta \end{aligned}$$

Theorem 29. Let Rg be a QAP generator defining a bilinear-group generator $aGen$ for which SKE holds. Then Groth’s SNARK [Gro16] with CRS verification (Fig. 5) for Rg satisfies subversion zero knowledge.

Proof. Let E denote the event that (R, vk, pk) passes CRS verification but either $E_{X'}$ or $E_{X''}$ fails to extract τ and δ . Since a correct (vk, pk) satisfies $e(pk_{H,1}, P_2) = e(P_1, pk'_{H,1})$ as well as $e(pk_{H,2}, P_2) = e(pk_{H,1}, pk'_{H,1})$, by SKE (Definition 15), the probability that $E_{X'}$ fails when X' outputs $(P_1, pk_{H,1}, pk_{H,2}, P_2, pk'_{H,1})$ is negligible. A correct CRS also satisfies both $e(P_1, P_2) = e(Z(\tau)^{-1}pk_{Z,0}, pk'_\delta)$ and $e(pk_\delta, P_2) = e(P_1, pk'_\delta)$, thus again by SKE, the probability that $E_{X''}$ fails when X'' outputs $(Z(\tau)^{-1}pk_{Z,0}, P_1, pk_\delta, P_2, pk'_\delta)$ is also negligible. By a union bound, the probability of E is thus negligible.

It now suffices to show that, conditioned on E not happening, game S-ZK when $b = 0$ is distributed as game S-ZK when $b = 1$. If (vk, pk) fails verification then $(\tau, \delta) = (\perp, \perp)$ and both the prover and the proof simulator return \perp .

If (vk, pk) verifies then we show that the outputs of the prover and the proof simulator are distributed equivalently. Above we argued that for some non-zero $\alpha, \beta, \gamma, \delta$ and τ with $Z(\tau) \neq 0$ we have that vk and pk are defined as in 3. and 4. in KEY GENERATION.

Since for a valid CRS both pk_δ and pk'_δ are non-zero, for honestly generated proofs the elements rp_k_δ in π_A , and $sp_k'_\delta$ in π'_B , make π_A and π'_B uniformly random. For fixed vk , π_A and π'_B , the verification equation uniquely determines π_C , since $vk'_\delta \neq 0$.

In a simulated proof π_A and π'_B are also uniformly random, so it suffices to show that the simulated π_C satisfies the verification equation:

$$\begin{aligned} \mathbf{e}(\pi_C, vk'_\delta) &= \mathbf{e}\left(\left(ab - C_0(\tau) - \sum x_i C_i(\tau) + \alpha(b - B_0(\tau) - \sum x_i B_i(\tau)) + \beta(a - A_0(\tau) - \sum x_i A_i(\tau))\right)P_1, P_2\right) \\ &= \mathbf{e}(abP_1, P_2) + \mathbf{e}(a\beta P_1, P_2) + \mathbf{e}(\alpha b P_1, P_2) + \mathbf{e}(\alpha\beta P_1, P_2) - \mathbf{e}(\alpha\beta P_1, P_2) \\ &\quad - \mathbf{e}\left(\left(\beta A_0(\tau) + \sum x_i \beta A_i(\tau) + \alpha B_0(\tau) + \sum x_i \alpha B_i(\tau) + C_0(\tau) + \sum x_i C_i(\tau)\right)P_1, P_2\right) \\ &= \mathbf{e}(\pi_A, \pi'_B) - vk_T - \mathbf{e}(vk_x, vk'_\gamma) \end{aligned}$$

This concludes the proof. \square

Corollary 30. *Groth's SNARK [Gro16] with CRS verification for a QAP generator Rg (Fig. 5) satisfies perfect subversion witness indistinguishability.*

Proof. The corollary follows analogously to Corollary 26. \square

Acknowledgments. The author would like to thank Mihir Bellare and Rosario Genaro for helpful discussions and the anonymous reviewers for PKC'18 for their valuable comments. The author is supported by the French ANR EFTiEC project (ANR-16-CE39-0002).

References

- [ABLZ17] Abdolmaleki, B., Bagheri, K., Lipmaa, H., Zając, M.: A subversion-resistant SNARK. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 3–33. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_1
- [AF07] Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 118–136. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_7
- [BB04] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4
- [BBS04] 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
- [BCCT12] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) ITCS 2012, pp. 326–349. ACM, January 2012

- [BCG+13] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_6
- [BCG+14a] Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE Computer Society Press, May 2014
- [BCG+14b] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: libsnark (2014). <https://github.com/scipr-lab/libsnark>
- [BCG+15] Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy, pp. 287–304. IEEE Computer Society Press, May 2015
- [BCI+10] Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_13
- [BCI+13] Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_18
- [BCPR14] Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: Shmoys, D.B. (ed.) 46th ACM STOC, pp. 505–514. ACM Press, May/June 2014
- [BCTV14] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von Neumann architecture. In: Fu, K., Jung, J. (eds.) USENIX Security Symposium, pp. 781–796. USENIX Association (2014)
- [BDMP91] Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. *SIAM J. Comput.* **20**(6), 1084–1118 (1991)
- [BF01] Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
- [BFM88] 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
- [BFS16] Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: security in the face of parameter subversion. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 777–804. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_26
- [BG93] Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_28
- [BGG17] Bove, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. *Cryptology ePrint Archive, Report 2017/602* (2017). <http://eprint.iacr.org/2017/602>

- [BP04] Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_17
- [BR93] Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93, pp. 62–73. ACM Press, November 1993
- [BR06] 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
- [BSBC+17] Ben-Sasson, E., et al.: Computational integrity with a public random string from quasi-linear PCPs. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 551–579. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_19
- [CGGN17] Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent payments revisited: attacks and payments for services. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17, pp. 229–243. ACM Press, October/November 2017
- [CNE+14] Checkoway, S., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D.J., Maskiewicz, J., Shacham, H., Fredrikson, M.: On the practical exploitability of Dual EC in TLS implementations. In: Fu, K., Jung, J. (ed.) USENIX Security Symposium, pp. 319–335. USENIX Association (2014)
- [Dam92] Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_36
- [DFGK14] Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 532–550. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_28
- [FKL17] Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. Cryptology ePrint Archive, Report 2017/620 (2017). <http://eprint.iacr.org/2017/620>
- [FLS90] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS, pp. 308–317. IEEE Computer Society Press, October 1990
- [FS87] Fiat, A., Shamir, A.: How To prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
- [Fuc17] Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. Cryptology ePrint Archive, Report 2017/587 (2017). <http://eprint.iacr.org/2017/587>
- [Gen04] Gennaro, R.: Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 220–236. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_14

- [GGPR13] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
- [GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
- [GO94] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. *J. Cryptol.* **7**(1), 1–32 (1994)
- [GOS06a] Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_6
- [GOS06b] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_21
- [Gro06] Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_29
- [Gro10] 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
- [Gro16] Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
- [GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_24
- [GW11] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC, pp. 99–108. ACM Press, June 2011
- [HT98] Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055744>
- [Lip12] 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 (2012). https://doi.org/10.1007/978-3-642-28914-9_10
- [Mic00] Micali, S.: Computationally sound proofs. *SIAM J. Comput.* **30**(4), 1253–1298 (2000)
- [Nak09] Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009). <http://bitcoin.org/bitcoin.pdf>
- [PHGR13] Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252. IEEE Computer Society Press, May 2013
- [Sch91] Schnorr, C.-P.: Efficient signature generation by smart cards. *J. Cryptol.* **4**(3), 161–174 (1991)

- [Sho97] 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
- [SvdW06] Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_36
- [Zca] Zcash. <http://z.cash>



Public-Key Encryption Resistant to Parameter Subversion and Its Realization from Efficiently-Embeddable Groups

Benedikt Auerbach^{1(✉)}, Mihir Bellare², and Eike Kiltz¹

¹ Horst-Görtz Institute for IT Security and Faculty of Mathematics,
Ruhr-University Bochum, Bochum, Germany
{benedikt.auerbach,eike.kiltz}@rub.de

² Department of Computer Science and Engineering,
University of California San Diego, 9500 Gilman Drive,
La Jolla, CA 92093, USA
mihir@eng.ucsd.edu

Abstract. We initiate the study of public-key encryption (PKE) schemes and key-encapsulation mechanisms (KEMs) that retain security even when public parameters (primes, curves) they use may be untrusted and subverted. We define a strong security goal that we call ciphertext pseudo-randomness under parameter subversion attack (CPR-PSA). We also define indistinguishability (of ciphertexts for PKE, and of encapsulated keys from random ones for KEMs) and public-key hiding (also called anonymity) under parameter subversion attack, and show they are implied by CPR-PSA, for both PKE and KEMs. We show that hybrid encryption continues to work in the parameter subversion setting to reduce the design of CPR-PSA PKE to CPR-PSA KEMs and an appropriate form of symmetric encryption. To obtain efficient, elliptic-curve-based KEMs achieving CPR-PSA, we introduce efficiently-embeddable group families and give several constructions from elliptic-curves.

1 Introduction

This paper initiates a study of public-key encryption (PKE) schemes, and key-encapsulation mechanisms (KEMs), resistant to subversion of public parameters. We give definitions, and efficient, elliptic-curve-based schemes. As a tool of independent interest, we define efficiently-embeddable group families and construct them from elliptic curves.

PARAMETER SUBVERSION. Many cryptographic schemes rely on some trusted, public parameters common to all users and implementations. Sometimes these are specified in standards. The Oakley primes [39], for example, are a small number of fixed prime numbers widely used for discrete-log-based systems. For ECC (Elliptic Curve Cryptography), the parameters are particular curves. Examples include the P-192, P-224, ... curves from the FIPS-186-4 [38] standard and Ed25519 [16].

There are many advantages to such broad use of public parameters. For example, it saves implementations from picking their own parameters, a task that can be error-prone and difficult to do securely. It also makes key-generation faster and allows concrete-security improvements in the multi-user setting [7]. Recent events indicate, however, that public parameters also bring a risk, namely that they can be *subverted*. The representative example is Dual-EC. We refer to [19] for a comprehensive telling of the story. Briefly, Dual EC was a PRG whose parameters consisted of a description of a cyclic group and two generators of the group. If the discrete logarithm of one generator to base the other were known, security would be compromised. The Snowden revelations indicate that NIST had adopted parameters provided by the NSA and many now believe these parameters had been subverted, allowing the NSA to compromise the security of Dual EC. Juniper’s use of Dual EC further underscores the dangers [21].

SECURITY IN THE FACE OF PARAMETER SUBVERSION. DGGJR [26] and BFS [9] initiated the study of cryptography that retains security in the face of subverted parameters, the former treating PRGs and the latter treating NIZKs, where the parameter is the common reference string. In this paper we treat encryption. We define what it means for parameter-using PKE schemes and KEMs to retain security in the face of subversion of their parameters. With regard to schemes, ECC relies heavily on trusted parameters. Accordingly we focus here, providing various efficient elliptic-curve-based schemes that retain security in the face of parameter subversion.

CURRENT MITIGATIONS. In practice, parameters are sometimes specified in a verifiable way, for example derived deterministically (via a public algorithm) from publicly-verifiable coins. The coins could be obtained by applying a hash function like SHA1 to some specified constants (as is in fact done for the FIPS-186-4 curves [38] and in the ECC brainpool project), via the first digits of the irrational number π , or via lottery outcomes [5]. This appears to reduce the possibility of subversion, but BCCHLN [15] indicate that the potential of subverting elliptic curves still remains, so there is cause for caution even in this regard. Also, even if such mechanisms might “work” in some sense, we need definitions to understand what “work” means, and proofs to ensure definitions are met. Our work gives such definitions.

BACKGROUND. A PKE scheme specifies a parameter generation algorithm that returns parameters π , a key-generation algorithm that takes π and returns a public key pk and matching secret key sk , an encryption algorithm that given π, pk and message m returns a ciphertext c , and a decryption algorithm that given π, sk, c recovers m . We denote the classical notions of security by IND—indistinguishability of ciphertexts under chosen-ciphertext attack [8, 22]—and PKH—public-key hiding, also called anonymity, this asks that ciphertexts not reveal the public key under which they were created [6]. For KEMs, parameter and key generation are the same, encryption is replaced by encapsulation—it takes π, pk to return an encapsulated key K and a ciphertext c that encapsulates K —and decryption is replaced by decapsulation—given π, sk, c it recov-

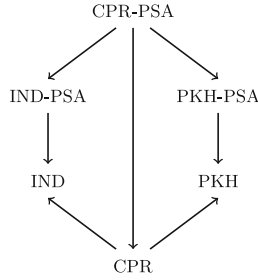


Fig. 1. Relations between notions of security. The notions are defined, and the relations hold, for both PKE schemes and KEMs. An arrow $A \rightarrow B$ is an implication: if a scheme meets A then it also meets B .

ers K . We continue to denote the classical goals by IND—this now asks for indistinguishability of encapsulated keys from random under chosen-ciphertext attack [23]—and PKH. We stress that these classical notions assume *honest parameter generation*, meaning *the parameters are trusted*.

We know that, in this setting, IND PKE is reduced, via hybrid encryption, to IND KEMs and ind-cpa symmetric encryption [23]. To the best of our knowledge, no analogous result exists for PKH.

Mass surveillance activities have made apparent the extent to which privacy can be violated purely by access to meta-data, including who is communicating with whom. PKE and KEMs providing PKH are tools towards systems that do more to hide identities of communicants. We will thus target this goal in the parameter subversion setting as well.

DEFINITIONS AND RELATIONS. For both PKE and KEMs, we formulate a goal called ciphertext pseudorandomness under parameter subversion attack, denoted CPR-PSA. It asks that ciphertexts be indistinguishable from strings drawn randomly from the ciphertext space, even under a chosen-ciphertext attack (CCA). We also extend the above-discussed classical goals to the parameter subversion setting, defining IND-PSA and PKH-PSA. For both PKE (Proposition 1) and KEMs (Proposition 2) we show that CPR-PSA implies both IND-PSA and PKH-PSA. We thus get the relations between the new and classical notions summarized in Fig. 1. (Here CPR is obtained by dropping the PSA in CPR-PSA, meaning it is our definition with honest parameter generation. This extends the notions of [26, 37] to allow a CCA.)

We ask whether we can reduce the design of CPR-PSA PKE to the design of CPR-PSA KEMs via hybrid encryption. Proposition 3 says the answer is yes, but, interestingly, requires that the KEM has an extra property of well-distributed ciphertexts that we denote WDC-PSA. (The symmetric encryption scheme is required to have pseudo-random ciphertexts. Such symmetric schemes are easily obtained.) We now have a single, strong target for constructions, namely CPR-PSA+WDC-PSA KEMs. (By the above they imply CPR-PSA PKE, which in turn implies IND-PSA PKE and PKH-PSA PKE.) Our goal thus becomes to build efficient KEMs that are CPR-PSA+WDC-PSA.

PARAMETER-FREE SCHEMES. We say that a scheme (PKE or KEM) is parameter free if there are no parameters. (Formally, the parameters are the empty string ε .) Note that a parameter-free scheme that is XXX secure is trivially also XXX-PSA secure. ($\text{XXX} \in \{\text{CPR}, \text{IND}, \text{PKH}\}$.) This is an important observation, and some of our schemes will indeed be parameter-free, but, as we discuss next, this observation does not trivialize the problem.

ISSUES AND CHALLENGES. In an attempt to achieve PSA security through the above observation, we could consider the following simple way to eliminate parameters. Given a XXX-secure parameter-using scheme, build a parameter-free version of it as follows: the new scheme sets its parameters to the empty string; key generation runs the old parameter generation algorithm to get π , then the old key generation algorithm to get pk and sk , setting the new public and secret keys to (π, pk) and (π, sk) , respectively; encryption and decryption can then follow the old scheme. This trivial construction, however, has drawbacks along two dimensions that we expand on below: (1) security and (2) efficiency.

With regard to security, the question is, if the old scheme is XXX, is the new one too? (If so, it is also XXX-PSA, since it is parameter free, so we only need to consider the classical notions.) The answer to the question is yes if $\text{XXX} = \text{IND}$, but *no if* $\text{XXX} \in \{\text{PKH}, \text{CPR}\}$. Imagine, as typical, that the parameters describe a group. Then in the new scheme, different users use different, independent groups. This will typically allow violation of PKH [6]. For example, in the El Gamal KEM, a ciphertext is a group element, so if two users have groups \mathbb{G}_0 and \mathbb{G}_1 , respectively, one can determine which user generated a ciphertext by seeing to which of the two groups it belongs. The same is true for RSA where the group $\mathbb{G}_i = \mathbb{Z}_{N_i}$ is determined by the modulus N_i in the key of user i . Even when the moduli have the same bit length, attacks in [6] show how to violate PKH-security of the simple RSA KEM.

With regard to efficiency, the drawback is that we lose the benefits of parameter-using schemes noted above. In particular, key-generation is less efficient (because it involves parameter generation for the old scheme, which can be costly), and public keys are longer (because they contain the parameters of the old scheme). We aim to retain, as much as possible, the efficiency benefits of parameters while adding resistance to PSA.

BBDP [6] give (1) parameter-free IND+PKH RSA-based PKE schemes and (2) parameter-using discrete-log based IND+PKH PKE schemes. The former, since parameter-free, are IND-PSA+PKH-PSA, but they are not CPR-PSA and they are not as efficient as ECC-based schemes. The latter, while ECC-based and fast, are not secure against PSA.

The open question that emerges is thus to design efficient, ECC-based KEMs that are CPR-PSA+WDC-PSA. The technical challenge is to achieve CPR-PSA (and thus PKH-PSA) even though the groups of different users may be different.

OVERVIEW OF THE APPROACH. We introduce and formalize *efficiently-embeddable group (eeg) families* and identify desirable security properties for them. We give a transform constructing CPR-PSA+WDC-PSA KEMs from secure eeg families. This reduces our task to finding secure eeg families. We

Table 1. Our elliptic curve based CPR-PSA+WDC-PSA KEMs. p denotes the modulus of the field. Efficiency of KE.G is dominated by the sampling time of the curves. Efficiency of KE.E (average, worst case) and KE.D (worst case) is given as the number of exponentiations on the curves. The key size is measured in bits, $k = \lceil \log_2 p \rceil$ being the bit length of the used modulus. For the rejection sampling based constructions, ℓ denotes the cut-off bound. For transform **eegToKE2** and the constructions based on Elligator curves (last two rows) see [4].

Eeg family	Transform	Parameter	Assumption	Efficiency			Key size
				KE.G	KE.E	KE.D	
EG_{twist}	eegToKE1	p	sCDH-PSA	t_{TGen}	2, 2	2	$10k$
EG_{twist}	eegToKE2	p	CDH-PSA	t_{TGen}	3, 3	3	$12k$
$EG_{\text{twist-rs}}^\ell$	eegToKE1	—	sCDH-PSA	t_{TGen}	3, $\ell+1$	1	$9k$
$EG_{\text{twist-rs}}^\ell$	eegToKE2	—	CDH-PSA	t_{TGen}	4, $\ell+2$	2	$11k$
$EG_{\text{twist-re}}$	eegToKE1	—	sCDH-PSA	t_{TGen}	3, 3	1	$9k$
$EG_{\text{twist-re}}$	eegToKE2	—	CDH-PSA	t_{TGen}	4, 4	2	$11k$
$EG_{\text{ell1}}^\ell, EG_{\text{ell2}}^\ell$	eegToKE1	p	sCDH-PSA	t_{EllGen}	3, $\ell+1$	1	$6k$
$EG_{\text{ell1-rs}}^\ell, EG_{\text{ell2-rs}}^\ell$	eegToKE1	—	sCDH-PSA	t_{EllGen}	5, $\ell+1$	1	$5k$

propose several instantiations of eeg families from elliptic curves with security based on different assumptions. An overview of the resulting KEMs is given in Table 1. We discuss our results in greater detail below.

EFFICIENTLY-EMBEDDABLE GROUP FAMILIES. As described above, having users utilize different groups typically enables linking ciphertexts to the intended receiver and hence violating CPR-PSA. However, certain families of groups allow to efficiently map group elements to a space, which is independent of the particular group of the family. Building on these types of group families it is possible to achieve CPR-PSA secure encryption while still allowing each user to choose his own group.

We formalize the required properties via *efficiently embeddable group families*, a novel abstraction that we believe is of independent interest. An eeg family EG specifies a parameter generation algorithm EG.P sampling parameters to be used by the other algorithms, and a group generation algorithm EG.G sampling a group from the family. Embedding algorithm EG.E embeds elements of the group into some embedding space EG.ES. The group element can be recovered using inversion algorithm EG.I. An important property is that the embedding space only depends on the parameters and in particular not on the used group. Looking ahead, the KEM’s public key will contain a group sampled with EG.S and ciphertexts will be embeddings. We require two security properties for EG in order to achieve CPR-PSA+WDC-PSA KEMs. Both assume parameter subversion attacks and are defined with respect to a sampling algorithm EG.S, which samples (not necessarily uniformly distributed) group elements. The first, embedding pseudorandomness (EPR-PSA), is that embeddings of group elements sampled

with EG.S are indistinguishable from uniform. Further we give a definition the strong computational Diffie-Hellman assumption (sCDH-PSA) with respect to EG —an adaption of the interactive assumption introduced in [2] to our setting. It differs from the usual strong computational Diffie-Hellman assumption in two points. The group used for the challenge is sampled using EG.G on a parameter of the adversary’s choice and additionally one of the exponents used in the challenge is sampled with sampling algorithm EG.S .

KEY ENCAPSULATION MECHANISMS FROM EEG FAMILIES. We provide a transform **eegToKE1** of eeg families to secure KEMs. If the eeg family is both EPR-PSA and sCDH-PSA the resulting KEM is CPR-PSA and WDC-PSA .

KEY ENCAPSULATION FROM WEAKER ASSUMPTIONS. In the full version of this paper [4] we give a second transform **eegToKE2** from eeg families to secure KEMs. It is applicable to eeg families consisting of groups, which order has no small prime factors. Its security is based on the weaker computational Diffie-Hellman assumption (CDH-PSA), i.e. it achieves a CPR-PSA and WDC-PSA KEM under the weaker assumption that EG is both EPR-PSA and CDH-PSA . However, this comes at the cost of larger key size and slower encryption and decryption.

INSTANTIATIONS FROM ELLIPTIC CURVES. We propose several instantiations of eeg families from elliptic curves. It is well known that elliptic curves are not all equal in security. We target elliptic-curve groups over the field \mathbb{F}_p for a large odd prime p since they are less vulnerable to discrete-log-finding attacks than groups over fields of characteristic two [28, 40]. While the usage of standardized primes allows for more efficient implementations, several cryptanalysts further suggest that p should be as random as possible for maximal security, see for example Brainpool’s RFC on ECC [36]. These constraints make building eeg families more challenging. We offer solutions for both cases. We first identify an eeg family implicitly given in prior work [34, 37]. The family consists of curve-twist pairs of elliptic curves. Its embedding space depends on the modulus p of the underlying field, which serves as parameter of the construction.

Building on eeg family EG_{twist} we also provide alternatives, which no longer rely on a fixed modulus. The constructions have empty parameters and p is sampled at random in the group generation algorithm. The technical challenge is to still achieve pseudorandom embeddings in an embedding space independent of the group. Our solution $\text{EG}_{\text{twist-rs}}^\ell$ achieves this by using rejection sampling with cut-off parameter ℓ . Its embedding space consists of bit strings of length only dependent on the security parameter. The sampling algorithm has a worst-case running time of ℓ exponentiations, but the average cost is two exponentiations independently of ℓ . Eeg family $\text{EG}_{\text{twist-re}}$ uses a range expansion technique from [33] and improves on $\text{EG}_{\text{twist-rs}}^\ell$ both in terms of efficiency and security. As in the other construction embeddings are bit strings, but sampling only requires a single exponentiation.

SECURITY OF THE INSTANTIATIONS. We now discuss the security properties of our instantiations in greater detail. An overview is given in Table 2. All of

Table 2. Security of our eeg families. The modulus of the used field is denoted by p . $\Delta_{\text{EPR-PSA}}$ denotes the maximal advantage of an (unbounded) adversary in breaking EPR-PSA. ℓ denotes the cut-off bound used in the construction based on rejection sampling.

Eeg family	Curve type	Parameter	$\Delta_{\text{EPR-PSA}}$	See
EG_{twist}	Twist	p	0	Sect. 5.2
$\text{EG}_{\text{twist-rs}}^\ell$	Twist	—	$(1/2)^\ell$	Sect. 5.3
$\text{EG}_{\text{twist-re}}$	Twist	—	0	Sect. 5.4
$\text{EG}_{\text{ell1}}^\ell, \text{EG}_{\text{ell2}}^\ell$	Elligator	p	$(2/3)^\ell$	[4]
$\text{EG}_{\text{ell1-rs}}^\ell, \text{EG}_{\text{ell2-rs}}^\ell$	Elligator	—	$(4/5)^\ell$	[4]

our constructions achieve EPR-PSA statistically. Embeddings in eeg families EG_{twist} , and $\text{EG}_{\text{twist-re}}$ are perfectly random, i.e. any (unbounded) adversary has advantage 0 in breaking EPR-PSA. For family $\text{EG}_{\text{twist-rs}}^\ell$ the advantage decays exponentially in the cut-off bound ℓ .

Diffie-Hellman problem sCDH-PSA is non standard. It is defined with respect to the eeg family’s sampling algorithm and assumes parameter subversion attacks. However, for all of our proposed instantiations we are able to show that sCDH-PSA can be reduced to assumptions, which no longer depend on the sampling algorithms, but use uniformly sampled exponents instead. Considering the parameters of our constructions, they belong to one of two classes. Eeg family EG_{twist} uses the modulus p as parameter, which might be subject to subversion. Accordingly sCDH-PSA in this case corresponds to the assumption that the adversary’s possibility to choose p does not improve its capacities in solving Diffie-Hellman instances on either the curve or its twist for a curve-twist pair sampled from the family. Eeg families $\text{EG}_{\text{twist-rs}}^\ell$ and $\text{EG}_{\text{twist-re}}$ serve as more conservative alternatives. They are parameter-free and each user choses his own modulus at random, resulting in the weaker assumption that solving Diffie-Hellman instances over curves sampled with respect to a randomly chosen modulus is hard.

INSTANTIATIONS FROM ELLIGATOR CURVES. In the full version of this paper [4] we provide alternatives to our curve-twist pair based constructions. Eeg families $\text{EG}_{\text{ell1}}^\ell, \text{EG}_{\text{ell2}}^\ell, \text{EG}_{\text{ell1-rs}}^\ell$ and $\text{EG}_{\text{ell2-rs}}^\ell$ make use of the Elligator1 and Elligator2 curves of [17]. $\text{EG}_{\text{ell1}}^\ell$ and $\text{EG}_{\text{ell2}}^\ell$ were implicitly given in [17] and use the modulus of the underlying field as parameter. Constructions $\text{EG}_{\text{ell1-rs}}^\ell$ and $\text{EG}_{\text{ell2-rs}}^\ell$ serve as parameter-free alternatives.

RELATED WORK. One might consider generating parameters via a multi-party computation protocol so that no particular party controls the outcome. It is unclear however what parties would perform this task and why one might trust any of them. PKE resistant to parameter subversion provides greater security.

Parameter subversion as we consider it allows the adversary full control of the parameters. This was first considered for NIZKs [9] and (under the term back-

doored) for PRGs [25, 26]. Various prior works, in various contexts, considered relaxing the assumptions on parameters in some way [20, 30, 32, 35], but these do not allow the adversary full control of the parameters and thus do not provide security against what we call parameter subversion.

Algorithm-substitution attacks, studied in [3, 10–12, 24], are another form of subversion, going back to the broader framework of kleptography [43, 44]. The cliptography framework of RTYZ [41] aims to capture many forms of subversion. In [42] the same authors consider PKE that retains security in the face of substitution of any of its algorithms, but do not consider parameter subversion.

2 Preliminaries

NOTATION. We let ε denote the empty string. If X is a finite set, we let $x \leftarrow_s X$ denote picking an element of X uniformly at random and assigning it to x . All our algorithms are randomized and polynomial time (PT) unless stated otherwise. An adversary is an algorithm. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \dots; r)$ denote running A with random coins r on inputs x_1, \dots and assigning the output to y . We let $y \leftarrow_s A(x_1, \dots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \dots; r)$. We let $[A(x_1, \dots)]$ denote the set of all possible outputs of A when invoked with inputs x_1, \dots . We use the code based game playing framework of [14]. (See Fig. 3 for an example.) By $\Pr[G]$ we denote the probability that the execution of game G results in the game returning true. We also adopt the convention that the running time of an adversary refers to the worst case execution time of the game with the adversary. This means that the time taken for oracles to compute replies to queries is included. The random oracle model [13] is captured by a game procedure RO that implements a variable output length random oracle. It takes a string x and an integer m and returns a random m -bit string. We denote by \mathcal{P}_k the set of primes of bit length k and by $[d]$ the set $\{0, \dots, d-1\}$. Furthermore, the uniform distribution on M is denoted by U_M . If two random variables X and Y are equal in distribution we write $X \sim Y$. The statistical distance between X and Y is denoted by $\Delta(X; Y)$. If $\Delta(X; Y) \leq \delta$ we say X is δ -close to Y .

3 Public-Key Encryption Resistant to Parameter Subversion

In this section we recall public-key encryption schemes and key encapsulation mechanisms. For both primitives we define the strong security notion of pseudorandomness of ciphertexts in the setting of parameter subversion and show that it implies both indistinguishability of encryptions and public-key hiding. We further define the security notion of well-distributedness of ciphertexts for key encapsulation mechanisms. Finally, we recall symmetric encryption schemes and revisit the hybrid encryption paradigm in the setting of ciphertext pseudorandomness under parameter subversion attacks.

3.1 Public-Key Encryption Schemes

Below we give a syntax for public-key encryption schemes. It follows [23], but uses slightly different notation and includes an additional algorithm setting up global parameters to be utilized by all users. We then formalize a novel security requirement of pseudorandomness of ciphertexts under parameter subversion attacks (CPR-PSA), which says that even if the parameters of the scheme are controlled by the adversary, ciphertexts obtained under any public key are indistinguishable from random elements of the ciphertext space, which depends only on the security parameter, the message length and the global parameters. We then recall two existing requirements of public-key encryption schemes adapting them to the setting of parameter subversion attacks. The first is the well-known notion of indistinguishability of encryptions [31], the second, from [1, 6], is that ciphertexts under different public keys are indistinguishable, which they called anonymity or key hiding and we call public-key hiding. In Proposition 1 we show that the first requirement implies the other two, allowing us to focus on it subsequently. We model the possibility of subverted parameters by having the adversary provide the parameters, which are used in the security games.

PUBLIC-KEY ENCRYPTION. A *public-key encryption scheme* (PKE) PE specifies the following. Parameter generation algorithm PE.P takes input 1^k , where $k \in \mathbb{N}$ is the security parameter, and returns global parameters π . Key-generation algorithm PE.G takes input $1^k, \pi$ and returns a tuple (pk, sk) consisting of the public (encryption) key pk and matching secret (decryption) key sk . PE.CS associates to k, π and message length $m \in \mathbb{N}$ a finite set $\text{PE.CS}(k, \pi, m)$ that is the *ciphertext space* of PE. Encryption algorithm PE.E takes $1^k, \pi, pk$ and a message $M \in \{0, 1\}^*$ and returns a ciphertext $c \in \text{PE.CS}(k, \pi, |M|)$. Deterministic decryption algorithm PE.D takes $1^k, \pi, sk$ and a ciphertext c and returns either a message $M \in \{0, 1\}^*$ or the special symbol \perp indicating failure. The correctness condition requires that for all $k \in \mathbb{N}$, all $\pi \in [\text{PE.P}(1^k)]$, all $(pk, sk) \in [\text{PE.G}(1^k, \pi)]$ and all $M \in \{0, 1\}^*$ we have $\Pr[\text{PE.D}(1^k, \pi, sk, c) = M] \geq 1 - \text{PE.de}(k)$, where the probability is over $c \leftarrow \text{PE.E}(1^k, \pi, pk, M)$ and $\text{PE.de} : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is the *decryption error* of PE. Our PKEs will be in the ROM [13], which means the encryption and decryption algorithms have access to a random oracle specified in the security games. Correctness must then hold for all choices of the random oracle. We say a PKE is *parameter-free* if PE.P returns ε on every input 1^k .

CIPHERTEXT PSEUDORANDOMNESS. Consider game $\mathbf{G}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(k)$ of Fig. 2 associated to PKE PE, adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(k)] - 1.$$

We say that PE has pseudorandom ciphertexts under parameter subversion attacks (also called CPR-PSA) if the function $\mathbf{Adv}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(\cdot)$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. When $b = 1$, the challenge ciphertext c^* is an encryption of a message of the adversary’s choice, but if $b = 0$ it is chosen at random from the ciphertext space. Given the public key and challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b , the game

<p>Games $\mathbf{G}_{\text{PE},\mathcal{A}}^{\text{cpr-psa}}(k), \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(k), \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(k)$</p> <p>$c^* \leftarrow \perp$ $b \leftarrow_{\\$} \{0, 1\}$ $b' \leftarrow_{\\$} \mathcal{A}^{\text{INIT,ENG,DEC,RO}}(1^k)$ Return $(b = b')$</p> <p>$\text{RO}(x, m) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{cpr-psa}}, \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}, \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}$ If $(T[x, m] = \perp)$ then $T[x, m] \leftarrow_{\\$} \{0, 1\}^m$ Return $T[x, m]$</p> <p>$\text{ENC}(M) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{cpr-psa}}$ If $(pk = \perp)$ then return \perp If $(b = 0)$ then $c^* \leftarrow_{\\$} \text{PE.CS}(k, \pi, M)$ Else $c^* \leftarrow_{\\$} \text{PE.E}^{\text{RO}}(1^k, \pi, pk, M)$ Return c^*</p> <p>$\text{ENC}(M_0, M_1) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}$ If $(pk = \perp)$ then return \perp If $(M_0 \neq M_1)$ then return \perp $c^* \leftarrow_{\\$} \text{PE.E}^{\text{RO}}(1^k, \pi, pk, M_b)$ Return c^*</p> <p>$\text{ENC}(M) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}$ If $(pk_0 = \perp \vee pk_1 = \perp)$ return \perp $c^* \leftarrow_{\\$} \text{PE.E}^{\text{RO}}(1^k, \pi, pk_b, M)$ Return c^*</p>	<p>$\text{INIT}(\pi) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{cpr-psa}}, \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}$ $(pk, sk) \leftarrow_{\\$} \text{PE.G}(1^k, \pi)$ Return pk</p> <p>$\text{INIT}(\pi) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}$ $(pk_0, sk_0) \leftarrow_{\\$} \text{PE.G}(1^k, \pi)$ $(pk_1, sk_1) \leftarrow_{\\$} \text{PE.G}(1^k, \pi)$ If $(pk_0 = \perp \vee pk_1 = \perp)$ return \perp Return (pk_0, pk_1)</p> <p>$\text{DEC}(c) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{cpr-psa}}, \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}$ If $(c = c^*)$ then return \perp Else return $\text{PE.D}^{\text{RO}}(1^k, \pi, sk, c)$</p> <p>$\text{DEC}(c) // \mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}$ If $(c = c^*)$ then return \perp $M_0 \leftarrow \text{PE.D}^{\text{RO}}(1^k, \pi, sk_0, c)$ $M_1 \leftarrow \text{PE.D}^{\text{RO}}(1^k, \pi, sk_1, c)$ Return (M_0, M_1)</p>
---	---

Fig. 2. Games defining security of PKEs. In each game the adversary is given access to oracles. The game, to which an oracle belongs, is indicated behind the oracle's name. In each game oracles INIT and ENC may be queried only once. Further INIT has to be queried before using any of the other oracles.

returning true in this case and false otherwise. The adversary has access to an oracle INIT, which sets up the public key using parameters of the adversary's choice, and an oracle ENC to generate the challenge ciphertext. Furthermore it has access to the random oracle and a decryption oracle crippled to not work on the challenge ciphertext. We require that the adversary queries the oracles INIT and ENC only once. Furthermore INIT has to be queried before using any of the other oracles.

INDISTINGUISHABILITY OF ENCRYPTIONS. Consider game $\mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(k)$ of Fig. 2 associated to PKE PE, adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(k)] - 1.$$

We say that PE has indistinguishable encryptions under parameter subversion attacks (also called IND-PSA) if the function $\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(\cdot)$ is negligible for every

\mathcal{A} . In the game, b is a challenge bit. The adversary has access to an oracle `INIT`, which sets up the public key using parameters of the adversary’s choice, and an oracle `ENC`, which receives as input two messages M_0, M_1 of the same length and outputs the challenge ciphertext c^* . When $b = 0$, the challenge ciphertext is an encryption of M_0 , if $b = 1$ an encryption of M_1 . Given the public key and challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b , the game returning `true` in this case and `false` otherwise. Again, the adversary has access to the random oracle and a decryption oracle crippled to not work on the challenge ciphertext. We require that the adversary queries the oracles `INIT` and `ENC` only once. Furthermore `INIT` has to be queried before using any of the other oracles.

PUBLIC-KEY HIDING. Consider game $\mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(k)$ of Fig. 2 associated to PKE PE, adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(k)] - 1.$$

We say that PE is public-key hiding under parameter subversion attacks (also called PKH-PSA) if the function $\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(\cdot)$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. Unlike the prior games, two key pairs are generated, not one. The challenge ciphertext c^* is an encryption of a message of the adversary’s choice under pk_b . Given the public keys and the challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b . This time the crippled decryption oracle returns decryptions under both secret keys. The adversary sets up the public keys with its call to oracle `INIT`, and an uses oracle `ENC` to generate the challenge ciphertext. Again we require that the adversary queries the oracles `INIT` and `ENC` only once. Furthermore `INIT` has to be queried before using any of the other oracles.

RELATIONS. The following says that pseudorandomness of ciphertexts implies both indistinguishable encryptions and anonymity. We give both asymptotic and concrete statements of the results.

Proposition 1. *Let PE be a PKE that has pseudorandom ciphertexts under parameter subversion attacks. Then:*

1. PE is IND-PSA. Concretely, given an adversary \mathcal{A} the proof specifies an adversary \mathcal{B}_0 such that $\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{ind-psa}}(k) \leq 2 \cdot \mathbf{Adv}_{\text{PE},\mathcal{B}_0}^{\text{cpr-psa}}(k)$ for every $k \in \mathbb{N}$, and \mathcal{B}_0 has the same running time and query counts as \mathcal{A} .
2. PE is PKH-PSA. Concretely, given an adversary \mathcal{A} the proof specifies an adversary \mathcal{B}_1 such that $\mathbf{Adv}_{\text{PE},\mathcal{A}}^{\text{pkh-psa}}(k) \leq 2 \cdot \mathbf{Adv}_{\text{PE},\mathcal{B}_1}^{\text{cpr-psa}}(k)$ for every $k \in \mathbb{N}$, and \mathcal{B}_0 has the same running time and query counts as \mathcal{A} .

The proof of the proposition can be found in the full version of this paper [4].

3.2 Key Encapsulation Mechanisms

Below we first give a syntax for key encapsulation mechanisms. It follows [23] but with notation a bit different and including an additional algorithm setting

<u>Game $\mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{cpr-psa}}(k)$</u> $b \leftarrow \mathcal{S} \{0, 1\}$ $b' \leftarrow \mathcal{S} \mathcal{A}^{\text{INIT, DEC, RO}}(1^k)$ Return $(b = b')$	$\text{RO}(x, m) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{cpr-psa}}, \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{ind-psa}}, \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{pkh-psa}}$ If $(T[x, m] = \perp)$ then $T[x, m] \leftarrow \mathcal{S} \{0, 1\}^m$ Return $T[x, m]$
<u>Game $\mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{ind-psa}}(k)$</u> $b \leftarrow \mathcal{S} \{0, 1\}$ $b' \leftarrow \mathcal{S} \mathcal{A}^{\text{INIT, DEC, RO}}(1^k)$ Return $(b = b')$	$\text{INIT}(\pi) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{cpr-psa}}$ $(pk, sk) \leftarrow \mathcal{S} \mathbf{KE.G}(1^k, \pi)$ If $(pk = \perp)$ then return \perp If $(b = 1)$ then $(K^*, c^*) \leftarrow \mathcal{S} \mathbf{KE.E}^{\text{RO}}(1^k, \pi, pk)$ Else $K^* \leftarrow \mathcal{S} \mathbf{KE.KS}(k)$ $c^* \leftarrow \mathcal{S} \mathbf{KE.CS}(k, \pi)$ Return (pk, K^*, c^*)
<u>Game $\mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{pkh-psa}}(k)$</u> $b \leftarrow \mathcal{S} \{0, 1\}$ $b' \leftarrow \mathcal{S} \mathcal{A}^{\text{INIT, DEC, RO}}(1^k)$ Return $(b = b')$	$\text{INIT}(\pi) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{ind-psa}}$ $(pk, sk) \leftarrow \mathcal{S} \mathbf{KE.G}(1^k, \pi)$ If $(pk = \perp)$ then return \perp $(K^*, c^*) \leftarrow \mathcal{S} \mathbf{KE.E}^{\text{RO}}(1^k, \pi, pk)$ If $(b = 0)$ then $K^* \leftarrow \mathcal{S} \mathbf{KE.KS}(k)$ Return (pk, K^*, c^*)
<u>$\text{DEC}(c) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{cpr-psa}}, \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{ind-psa}}$</u> If $(c = c^*)$ then return \perp $K \leftarrow \mathbf{KE.D}^{\text{RO}}(1^k, \pi, sk, c)$ Return K	$\text{INIT}(\pi) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{pkh-psa}}$ $(pk_0, sk_0) \leftarrow \mathcal{S} \mathbf{KE.G}(1^k, \pi)$ $(pk_1, sk_1) \leftarrow \mathcal{S} \mathbf{KE.G}(1^k, \pi)$ If $(pk_0 = \perp \vee pk_1 = \perp)$ then return \perp $(K^*, c^*) \leftarrow \mathcal{S} \mathbf{KE.E}^{\text{RO}}(1^k, \pi, pk_b)$ Return (pk_0, pk_1, K^*, c^*)
<u>$\text{DEC}(c) // \mathbf{G}_{\mathbf{KE}, \mathcal{A}}^{\text{pkh-psa}}$</u> If $(c = c^*)$ then return \perp $K_0 \leftarrow \mathbf{KE.D}^{\text{RO}}(1^k, \pi, sk_0, c)$ $K_1 \leftarrow \mathbf{KE.D}^{\text{RO}}(1^k, \pi, sk_1, c)$ Return (K_0, K_1)	

Fig. 3. Games defining security of key encapsulation mechanism KE. In each game the adversary is given access to oracles. The game, to which an oracle belongs, is indicated behind the oracle’s name. In each game oracle INIT must be queried only once, which has to be done before using any of the other oracles.

up global parameters to be utilized by all users. As for public-key encryption schemes we formalize the security requirement of pseudorandomness of ciphertexts under parameter subversion attacks (CPR-PSA). We then adapt the two existing KEM requirements of indistinguishability of encryptions [23] and public-key hiding [1, 6] to the setting of parameter subversion attacks. In Proposition 2 we show that—as in the case of public-key encryption—the first requirement implies the other two. We furthermore define a new security requirement called well-distributedness of ciphertexts, which is necessary to achieve CPR-PSA in the hybrid PKE construction. It states that key-ciphertext pairs generated using the KEM’s encapsulation algorithm are indistinguishable from choosing a ciphertext at random and then computing its decapsulation.

KEMs. A *key encapsulation mechanism* (KEM) KE specifies the following. Parameter generation algorithm KE.P takes input 1^k , where $k \in \mathbb{N}$ is the security parameter, and returns global parameters π . Key-generation algorithm KE.G

takes input $1^k, \pi$ and returns a tuple (pk, sk) consisting of the public (encryption) key pk and matching secret (decryption) key sk . KE.KS associates to k a finite set $\text{KE.KS}(k)$ only depending on the security parameter that is the *key space* of KE . KE.CS associates to k and parameters π a finite set $\text{KE.CS}(k, \pi)$ that is the *ciphertext space* of KE . Encapsulation algorithm KE.E takes $1^k, \pi, pk$ and returns (K, c) where $K \in \text{KE.KS}(k)$ is the *encapsulated key* and $c \in \text{KE.CS}(k, \pi)$ is a ciphertext encapsulating K . Deterministic decapsulation algorithm KE.D takes $1^k, \pi, sk$ and a ciphertext c and returns either a key $K \in \text{KE.KS}(k)$ or the special symbol \perp indicating failure. The correctness condition requires that for all $k \in \mathbb{N}$, all $\pi \in [\text{KE.P}(1^k)]$ and all $(pk, sk) \in [\text{KE.G}(1^k, \pi)]$ we have $\Pr[\text{KE.D}(1^k, \pi, sk, c) = K] \geq 1 - \text{KE.de}(k)$, where the probability is over $(K, c) \leftarrow_s \text{KE.E}(1^k, \pi, pk)$ and $\text{KE.de} : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is the *decryption error* of KE . Our KEMs will be in the ROM [13], which means the encapsulation and decapsulation algorithms have access to a random oracle specified in the security games. Correctness must then hold for all choices of the random oracle. We say a KEM is *parameter-free* if KE.P returns ε on every input 1^k .

CIPHERTEXT PSEUDORANDOMNESS. Consider game $\mathbf{G}_{\text{KE}, \mathcal{A}}^{\text{cpr-psa}}(k)$ of Fig. 3 associated to KEM KE , adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{KE}, \mathcal{A}}^{\text{cpr-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{KE}, \mathcal{A}}^{\text{cpr-psa}}(k)] - 1.$$

We say that KE has pseudorandom ciphertexts under parameter subversion attacks (also called CPR-PSA) if the function $\mathbf{Adv}_{\text{KE}, \mathcal{A}}^{\text{cpr-psa}}(\cdot)$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. When $b = 1$, the challenge key K^* and ciphertext c^* are generated via the encapsulation algorithm, but if $b = 0$ they are chosen at random, from the key space and ciphertext space, respectively. Given the public key, challenge key and challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b , the game returning **true** in this case and **false** otherwise. The adversary has access to an oracle INIT , which sets up the challenge. We require that the adversary queries INIT before using any of the other oracles and that it queries INIT only once. Further the adversary has access to an oracle for decapsulation under sk , crippled to not work when invoked on the challenge ciphertext. It, and the encapsulation and decapsulation algorithms, have access to the random oracle RO . The parameters used in the game are provided by the adversary via its call to INIT .

INDISTINGUISHABILITY OF ENCAPSULATED KEYS FROM RANDOM. Consider game $\mathbf{G}_{\text{KE}, \mathcal{A}}^{\text{ind-psa}}(k)$ of Fig. 3 associated to KEM KE , adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{KE}, \mathcal{A}}^{\text{ind-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{KE}, \mathcal{A}}^{\text{ind-psa}}(k)] - 1.$$

We say that KE has encapsulated keys indistinguishable from random under parameter subversion attacks (also called IND-PSA) if the function $\mathbf{Adv}_{\text{KE}, \mathcal{A}}^{\text{ind-psa}}(\cdot)$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. When $b = 1$, the challenge key K^* and ciphertext c^* are generated via the encapsulation algorithm, while if $b = 0$ the key is switched to one drawn randomly from the key space,

the ciphertext remaining real. Given the public key, challenge key and challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b . Again the adversary has access to a crippled decapsulation oracle and the random oracle and provides the parameters used in the game via his call to the oracle INIT, which has to be queried before using any of the other oracles.

PUBLIC-KEY HIDING. Consider game $\mathbf{G}_{\text{KE},\mathcal{A}}^{\text{pkh-psa}}(k)$ of Fig. 3 associated to KEM KE, adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{pkh-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{KE},\mathcal{A}}^{\text{pkh-psa}}(k)] - 1.$$

We say that KE is public-key hiding under parameter subversion attacks (also called PKH-PSA) if the function $\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{pkh-psa}}(\cdot)$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. Unlike the prior games, two key pairs are generated, not one. The challenge key K^* and ciphertext c^* are generated via the encapsulation algorithm under pk_b . Given the public keys, challenge key and challenge ciphertext, the adversary outputs a guess b' and wins if b' equals b . This time the crippled decapsulation oracle returns decapsulations under both secret keys. Again the adversary provides the parameters to be used in the game via his single call to the oracle INIT, which has to be queried before using any of the other oracles.

RELATIONS. The following says that in the parameter subversion setting CPR-PSA implies both IND-PSA and PKH-PSA. We give both the asymptotic and concrete statements of the results.

Proposition 2. *Let KE be a KEM that has pseudorandom ciphertexts under parameter subversion attacks. Then:*

1. KE is IND-PSA. *Concretely, given an adversary \mathcal{A} the proof specifies an adversary \mathcal{B} such that $\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{ind-psa}}(k) \leq 2 \cdot \mathbf{Adv}_{\text{KE},\mathcal{B}}^{\text{cpr-psa}}(k)$ for every $k \in \mathbb{N}$, and \mathcal{B} has the same running time and query counts as \mathcal{A} .*
2. KE is PKH-PSA. *Concretely, given an adversary \mathcal{A} the proof specifies an adversary \mathcal{B} such that $\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{pkh-psa}}(k) \leq 2 \cdot \mathbf{Adv}_{\text{KE},\mathcal{B}}^{\text{cpr-psa}}(k)$ for every $k \in \mathbb{N}$, and \mathcal{B} has the same running time and query counts as \mathcal{A} .*

The proof of the proposition can be found in the full version of this paper [4].

WELL-DISTRIBUTED CIPHERTEXTS. Consider game $\mathbf{G}_{\text{KE},\mathcal{A}}^{\text{wdc-psa}}(k)$ of Fig. 4 associated to KEM KE, adversary \mathcal{A} and security parameter k , and let

$$\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{wdc-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{KE},\mathcal{A}}^{\text{wdc-psa}}(k)] - 1.$$

We say KE has well distributed ciphertexts under parameter subversion attacks (also called WDC-PSA), if the function $\mathbf{Adv}_{\text{KE},\mathcal{A}}^{\text{wdc-psa}}(\cdot)$ is negligible for every adversary \mathcal{A} . In the game b is a challenge bit. If b equals 1 the adversary as response to querying the initialization procedure, which may be done at most once, receives a key-ciphertext pair generated using KE.E. If b equals 0 it receives a pair (c^*, K^*) generated by choosing c^* at random and then setting K^* to be the decapsulation of c^* . The adversary has access to a decryption oracle. We require that the adversary queries INIT before querying any of the other oracles. Looking ahead, all of our instantiations achieve this notion statistically.

<p>Game $\mathbf{G}_{\text{KE}, \mathcal{A}}^{\text{wdc-psa}}(k)$</p> <p>$b \leftarrow_{\\$} \{0, 1\}$</p> <p>$b' \leftarrow_{\\$} \mathcal{A}^{\text{INIT, DEC, RO}}(1^k)$</p> <p>Return $(b = b')$</p> <hr/> <p>INIT(π)</p> <p>$(pk, sk) \leftarrow_{\\$} \text{KE.G}(1^k, \pi)$</p> <p>If $(pk = \perp)$ then return \perp</p> <p>If $(b = 1)$ then $(K^*, c^*) \leftarrow_{\\$} \text{KE.E}^{\text{RO}}(1^k, \pi, pk)$</p> <p>Else $c^* \leftarrow_{\\$} \text{KE.CS}(k, \pi)$</p> <p>$K^* \leftarrow \text{KE.D}^{\text{RO}}(1^k, \pi, sk, c^*)$</p> <p>Return (pk, K^*, c^*)</p>	<p>$\text{RO}(x, m)$</p> <p>If $(T[x, m] = \perp)$</p> <p style="padding-left: 20px;">then $T[x, m] \leftarrow_{\\$} \{0, 1\}^m$</p> <p>Return $T[x, m]$</p> <hr/> <p>DEC(c)</p> <p>If $(c = c^*)$ then return \perp</p> <p>$K \leftarrow \text{KE.D}^{\text{RO}}(1^k, \pi, sk, c)$</p> <p>Return K</p>
---	--

Fig. 4. Game defining well-distributedness of ciphertexts of KEs.

3.3 Symmetric Encryption

Below, we recall symmetric encryption. Our definition follows [23] but uses different notation. We further define the security notion of ciphertext pseudorandomness for symmetric encryption.

ONE-TIME SYMMETRIC-KEY ENCRYPTION. A symmetric-key encryption scheme (SKE) specifies the following. SE.KS associates to security parameter k key space $\text{SE.KS}(k)$. SE.CS associates to security parameter k and message length $m \in \mathbb{N}$ the ciphertext space $\text{SE.CS}(k, m)$. Deterministic encryption algorithm SE.E takes as input 1^k , key $K \in \text{SE.KS}(k)$ and a message $M \in \{0, 1\}^*$ and returns ciphertext $c \in \text{SE.CS}(k, |M|)$. Deterministic decryption algorithm SE.D on input $1^k, K \in \text{SE.KS}(k), c \in \text{SE.CS}(k, m)$ returns either a message $M \in \{0, 1\}^m$ or the special symbol \perp indicating failure. For correctness we require that $M = \text{SE.D}(1^k, K, c)$ for all k , all $K \in \text{SE.KS}(k)$ and all $M \in \{0, 1\}^*$, where $c \leftarrow \text{SE.E}(1^k, K, M)$.

ONE-TIME SECURITY. Consider game $\mathbf{G}_{\text{SE}, \mathcal{A}}^{\text{cpr}}(k)$ of Fig. 5 associated to SKE SE, adversary \mathcal{A} and security parameter k , and let

$$\text{Adv}_{\text{SE}, \mathcal{A}}^{\text{cpr}}(k) = 2 \Pr[\mathbf{G}_{\text{SE}, \mathcal{A}}^{\text{cpr}}(k)] - 1.$$

We say that SE has pseudorandom ciphertexts (also called CPR) if the function $\text{Adv}_{\text{SE}, \mathcal{A}}^{\text{cpr}}(\cdot)$ is negligible for every \mathcal{A} . We require that ENC is queried at most once.

3.4 PKE from Key Encapsulation and Symmetric-Key Encryption

Below, we analyze hybrid encryption in the setting of parameter subversion. Formally we give a transform $\mathbf{KEMToPE}$ that associates to KEM KE and symmetric-key encryption scheme SE a public-key encryption scheme PE. The construction essentially is the hybrid encryption scheme of [23] including an additional parameter generation algorithm. The scheme’s parameter generation, key

<u>Game $\mathbf{G}_{\text{SE}, \mathcal{A}}^{\text{cpr}}(k)$</u>	<u>$\text{ENC}(M)$</u>
$b \leftarrow_{\$} \{0, 1\}$	If $(b = 0)$ then $c^* \leftarrow_{\$} \text{SE.CS}(k, M)$
$K \leftarrow_{\$} \text{SE.KS}(k)$	Else $c^* \leftarrow \text{SE.E}(1^k, K, M)$
$b' \leftarrow \mathcal{A}^{\text{ENC, DEC}}(1^k)$	Return c^*
Return $(b = b')$	<u>$\text{DEC}(c)$</u>
	If $(c = c^*)$ then return \perp
	Else return $\text{SE.D}(1^k, K, c)$

Fig. 5. Game defining one-time security notions of SKEs.

<u>$\text{PE.P}(1^k)$</u>	<u>$\text{PE.E}(1^k, \pi, pk, M)$</u>
$\pi \leftarrow_{\$} \text{KE.P}(1^k)$	$(K, c_1) \leftarrow_{\$} \text{KE.E}^{\text{RO}}(1^k, \pi, pk)$
Return π	$c_2 \leftarrow \text{SE.E}(1^k, K, M)$
<u>$\text{PE.G}(1^k, \pi)$</u>	Return (c_1, c_2)
$(pk, sk) \leftarrow_{\$} \text{KE.G}(1^k, \pi)$	<u>$\text{PE.D}(1^k, \pi, sk, c)$</u>
Return (pk, sk)	$(c_1, c_2) \leftarrow c$
	$K \leftarrow \text{KE.D}^{\text{RO}}(1^k, \pi, sk, c_1)$
	$M \leftarrow \text{SE.D}(1^k, K, c_2)$
	Return M

Fig. 6. PKE $\mathbf{KEMToPE}[\text{KE}, \text{SE}]$ associated to KEM KE and SE SE.

generation encryption and decryption algorithms are in Fig. 6. PE's ciphertext space is given by $\text{PE.CS}(k, \pi, m) = \text{KE.CS}(k, \pi) \times \text{SE.CS}(k, m)$. It is easy to verify that PE has decryption error $\text{PE.de}(k) = \text{KE.de}(k)$. The following essentially states that hybrid encryption also works in setting of ciphertext pseudorandomness under parameter subversion attacks, i.e., combining a KEM that is both CPR-PSA and WDC-PSA with a SKE that is CPR yields a CPR-PSA PKE, where the well-distributedness of the KEM's ciphertext is necessary to correctly simulate the decryption oracle in the CPR-PSA game with respect to PE.

Proposition 3. *Let KE a KEM and SE a SE such that $\text{KE.KS}(k) = \text{SE.KS}(k)$ for all $k \in \mathbb{N}$. Let $\text{PE} = \mathbf{KEMToPE}[\text{KE}, \text{SE}]$ be the PKE associated to KE and SE. If KE is CPR-PSA and WDC-PSA and if SE is CPR then PE is CPR-PSA. Concretely, given adversary \mathcal{A} against $\mathbf{G}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(k)$, there exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ having the same running time and query count as \mathcal{A} , which satisfy*

$$\text{Adv}_{\text{PE}, \mathcal{A}}^{\text{cpr-psa}}(k) \leq 2 \text{Adv}_{\text{KE}, \mathcal{B}_1}^{\text{cpr-psa}}(k) + \text{Adv}_{\text{KE}, \mathcal{B}_2}^{\text{wdc-psa}}(k) + \text{Adv}_{\text{SE}, \mathcal{B}_3}^{\text{cpr}}(k) + \text{KE.de}(k).$$

The proof of the proposition can be found in the full version of this paper [4].

Game $\mathbf{G}_{\text{EG}, \mathcal{A}}^{\text{epr-psa}}(k)$	$\text{INT}(\pi)$
$b \leftarrow_{\$} \{0, 1\}$	$G \leftarrow_{\$} \text{EG.G}(1^k, \pi)$
$b' \leftarrow_{\$} \mathcal{A}^{\text{INT}}(1^k)$	If $(G = \perp)$ then return \perp
Return $(b = b')$	$(\langle \mathbb{G} \rangle, n, g) \leftarrow G$
	If $(b = 1)$ then
	$y \leftarrow_{\$} \text{EG.S}(1^k, \pi, G)$
	$c \leftarrow_{\$} \text{EG.E}(1^k, \pi, G, g^y)$
	Else $c \leftarrow_{\$} \text{EG.ES}(k, \pi)$
	Return (G, c)

Fig. 7. Game defining embedding pseudorandomness of eeg family EG.

4 KEMs from Efficiently Embeddable Group Families

In this section we define efficiently embeddable group families (eeg). We define the security notion of pseudorandom embeddings under parameter subversion attacks (EPR-PSA) and adapt the strong computational Diffie-Hellman problem (sCDH-PSA) to the setting of efficiently embeddable group families and parameter subversion. Further we give a generic constructions of key encapsulation mechanisms from eeg families. It achieves security assuming the eeg family is sCDH-PSA and EPR-PSA.

4.1 Efficiently Embeddable Group Families

EFFICIENTLY EMBEDDABLE GROUP FAMILIES. An embeddable group family EG specifies the following. Parameter generation algorithm EG.P takes as input 1^k , where $k \in \mathbb{N}$ is the security parameter, and returns parameters π . Group generation algorithm EG.G on input $1^k, \pi$ returns a tuple $G = (\langle \mathbb{G} \rangle, n, g)$, where $\langle \mathbb{G} \rangle$ is a description of a cyclic group \mathbb{G} of order n , and g is a generator of \mathbb{G} . EG.ES associates to k a finite set $\text{EG.ES}(k, \pi)$ called the embedding space that is only dependent on k and π . Sampling algorithm EG.S on input of $1^k, \pi$ and $G \in [\text{EG.G}(1^k, \pi)]$ outputs $y \in \mathbb{Z}_n$. (Not necessarily uniformly distributed.) Embedding algorithm EG.E receives as input $1^k, \pi, G \in [\text{EG.G}(1^k, \pi)]$ and $h \in \mathbb{G}$ and returns an element $c \in \text{EG.ES}(k, \pi)$. Deterministic inversion algorithm EG.I on input of $1^k, \pi, G \in [\text{EG.G}(1^k, \pi)]$ and $c \in \text{EG.ES}(k, \pi)$ returns an element of \mathbb{G} . The correctness condition requires that for all $k \in \mathbb{N}$, all $\pi \in \text{EG.P}(1^k)$ and all $G \in [\text{EG.G}(1^k, \pi)]$ we have $\Pr[\text{EG.I}(1^k, \pi, G, h) = g^y] \geq 1 - \text{EG.ie}(k)$, where the probability is over $y \leftarrow_{\$} \text{EG.S}(1^k, \pi, G)$ and $h \leftarrow_{\$} \text{EG.E}(1^k, \pi, G, g^y)$, and $\text{EG.ie} : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is the *inversion error* of EG. If EG.P returns ε on every input 1^k , i.e. if no parameters are used, we say that EG is *parameter-free*.

EMBEDDING PSEUDORANDOMNESS. Consider game $\mathbf{G}_{\text{EG}, \mathcal{A}}^{\text{epr-psa}}(k)$ of Fig. 7 associated to eeg family EG, adversary \mathcal{A} and security parameter k . Let

$$\text{Adv}_{\text{EG}, \mathcal{A}}^{\text{epr-psa}}(k) = 2 \Pr[\mathbf{G}_{\text{EG}, \mathcal{A}}^{\text{epr-psa}}(k)] - 1.$$

Game $\mathbf{G}_{\mathbf{EG}, \mathcal{A}}^{\text{scdh-psa}}(k)$	INIT(π)
$Z \leftarrow_{\$} \mathcal{A}^{\text{INIT, DDH}}(1^k)$	$G \leftarrow_{\$} \mathbf{EG.G}(1^k, \pi)$
Return ($Z = g^{xy} \wedge G \neq \perp$)	If ($G = \perp$) then return \perp
$\text{DDH}(\tilde{Y}, \tilde{Z})$	$(\langle \mathbb{G} \rangle, n, g) \leftarrow G$
Return ($\tilde{Y}^x = \tilde{Z}$)	$x \leftarrow_{\$} \mathbb{Z}_n$
	$y \leftarrow_{\$} \mathbf{EG.S}(1^k, \pi, G)$
	Return (G, g^x, g^y)

Fig. 8. Experiment for the strong computational Diffie-Hellman problem with respect to eeg family EG. Oracle INIT may be queried only once and has to be queried before using oracle DDH.

We say that EG has pseudorandom embeddings under parameter subversion attacks (also called EPR-PSA) if the function $\mathbf{Adv}_{\mathbf{EG}, \mathcal{A}}^{\text{epr-psa}}$ is negligible for every \mathcal{A} . In the game, b is a challenge bit. When $b = 1$, the challenge embedding c^* is generated by sampling an exponent using EG.S and embedding the group generator raised to the exponent with EG.E. If $b = 0$ the adversary is given an embedding sampled uniformly from the embedding space. Given the group and the embedding, the adversary outputs a guess b' and wins if b' equals b . The parameters used in the game are provided by the adversary making a single call to the oracle INIT. All of our instantiations sample exponents such that the resulting embeddings are statistically close to uniform on EG.ES(k, π), and hence achieve this notion statistically.

DIFFIE-HELLMAN PROBLEM WITH RESPECT TO EG. The computational Diffie-Hellman problem for a cyclic group \mathbb{G} of order n , which is generated by g , asks to compute g^{xy} given g^x and g^y , where $x, y \leftarrow_{\$} \mathbb{Z}_n$. In the strong computational Diffie-Hellman problem introduced by Abdalla et al. in [2] the adversary additionally has access to an oracle, which may be used to check whether $Y^x = Z$ for group elements $Y, Z \in \mathbb{G}$. We provide a definition for the strong computational Diffie-Hellman problem with respect to eeg families EG, which allows parameter subversion. An additional difference is that y is not chosen uniformly from \mathbb{Z}_n but instead sampled using EG.S.

Thus, consider game $\mathbf{G}_{\mathbf{EG}, \mathcal{A}}^{\text{scdh-psa}}(k)$ of Fig. 8. The game is associated to eeg family EG, adversary \mathcal{A} and security parameter k . The adversary has access to an oracle INIT setting up a problem instance according to the parameters it is provided. Let

$$\mathbf{Adv}_{\mathbf{EG}, \mathcal{A}}^{\text{scdh-psa}}(k) := \Pr \left[\mathbf{G}_{\mathbf{EG}, \mathcal{A}}^{\text{scdh-psa}}(k) \right].$$

We say that the strong computational Diffie-Hellman problem under parameter subversion (also called sCDH-PSA) is hard with respect to EG if $\mathbf{Adv}_{\mathbf{EG}, \mathcal{A}}^{\text{scdh-psa}}(\cdot)$ is negligible for every adversary \mathcal{A} .

$\text{KE.G}_1(1^k, \pi)$	$\text{KE.E}_1^{\text{RO}}(1^k, \pi, pk)$	$\text{KE.D}_1^{\text{RO}}(1^k, \pi, sk, c)$
$G \leftarrow_s \text{EG.G}(1^k, \pi)$	$(G, X) \leftarrow pk$	$(G, x, pk) \leftarrow sk$
If $(G = \perp)$ return \perp	$y \leftarrow_s \text{EG.S}(1^k, \pi, G)$	$Y \leftarrow \text{EG.I}(1^k, \pi, G, c)$
$(\langle \mathbb{G} \rangle, n, g) \leftarrow G$	$Y \leftarrow g^y$	$K \leftarrow \text{RO}((pk, c, Y^x), m(k))$
$x \leftarrow_s \mathbb{Z}_n; X \leftarrow g^x$	$c \leftarrow_s \text{EG.E}(1^k, \pi, G, Y)$	Return K
$pk \leftarrow (G, X)$	$K \leftarrow \text{RO}((pk, c, X^y), m(k))$	$\text{KE.P}_1(1^k)$
$sk \leftarrow (G, x, pk)$	Return (K, c)	$\pi \leftarrow_s \text{EG.P}(1^k)$
Return (pk, sk)		Return π

Fig. 9. KEM $\text{KE}_1 = \text{eegToKE1}[\text{EG}, m]$ built from eeg family EG and polynomial m via our transform. The KE has key space $\text{KE.KS}(k) = \{0, 1\}^{m(k)}$ and ciphertext space $\text{KE.CS}(k, \pi) = \text{EG.ES}(k, \pi)$.

4.2 Key Encapsulation from Efficiently Embeddable Group Families

In this section we give a generic construction of a key encapsulation mechanism from an eeg family EG. Its security is based on the strong Diffie-Hellman problem, i.e. if sCDH-PSA is hard with respect to EG, the KEM is IND-PSA. If additionally EG has pseudorandom embeddings, the KEM has pseudorandom and well-distributed ciphertexts. The construction is similar to the standard El Gamal based key encapsulation mechanism as for example used in [2, 23]. As an intermediate step in the proof that the construction is CPR-PSA we obtain that it is IND-PSA. The proof of this property follows the outlines of the proofs given in [2, 23]. Afterwards we use the pseudorandomness of the eeg family’s embeddings to show, that our construction achieves pseudorandom and well-distributed ciphertexts.

Formally, we define a transform **eegToKE1** that associates to an eeg family EG and a polynomial $m : \mathbb{N} \rightarrow \mathbb{N}$ a KEM $\text{KE} = \text{eegToKE1}[\text{EG}, m]$. The parameter generation, key generation, encryption and decryption algorithms of KE are in Fig. 9. The construction is in the ROM, so that encryption and decryption invoke the RO oracle. The key space is $\text{KE.KS}(k) = \{0, 1\}^{m(k)}$. The ciphertext space $\text{KE.CS}(k, \pi) = \text{EG.ES}(k, \pi)$ is the embedding space of EG. It is easy to verify that $\text{KE.de} = \text{EG.ie}$, meaning the decryption error of the KEM equals the inversion error of the eeg family.

SECURITY OF THE CONSTRUCTION. The following says that if sCDH-PSA is hard with respect to eeg family EG then **eegToKE1**[EG, m] has desirable security properties.

Theorem 4. *Let $\text{KE} = \text{eegToKE1}[\text{EG}, m]$ be the KEM associated to eeg family EG and polynomial $m : \mathbb{N} \rightarrow \mathbb{N}$ as defined in Fig. 9. Assume that EG is EPR-PSA and that sCDH-PSA is hard with respect to EG. Then*

- (i) KE has pseudorandom ciphertexts under parameter subversion attacks.
- (ii) KE has well-distributed ciphertexts under parameter subversion attacks.

Moreover, if EG is parameter-free so is KE. Concretely, given an adversary \mathcal{A} making at most $q(k)$ queries to RO the proof specifies adversaries \mathcal{B}_1 and \mathcal{B}_2 having the same running time as \mathcal{A} satisfying

$$\text{Adv}_{\text{KE}}^{\text{cpr-psa}}(\mathcal{A})(k) \leq \text{Adv}_{\text{EG}, \mathcal{B}_1}^{\text{scdh-psa}}(k) + \text{Adv}_{\text{EG}, \mathcal{B}_2}^{\text{epr-psa}}(k),$$

where \mathcal{B}_2 makes at most $q(k)$ queries to DDH. Furthermore given an adversary \mathcal{A}' the proof specifies an adversary \mathcal{B}' having the same running time as \mathcal{A}' such that,

$$\text{Adv}_{\text{KE}, \mathcal{A}'}^{\text{wdc-psa}}(k) \leq \text{Adv}_{\text{EG}, \mathcal{B}'}^{\text{epr-psa}}(k) + \text{EG.ie}(k).$$

The proof of the theorem can be found in the full version of this paper [4]. In the full version of this paper [4] we also provide a transform **eegToKE2**, which achieves security under the weaker CDH-PSA assumption with respect to EG.

5 Efficiently Embeddable Group Families from Curve-Twist Pairs

In this section we give instantiations of eeg families based on elliptic curves. The main tool of the constructions is a bijection of [34] mapping points of an elliptic curve and its quadratic twist to an interval of integers. We first give a construction using parameters, the parameter being a prime p of length k serving as the modulus of the prime field the curves are defined over. The construction has embedding space $[2p + 1]$. Since we assume, that the parameter shared by all users might be subject to subversion, security of this construction corresponds to the assumption that there exist no inherently bad choices for p , i.e. that for any sufficiently large prime p it is possible to find elliptic curves defined over \mathbb{F}_p on which the strong computational Diffie-Hellman assumption holds.

As an alternative we also give parameter-free eeg-families whose security is based on the weaker assumption that for random k -bit prime p it is possible to find elliptic curves defined over \mathbb{F}_p , such that the strong computational Diffie-Hellman assumption holds. Since in this construction the modulus p is sampled along with the curve, it is no longer possible to use $[2p + 1]$ as the embedding space of the eeg family. We propose two solutions to overcome this, one using rejection sampling to restrict the embedding space to the set $[2^k]$, the other one is based on a technique from [33] and expands the embedding space to $[2^{k+1}]$.

5.1 Elliptic Curves

Let $p \geq 5$ be prime and \mathbb{F}_p a field of order p . An elliptic curve over \mathbb{F}_p can be expressed in short Weierstrass form, that is as the set of projective solutions of an equation of the form

$$YZ^2 = X^3 + aXZ^2 + bZ^3,$$

where $a, b \in \mathbb{F}_p$ with $4a^3 + 27b^2 \neq 0$. We denote the elliptic curve generated by p, a, b by $E(p, a, b)$. $E(p, a, b)$ possesses exactly one point with Z -coordinate 0,

the so called point at infinity $\mathcal{O} = (0 : 1 : 0)$. After normalizing by $Z = 1$ the curve's other points can be interpreted as the solutions $(x, y) \in \mathbb{F}_p^2$ of the affine equation $y^2 = x^3 + ax + b$. It is possible to establish an efficiently computable group law on $E(p, a, b)$ with \mathcal{O} serving as the neutral element of the group. We use multiplicative notation for the group law to be consistent with the rest of the paper.

TWISTS OF ELLIPTIC CURVES. In [34, Sect. 4] Kaliski establishes the following one-to-one correspondence between two elliptic curves defined over \mathbb{F}_p which are related by twisting and a set of integers.

Lemma 5. *Let $p \in \mathbb{N}_{\geq 5}$ be prime. Let $u \in \mathbb{Z}_p$ be a quadratic nonresidue modulo p and $a, b \in \mathbb{Z}_p$ such that $4a^3 + 27b^2 \neq 0$. Consider the elliptic curves $E_0 := E(p, a, b)$ and $E_1 := E(p, au^2, bu^3)$. Then $|E_0| + |E_1| = 2p + 2$. Furthermore, the functions $l_0 : E_0 \rightarrow [2p + 2]$ and $l_1 : E_1 \rightarrow [2p + 2]$ defined as*

$$l_0(P) = \begin{cases} p & \text{if } P = \mathcal{O}_0 \\ (ux \bmod p) & \text{if } (P = (x, y)) \wedge (0 \leq y \leq (p - 1)/2), \\ ((ux \bmod p) + p + 1) & \text{if } (P = (x, y)) \wedge ((p - 1)/2 < y) \end{cases}$$

$$l_1(P) = \begin{cases} 2p + 1 & \text{if } P = \mathcal{O}_1 \\ x & \text{if } (P = (x, y)) \wedge (0 < y \leq (p - 1)/2) \\ x + p + 1 & \text{if } (P = (x, y)) \wedge ((y = 0) \vee ((p - 1)/2 < y)) \end{cases}$$

are injective with nonintersecting ranges, where \mathcal{O}_0 and \mathcal{O}_1 denote the neutral elements of E_0 and E_1 respectively.

Lemma 6. *The functions l_0 and l_1 can be efficiently inverted. That is, given $z \in [2p + 1]$, one can efficiently compute the unique $(P, \delta) \in E_0 \cup E_1 \times \{0, 1\}$ such that $l_\delta(P) = z$.*

The proof of the lemma can be found in the full version of this paper [4].

Definition 7. *A curve-twist generator TGen on input of security parameter 1^k and a k -bit prime p returns (G_0, G_1) , where $G_0 = (\langle E_0 \rangle, n_0, g_0)$ and $G_1 = (\langle E_1 \rangle, n_1, g_1)$ are secure cyclic elliptic curves defined over the field \mathbb{F}_p . More precisely we require $E_0 := E(p, a, b)$ and $E_1 := E(p, au^2, bu^3)$ for $a, b \in \mathbb{F}_p$ such that $(4a^3 + 27b^2) \neq 0$ and quadratic nonresidue u . Furthermore we require that g_0 generates E_0 and g_1 generates E_1 as well as $|E_0| = n_0$, $|E_1| = n_1$ and $\text{gcd}(n_0, n_1) = 1$.*

GENERATION OF SECURE TWISTED ELLIPTIC CURVES. There exist several proposals for properties an elliptic curve over a prime field \mathbb{F}_p should have to be considered secure (e.g., [18, 27]). Firstly, the elliptic curve's order is required to be either the product of a big prime and a small cofactor—or preferably prime. Secondly, several conditions preventing the transfer of discrete logarithm problems on the curve to groups, where faster algorithms to compute discrete

logarithms may be applied, should be fulfilled. Finally, for our applications we need both the elliptic curve and its quadratic twist to be secure, a property usually called twist security. For concreteness, we suggest to implement $\text{TGen}(1^k, p)$ by sampling the necessary parameters a, b, u with rejection sampling such that the resulting curve $E(p, a, b)$ fulfills the three security requirement mentioned above. This way, TGen can be implemented quite efficiently¹ and furthermore, with overwhelming probability, the resulting curve fulfills all relevant security requirements from [18, 27] that are not covered by the three security properties explicitly mentioned above.

COMPUTATIONAL PROBLEMS ASSOCIATED TO TGen . Let TGen a curve-twist generator. We give two versions of the strong computational Diffie-Hellman assumption with respect to TGen . In the first version the prime p on which TGen is invoked is chosen by the adversary, while in the second version p is sampled uniformly at random from all k -bit primes. For $d \in \{0, 1\}$ consider games $\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-cp-scdh}}(\cdot)$ and $\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-up-scdh}}(\cdot)$ of Fig. 10. We define advantage functions

$$\begin{aligned} \text{Adv}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-cp-scdh}}(k) &= \Pr \left[\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-cp-scdh}}(k) \right], \\ \text{Adv}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-up-scdh}}(k) &= \Pr \left[\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-up-scdh}}(k) \right]. \end{aligned}$$

Definition 8. *Let TGen be a curve-twist generator. We say the strong computational Diffie-Hellman assumption for chosen (uniform) primes holds with respect to curve-twist generator TGen , if both $\text{Adv}_{\text{TGen}, \mathcal{A}}^{\text{twist}_0\text{-cp-scdh}}(\cdot)$ and $\text{Adv}_{\text{TGen}, \mathcal{A}}^{\text{twist}_1\text{-cp-scdh}}(\cdot)$ (or $\text{Adv}_{\text{TGen}, (P_k)_{k, \mathcal{A}}}^{\text{twist}_0\text{-up-scdh}}(\cdot)$ and $\text{Adv}_{\text{TGen}, (P_k)_{k, \mathcal{A}}}^{\text{twist}_1\text{-up-scdh}}(\cdot)$ respectively) are negligible for all adversaries \mathcal{A} .*

5.2 An Eeg Family from Elliptic Curves

In [34] Kaliski implicitly gives an eeg family based on elliptic curves. The family is parameter-using, the parameter being a prime p serving as the modulus of the field the elliptic curves are defined over. The definition of eeg family EG_{twist} may be found in Fig. 11. Parameter generation algorithm $\text{EG}_{\text{twist}} \cdot \text{P}$ on input of security parameter 1^k returns a randomly sampled k -bit prime² p . Group generation algorithm $\text{EG}_{\text{twist}} \cdot \text{G}$ on input of parameter $\pi = p$ checks, whether p is

¹ In [29] Galbraith and McKee consider elliptic curves E chosen uniformly from the set of elliptic curves over a fixed prime field \mathbb{F}_p . They give a conjecture (together with some experimental evidence) for a lower bound on the probability of $|E|$ being prime. Using a similar technique [27] argue, that the probability of a uniformly chosen elliptic curve over a fixed prime field \mathbb{F}_p to be both secure and twist secure is bounded from below by $0.5/\log^2(p)$. Since their definition of security of an elliptic curve includes primality of the curve order and since due to Lemma 5 the orders of curve and twist sum up to $2p + 2$, this in particular implies that the curve and its twist are cyclic and have coprime group order.

² In practice one would preferably instantiate EG_{twist} with a standardized prime.

Game $\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-cp-scdh}}(k)$	$\text{INIT}(\pi)$
$Z \leftarrow \mathcal{A}^{\text{INIT}, \text{DDH}}(1^k)$	$p \leftarrow \pi$
Return $(Z = g_d^{xy})$	If $(p \notin \mathcal{P}_k)$ then return \perp
$\text{DDH}(\tilde{Y}_d, \tilde{Z}_d)$	$(G_0, G_1) \leftarrow \text{TGen}(1^k, p)$
If $\tilde{Y}_d \notin E_d \vee \tilde{Z}_d \notin E_d$	$(\langle E_d \rangle, n_d, g_d) \leftarrow G_d$
return \perp	$x \leftarrow \mathbb{Z}_{n_d}; y \leftarrow \mathbb{Z}_{n_d}$
Return $(\tilde{Y}_d^x = \tilde{Z}_d)$	$X \leftarrow g_d^x, Y \leftarrow g_d^y$
	Return (G_0, G_1, X, Y)

Game $\mathbf{G}_{\text{TGen}, \mathcal{A}}^{\text{twist}_d\text{-up-scdh}}(k)$	INIT
$Z \leftarrow \mathcal{A}^{\text{INIT}, \text{DDH}}(1^k)$	$p \leftarrow \mathcal{P}_k$
Return $(Z = g_d^{xy})$	$(G_0, G_1) \leftarrow \text{TGen}(1^k, p)$
$\text{DDH}(\tilde{Y}_d, \tilde{Z}_d)$	$(\langle E_d \rangle, n_d, g_d) \leftarrow G_d$
If $(\tilde{Y}_d \notin E_d \vee \tilde{Z}_d \notin E_d)$	$x \leftarrow \mathbb{Z}_{n_d}; y \leftarrow \mathbb{Z}_{n_d}$
return \perp	$X \leftarrow g_d^x, Y \leftarrow g_d^y$
Return $(\tilde{Y}_d^x = \tilde{Z}_d)$	Return (G_0, G_1, p, X, Y)

Fig. 10. Experiments for the sCDH problem for chosen (uniform) primes with respect to $d \in \{0, 1\}$, adversary \mathcal{A} and curve-twist generator TGen .

indeed a prime of appropriate length, and—if so—runs a curve-twist generator $\text{TGen}(1^k, \pi)$ to obtain the description of two cyclic secure cyclic elliptic curves $G_0 = (\langle E_0 \rangle, n_0, g_0)$ and $G_1 = (\langle E_1 \rangle, n_1, g_1)$. Its output is $(\langle \mathbb{G} \rangle, n, g)$, where $\mathbb{G} \leftarrow E_0 \times E_1$ is the direct product of the two elliptic curves, $n \leftarrow n_0 \cdot n_1$ and $g \leftarrow (g_0, g_1)$. Here we assume that the description $\langle \mathbb{G} \rangle$ of \mathbb{G} includes the values n_0 and n_1 , which are used by EG_{twist} 's other algorithms. Note that $|\mathbb{G}| = n$ and since n_0 and n_1 are coprime, g generates \mathbb{G} . Furthermore, if we regard E_0 and E_1 as subgroups of $\mathbb{G} = E_0 \times E_1$ in the natural way, we may rewrite the set $E_0 \cup E_1 \subseteq \mathbb{G}$ as

$$\begin{aligned} E_0 \cup E_1 &= \{(h_0, \mathcal{O}_1) \mid h_0 \in E_0\} \cup \{(\mathcal{O}_0, h_1) \mid h_1 \in E_1\} \\ &= \{(g_0, g_1)^y \mid y \in \mathbb{Z}_n : y \equiv 0 \pmod{n_0} \text{ or } y \equiv 0 \pmod{n_1}\} \end{aligned}$$

Algorithm $\text{EG}_{\text{twist}}.\text{S}$ uses this property to efficiently sample $y \in \mathbb{Z}_n$ such that $g^y \sim U_{E_0 \cup E_1}$. It first samples $z \leftarrow \mathbb{Z}_{2p+1}$. If $z < n_0$ it returns $\varphi_{\text{crt}}(z, 0)$. Else it returns $\varphi_{\text{crt}}(0, z - n_0 - 1)$. Here φ_{crt} denotes the canonical isomorphism $\varphi_{\text{crt}} : \mathbb{Z}_{n_0} \times \mathbb{Z}_{n_1} \rightarrow \mathbb{Z}_n$. As a result $y \leftarrow \text{EG}_{\text{twist}}.\text{S}(1^k, G)$ satisfies $y \sim U_M$, where $M := \{y \in \mathbb{Z}_n \mid y \equiv 0 \pmod{n_0} \text{ or } y \equiv 0 \pmod{n_1}\}$. Embedding algorithm $\text{EG}_{\text{twist}}.\text{E}$ receives as input $1^k, \pi, G$ and $h = (h_0, h_1) \in \mathbb{G}$. It first checks, whether h lies outside of the support $[\text{EG}_{\text{twist}}.\text{S}(1^k, \pi, G)]$ of the sampling algorithm, i.e. whether both $h_0 \neq \mathcal{O}_0$ and $h_1 \neq \mathcal{O}_1$. In this case the element is mapped to 0. If h is an element of $[\text{EG}_{\text{twist}}.\text{S}(1^k, \pi, G)]$, algorithm $\text{EG}_{\text{twist}}.\text{E}$ returns $l_0(h_0)$ if $h_1 = \mathcal{O}_1$, and $l_1(h_1)$ if $h_1 \neq \mathcal{O}_1$. Here $l_0 : E_0 \rightarrow [2p+2]$ and $l_1 : E_1 \rightarrow [2p+2]$ denote the maps of Lemma 5. By Lemma 5 the map $\text{EG}_{\text{twist}}.\text{E}(1^k, G, \cdot)|_{E_0 \cup E_1}$ is

$\underline{\text{EG}_{\text{twist}}.\text{P}(1^k)}$ $p \leftarrow \text{\$ } \mathcal{P}_k$ $\pi \leftarrow p$ $\text{Return } \pi$ $\underline{\text{EG}_{\text{twist}}.\text{G}(1^k, \pi)}$ $p \leftarrow \pi$ $\text{If } (p \notin \mathcal{P}_k) \text{ return } \perp$ $(G_0, G_1) \leftarrow \text{\$ } \text{TGen}(1^k, p)$ $(\langle E_0 \rangle, g_0, n_0) \leftarrow G_0; (\langle E_1 \rangle, g_1, n_1) \leftarrow G_1$ $\mathbb{G} \leftarrow E_0 \times E_1; g \leftarrow (g_0, g_1); n \leftarrow n_0 \cdot n_1$ $G \leftarrow (\langle \mathbb{G} \rangle, n, g)$ $\text{Return } G$	$\underline{\text{EG}_{\text{twist}}.\text{S}(1^k, \pi, G)}$ $p \leftarrow \pi$ $z \leftarrow \text{\$ } \mathbb{Z}_{2p+1}$ $\text{If } (z < n_0) \text{ return } \varphi_{\text{crt}}(z, 0)$ $\text{Else return } \varphi_{\text{crt}}(0, z - n_0 - 1)$ $\underline{\text{EG}_{\text{twist}}.\text{E}(1^k, \pi, G, (h_0, h_1))}$ $\text{If } (h_0 \neq \mathcal{O}_0 \wedge h_1 \neq \mathcal{O}_1) \text{ return } 0$ $\text{Elseif } h_1 = \mathcal{O}_1 \text{ return } l_0(h_0)$ $\text{Else return } l_1(h_1)$ $\underline{\text{EG}_{\text{twist}}.\text{I}(1^k, \pi, G, z)}$ $\text{If } (z \in \text{im}(l_0)) \text{ return } l_0^{-1}(z)$ $\text{Else return } l_1^{-1}(z)$
--	--

Fig. 11. Definition of eeg family EG_{twist} with embedding space $\text{EG}_{\text{twist}}.\text{ES}(k, \pi) = [2p + 1]$. l_0 and l_1 denote the maps from Lemma 5, φ_{crt} the canonical isomorphism $\mathbb{Z}_{n_0} \times \mathbb{Z}_{n_1} \rightarrow \mathbb{Z}_n$.

a bijection between $E_0 \cup E_1$ and $[2p + 1]$ and we obtain $\text{EG}_{\text{twist}}.\text{E}(1^k, G, g^y) \sim U_{[2p+1]}$ for y sampled with $\text{EG}_{\text{twist}}.\text{S}(1^k, G)$. We obtain the following.

Lemma 9. EG_{twist} as defined in Fig. 11 is an eeg family with embedding space $\text{EG}_{\text{twist}}.\text{ES}(k, G) = [2p + 1]$ and inversion error $\text{EG}_{\text{twist}}.\text{ie}(k) = 0$. Furthermore EG_{twist} has pseudorandom embeddings. More precisely, for every (potentially unbounded) adversary \mathcal{A} we have

$$\text{Adv}_{\text{EG}_{\text{twist}}, \mathcal{A}}^{\text{epf-psa}}(k) = 0.$$

A proof of the lemma can be found in the full version of the paper [4]. Concerning the hardness of sCDH-PSA with respect to EG_{twist} we obtain the following.

Lemma 10. Let EG_{twist} be the embeddable group generator constructed with respect to twisted elliptic curve generator TGen as described above. If the strong Diffie-Hellman assumption for chosen primes holds with respect to TGen , then the strong Diffie-Hellman assumption holds with respect to EG_{twist} .

Concretely for every adversary \mathcal{A} against game $\mathbf{G}_{\text{EG}_{\text{twist}}, \mathcal{A}}^{\text{scdh-psa}}(\cdot)$, which makes at most Q queries to its DDH-oracle, there exist adversaries $\mathcal{B}_0, \mathcal{B}_1$ against games $\mathbf{G}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-cp-scdh}}(\cdot)$ or $\mathbf{G}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-cp-scdh}}(\cdot)$ respectively making at most Q queries to their DDH-oracles, satisfying

$$\text{Adv}_{\text{EG}_{\text{twist}}, \mathcal{A}}^{\text{scdh-psa}}(k) \leq \text{Adv}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-cp-scdh}}(k) + \text{Adv}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-cp-scdh}}(k).$$

The proof of the lemma can be found in the full version of this paper [4].

5.3 A Parameter-Free Eeg Family Using Rejection Sampling

Eeg family EG_{twist} of Sect. 5.2 is parameter-using, the parameter being the size p of the field \mathbb{F}_p . Correspondingly, hardness of sCDH-PSA with respect to EG_{twist}

$\text{EG}_{\text{twist-rs}}^\ell \cdot \text{P}(1^k)$	$\text{EG}_{\text{twist-rs}}^\ell \cdot \text{S}(1^k, \pi, G)$	$\text{EG}_{\text{twist-rs}}^\ell \cdot \text{E}(1^k, \pi, G, h)$
Return ε	$(G', p) \leftarrow G$	$(G', p) \leftarrow G'$
$\text{EG}_{\text{twist-rs}}^\ell \cdot \text{G}(1^k, \pi)$	For $\ell^* = 1$ to ℓ	$z \leftarrow_{\$} \text{EG}_{\text{twist}} \cdot \text{E}(1^k, p, G', h)$
$p \leftarrow_{\$} \mathcal{P}_k$	Do $y \leftarrow \text{EG}_{\text{twist}} \cdot \text{S}(1^k, p, G')$	Return z
$G' \leftarrow_{\$} \text{EG}_{\text{twist}} \cdot \text{G}(1^k, p)$	If $(\text{EG}_{\text{twist}} \cdot \text{E}(1^k, p, G, g^y) < 2^k)$	$\text{EG}_{\text{twist-rs}}^\ell \cdot \text{I}(1^k, \pi, G, z)$
$G \leftarrow (G', p)$	return y	$(G', p) \leftarrow G'$
Return G	Return \perp	$h \leftarrow \text{EG}_{\text{twist}} \cdot \text{I}(1^k, p, G', z)$
		Return h

Fig. 12. Parameter-free eeg family $\text{EG}_{\text{twist-rs}}^\ell$.

follows from the assumption, that the elliptic curves output by curve-twist generator TGen are secure, independently of the prime p the curve-twist generator TGen is instantiated with. In this section we show how EG_{twist} can be used to construct an eeg family $\text{EG}_{\text{twist-rs}}^\ell$ for which hardness of sCDH-PSA follows from the weaker assumption that TGen instantiated with a *randomly* chosen prime is able to sample secure elliptic curves. The construction is parameter-free and has embedding space $[2^k]$. The size p of the field over which the elliptic curves are defined is now sampled as part of the group generation. The embedding algorithm uses rejection sampling to ensure that embeddings of group elements g^y for y sampled with $\text{EG}_{\text{twist-rs}}^\ell \cdot \text{S}$ are elements of $[2^k]$. The specification of $\text{EG}_{\text{twist-rs}}^\ell$'s algorithms may be found in Fig. 12.

Theorem 11. *Let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial. $\text{EG}_{\text{twist-rs}}^\ell$ as described above is an eeg family with embedding space $\text{EG}_{\text{twist-rs}}^\ell \cdot \text{ES}(k, \pi) = [2^k]$ and inversion error $\text{EG}_{\text{twist-rs}}^\ell \cdot \text{ie}(k) \leq 2^{-\ell(k)}$. Furthermore $\text{EG}_{\text{twist-rs}}^\ell$ has pseudorandom embeddings. More precisely, for every (potentially unbounded) adversary \mathcal{A} we have*

$$\text{Adv}_{\text{EG}_{\text{twist-rs}}^\ell, \mathcal{A}}^{\text{epr-psa}}(k) \leq 2^{-\ell(k)}.$$

The proof of the theorem can be found in the full version of this paper [4]. As discussed above, we obtain that—assuming that TGen invoked on randomly sampled prime p returns a secure curve-twist pair—the sCDH-PSA-problem with respect to eeg family $\text{EG}_{\text{twist-rs}}^\ell$ is hard.

Lemma 12. *Let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial and $\text{EG}_{\text{twist-rs}}^\ell$ the eeg family with underlying curve-twist generator TGen as described above. If the sCDH assumption for uniform primes holds with respect to TGen , then sCDH-PSA is hard with respect to $\text{EG}_{\text{twist-rs}}^\ell$. Concretely, for every adversary \mathcal{A} against game $\mathbf{G}_{\text{EG}_{\text{twist-rs}}^\ell, \mathcal{A}}^{\text{scdh-psa}}(\cdot)$ making at most Q queries to its DDH-oracle there exist adversaries $\mathcal{B}_0, \mathcal{B}_1$ against $\mathbf{G}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-up-scdh}}(\cdot)$ or $\mathbf{G}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-up-scdh}}(\cdot)$ respectively, making at most Q queries to their DDH-oracles and running in the same time as \mathcal{A} , which satisfy*

$$\text{Adv}_{\text{EG}_{\text{twist-rs}}^\ell, \mathcal{A}}^{\text{scdh-psa}}(k) \leq 3 \left(\text{Adv}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-up-scdh}}(k) + \text{Adv}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-up-scdh}}(k) \right) + 2^{-\ell(k)}$$

for all $k \in \mathbb{N}_{\geq 6}$.

$\underline{\text{EG}_{\text{twist-re}}.\text{G}(1^k, \pi)}$ $p \leftarrow_{\$} \mathcal{P}_k$ $G' \leftarrow_{\$} \text{EG}_{\text{twist-re}}.\text{G}(1^k, p); G \leftarrow (G', p)$ $\text{Return } G$ $\underline{\text{EG}_{\text{twist-re}}.\text{S}(1^k, \pi, G)}$ $(G', p) \leftarrow G$ $z \leftarrow_{\$} [2^{k+1}]$ $\text{If } (z \leq 2p)$ $y \leftarrow \psi_G(z)$ $\text{If } (\text{EG}_{\text{twist-re}}.\text{E}(1^k, p, G', g^y) < 2^{k+1} - (2p + 1))$ $\quad \text{return } y$ $\text{Else } z \leftarrow z - (2p + 1)$ $y \leftarrow \psi_G(z)$ $\text{Return } y$	$\underline{\text{EG}_{\text{twist-re}}.\text{P}(1^k)}$ $\text{Return } \varepsilon$ $\underline{\text{EG}_{\text{twist-re}}.\text{E}(1^k, \pi, G, h)}$ $(G', p) \leftarrow G; b \leftarrow_{\$} \{0, 1\}$ $z \leftarrow \text{EG}_{\text{twist-re}}.\text{E}(1^k, p, G', h)$ $\text{If } z < 2^{k+1} - (2p + 1)$ $\quad z \leftarrow z + b(2p + 1)$ $\text{Return } z$ $\underline{\text{EG}_{\text{twist-re}}.\text{I}(1^k, \pi, G, z)}$ $(G', p) \leftarrow G$ $\text{If } (z \geq 2p + 1)$ $\quad z \leftarrow z - (2p + 1)$ $h \leftarrow \text{EG}_{\text{twist-re}}.\text{I}(1^k, p, G', z)$ $\text{Return } h$
--	--

Fig. 13. Definition of eeg family $\text{EG}_{\text{twist-re}}$ with embedding space $\text{EG}_{\text{twist-re}}.\text{ES}(k, \pi) := [2^{k+1}]$. ψ_G denotes the bijection $[2p + 1] \rightarrow [\text{EG}_{\text{twist-re}}.\text{S}(1^k, p, G')]$ defined in Sect. 5.4.

The proof of the lemma can be found in the full version of this paper [4].

5.4 A Parameter-Free Family Using Range Expansion

In this section we modify the algorithms of EG_{twist} to obtain an embeddable group family $\text{EG}_{\text{twist-re}}$ with embedding space $\text{EG}_{\text{twist-re}}.\text{ES}(k, \pi) = [2^{k+1}]$. The eeg family has inversion error $\text{EG}_{\text{twist-re}}.\text{ie}(k) = 0$ and achieves uniformly distributed embeddings. The construction is building on a technique introduced by Hayashi et al. [33], where it is used to expand the range of one way permutations. As in Sect. 5.3, the hardness sCDH-PSA with respect to $\text{EG}_{\text{twist-re}}$ is based on the hardness of the sCDH problem for uniform primes with respect to TGen. The sampling algorithm—in contrast to the construction based on rejection sampling—needs access to only one uniformly random sampled integer, performs at most one exponentiation in the group and uses at most one evaluation of $\text{EG}_{\text{twist}}.\text{E}$ to output y with the correct distribution. Furthermore, exponents sampled by $\text{EG}_{\text{twist-re}}.\text{S}$ are distributed such that the eeg family achieves $\text{EG}_{\text{twist-re}}.\text{ie}(k) = 0$ and for every (potentially unbounded) adversary \mathcal{A} we additionally have $\text{Adv}_{\text{EG}_{\text{twist-re}}, \mathcal{A}}^{\text{ep-PSA}}(k) = 0$.

The description of $\text{EG}_{\text{twist-re}}$ may be found in Fig. 13. We now discuss the construction in greater detail. Let $(G', p) = G \in [\text{EG}_{\text{twist-re}}.\text{G}(k, \pi)]$, where $G' = ((\mathbb{G}), n, g)$. The idea of the construction is to partition $[\text{EG}_{\text{twist-re}}.\text{S}(1^k, p, G')]$ into two sets M_1, M_2 with $M_1 \cup M_2 = [\text{EG}_{\text{twist-re}}.\text{S}(1^k, p, G')]$, $\{\text{EG}_{\text{twist-re}}.\text{E}(1^k, p, G', g^y) \mid y \in M_1\} = \{2^{k+1} - (2p + 1), \dots, 2p\}$ and $\{\text{EG}_{\text{twist-re}}.\text{E}(1^k, p, G', g^y) \mid y \in M_2\} = \{0, \dots, 2^{k+1} - (2p + 2)\}$. The sampling algorithm $\text{EG}_{\text{twist-re}}.\text{S}$ is constructed such that for y sampled by $\text{EG}_{\text{twist-re}}.\text{S}(1^k, \pi, G)$, the probability $\Pr[y = y']$ equals 2^{-k} for all $y' \in M_2$ and $2^{-(k+1)}$ for all $y' \in M_1$. Embedding algorithm $\text{EG}_{\text{twist-re}}.\text{E}$

on input $(1^k, \pi, G, h)$ first computes $c \leftarrow \text{EG}_{\text{twist}} \cdot \text{E}(1^k, p, G', h)$. If $c \in \{2^{k+1} - (2p + 1), \dots, 2p\}$ its output remains unchanged. Otherwise it is shifted to $\{2p + 1, \dots, 2^{k+1} - 1\}$ with probability $1/2$. In this way we achieve embeddings, which are uniformly distributed on $\text{EG}_{\text{twist-re}} \cdot \text{ES}(k, \pi) = [2^{k+1}]$.

Our construction relies on the existence of a bijection $\psi_G : [2p + 1] \rightarrow [\text{EG}_{\text{twist}} \cdot \text{S}(1^k, p, G')]$ for all $(G', p) = G \in [\text{EG}_{\text{twist-re}} \cdot \text{G}(1^k, \pi)]$. We use the bijection, which was implicitly given in the definition of $\text{EG}_{\text{twist}} \cdot \text{S}$. That is, for $z \in [2p + 1]$ we define

$$\psi_G(z) := \begin{cases} \varphi_{\text{crt}}(z, 0) & \text{if } z < n_0 \\ \varphi_{\text{crt}}(0, z - n_0 - 1) & \text{else,} \end{cases}$$

where φ_{crt} denotes the canonical isomorphism $\mathbb{Z}_{n_0} \times \mathbb{Z}_{n_1} \rightarrow \mathbb{Z}_n$.

Theorem 13. *$\text{EG}_{\text{twist-re}}$ as specified in Fig. 13 is an embeddable group family with embedding space $\text{EG}_{\text{twist-re}} \cdot \text{ES}(k, \pi) = [2^{k+1}]$ and inversion error $\text{EG}_{\text{twist-re}} \cdot \text{ie}(k) = 0$. Furthermore $\text{EG}_{\text{twist-re}}$ has pseudorandom embeddings. More precisely, for every (potentially unbounded) adversary \mathcal{A} we have*

$$\text{Adv}_{\text{EG}_{\text{twist-re}}, \mathcal{A}}^{\text{epr-psa}}(k) = 0.$$

The proof of the theorem can be found in the full version of this paper [4]. As in the case of $\text{EG}_{\text{twist-rs}}^\ell$, we obtain that—assuming that TGen invoked on randomly sampled prime p returns a secure curve-twist pair— sCDH-PSA with respect to eeg family $\text{EG}_{\text{twist-re}}$ is hard.

Lemma 14. *Let $\text{EG}_{\text{twist-re}}$ be the eeg family defined above with underlying curve-twist generator TGen . If the sCDH assumption holds with respect to TGen , then sCDH-PSA is hard with respect to $\text{EG}_{\text{twist-re}}$. Concretely, for every adversary \mathcal{A} against $\mathbf{G}_{\text{EG}_{\text{twist-re}}, \mathcal{A}}^{\text{scdh-psa}}(\cdot)$ making at most Q queries to its DDH -oracle there exist adversaries $\mathcal{B}_0, \mathcal{B}_1$ against $\mathbf{G}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-up-scdh}}(\cdot)$ or $\mathbf{G}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-up-scdh}}(\cdot)$ respectively running in the same time as \mathcal{A} and making at most Q queries to their DDH -oracles, which satisfy*

$$\text{Adv}_{\text{EG}_{\text{twist-re}}, \mathcal{A}}^{\text{scdh-psa}}(k) \leq 2 \left(\text{Adv}_{\text{TGen}, \mathcal{B}_0}^{\text{twist}_0\text{-up-scdh}}(k) + \text{Adv}_{\text{TGen}, \mathcal{B}_1}^{\text{twist}_1\text{-up-scdh}}(k) \right).$$

The proof of the lemma can be found in the full version of this paper [4].

Acknowledgments. Benedikt Auerbach was supported by the NRW Research Training Group SecHuman. Mihir Bellare was supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft corporation. Eike Kiltz was supported in part by ERC Project ERCC FP7/615074 and by DFG SPP 1736 Big Data.

References

1. Abdalla, M., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_13
2. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_12
3. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signature schemes. In: Ray, I., Li, N., Kruegel, N. (eds.) ACM CCS 15: 22nd Conference on Computer and Communications Security, pp. 364–375. ACM Press, October 2015
4. Auerbach, B., Bellare, M., Kiltz, E.: Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. Cryptology ePrint Archive, Report 2018/023 (2018). <http://eprint.iacr.org/2018/023>
5. Baignères, T., Delerablée, C., Finiasz, M., Goubin, L., Lepoint, T., Rivain, M.: Trap me if you can - million dollar curve. Cryptology ePrint Archive, Report 2015/1249 (2015). <http://eprint.iacr.org/2015/1249>
6. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
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
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., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: security in the face of parameter subversion. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 Part II. LNCS, vol. 10032, pp. 777–804. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_26
10. Bellare, M., Hoang, V.T.: Resisting randomness subversion: fast deterministic and hedged public-key encryption in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015 Part II. LNCS, vol. 9057, pp. 627–656. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_21
11. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: strongly undetectable algorithm-substitution attacks. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 15: 22nd Conference on Computer and Communications Security, pp. 1431–1440. ACM Press, October 2015
12. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014 Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_1
13. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 93: 1st Conference on Computer and Communications Security, pp. 62–73. ACM Press, November 1993

14. 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
15. Bernstein, D.J., Chou, T., Chuengsatiansup, C., Hülsing, A., Lange, T., Niederhagen, R., van Vredendaal, C.: How to manipulate curve standards: a white paper for the black hat. Cryptology ePrint Archive, Report 2014/571 (2014). <http://eprint.iacr.org/2014/571>
16. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_9
17. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13: 20th Conference on Computer and Communications Security, pp. 967–980. ACM Press, November 2013
18. Bernstein, D.J., Lange, T.: SafeCurves: choosing safe curves for elliptic-curve cryptography. <https://safecurves.cr.yt.to>. Accessed 18 May 2016
19. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: a standardized back door. Cryptology ePrint Archive, Report 2015/767 (2015). <http://eprint.iacr.org/2015/767>
20. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: how to use an imperfect reference string. In: 48th Annual Symposium on Foundations of Computer Science, pp. 249–259. IEEE Computer Society Press, October 2007
21. Checkoway, S., Cohnsey, S., Garman, C., Green, M., Heninger, N., Maskiewicz, J., Rescorla, E., Shacham, H., Weinmann, R.-P.: A systematic analysis of the juniper dual EC incident. In: Proceedings of the 23rd ACM conference on Computer and communications security. ACM (2016)
22. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055717>
23. 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)
24. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 579–598. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_28
25. Degabriele, J.P., Paterson, K.G., Schuldt, J.C.N., Woodage, J.: Backdoors in pseudorandom number generators: possibility and impossibility results. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016 Part I. LNCS, vol. 9814, pp. 403–432. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_15
26. Dodis, Y., Ganesha, C., Golovnev, A., Juels, A., Ristenpart, T.: A formal treatment of backdoored pseudorandom generators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015 Part I. LNCS, vol. 9056, pp. 101–126. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_5
27. Flori, J.-P., Plüt, J., Reinhard, J.-R., Ekerå, M.: Diversity and transparency for ECC. Cryptology ePrint Archive, Report 2015/659 (2015). <http://eprint.iacr.org/>
28. Frey, G.: How to disguise an elliptic curve (Weil descent). Talk given at ECC 1998 (1998)

29. Galbraith, S.D., McKee, J.: The probability that the number of points on an elliptic curve over a finite field is prime. *J. Lond. Math. Soc.* **62**(3), 671–684 (2000)
30. Garg, S., Goyal, V., Jain, A., Sahai, A.: Bringing people of different beliefs together to do UC. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 311–328. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_19
31. Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
32. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_18
33. Hayashi, R., Okamoto, T., Tanaka, K.: An RSA family of trap-door permutations with a common domain and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 291–304. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24632-9_21
34. Kaliski Jr., B.S.: One-way permutations on elliptic curves. *J. Cryptol.* **3**(3), 187–199 (1991)
35. Katz, J., Kiayias, A., Zhou, H.-S., Zikas, V.: Distributing the setup in universally composable multi-party computation. In: Halldórsson, M.M., Dolev, S. (eds.) 33rd ACM Symposium Annual on Principles of Distributed Computing, pp. 20–29. Association for Computing Machinery, July 2014
36. Lochter, M., Meikle, J.: RFC 5639: ECC Brainpool Standard Curves & Curve Generation. Internet Engineering Task Force, March 2010
37. Möller, B.: A public-key encryption scheme with pseudo-random ciphertexts. In: Samarati, P., Ryan, P., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 335–351. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30108-0_21
38. NIST: Digital signature standard (DSS) 2013. FIPS PUB 186–4
39. Orman, H.: The OAKLEY key determination protocol (1998)
40. Petit, C., Quisquater, J.-J.: On polynomial systems arising from a Weil descent. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 451–466. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_28
41. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Cliptography: clipping the power of kleptographic attacks. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016 Part II. LNCS, vol. 10032, pp. 34–64. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_2
42. Russell, A., Tang, Q., Yung, M., Zhou, H.-S.: Generic semantic security against a kleptographic adversary. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 17: 24th Conference on Computer and Communications Security, pp. 907–922. ACM Press, October 2017
43. Young, A., Yung, M.: The dark side of “black-box” cryptography, or: should we trust capstone? In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)
44. Young, A., Yung, M.: Kleptography: using cryptography against cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_6

Cryptanalysis



A Practical Cryptanalysis of WalnutDSA™

Daniel Hart¹, DoHoon Kim¹, Giacomo Micheli¹, Guillermo Pascual-Perez¹,
Christophe Petit²(✉) , and Yuxuan Quek¹

¹ University of Oxford, Oxford, UK

² University of Birmingham, Birmingham, UK
christophe.f.petit@gmail.com

Abstract. We present a practical cryptanalysis of WalnutDSA, a digital signature algorithm trademarked by SecureRF. WalnutDSA uses techniques from permutation groups, matrix groups and braid groups, and is designed to provide post-quantum security in lightweight IoT device contexts. The attack given in this paper bypasses the E-Multiplication™ and cloaked conjugacy search problems at the heart of the algorithm and forges signatures for arbitrary messages in approximately two minutes. We also discuss potential countermeasures to the attack.

1 Introduction

Most of the cryptosystems in use today are based on two difficult problems: the integer factorization problem and the Discrete Logarithm Problem (DLP). Both of these problems can be solved efficiently by running Shor's algorithm [1] on a sufficiently large quantum computer. As of now, such quantum computers do not exist, but organisations such as NIST and the NSA are striving for cryptosystems resilient to quantum attacks to prepare for the time when they become a reality [2–4].

The problem at the heart of Shor's algorithms, the so-called hidden subgroup problem, can be solved in polynomial time on a quantum computer for any finite abelian group, but has so far appeared much harder in the case for non-abelian groups. Cryptography based on non-abelian groups is therefore considered an appealing direction for post-quantum cryptography. Braid groups have traditionally been used in non-abelian group based cryptography: for example, the Anshel-Anshel-Goldfeld (AAG) key-exchange protocol and the Diffie-Hellman-type key-exchange protocol are both based on the conjugacy search problem (or at least one of its variants) in a braid group [5, Sect. 1.6]. Today, more advanced protocols have evolved from these schemes.

SecureRF [6] is a corporation founded in 2004 specializing in security for the Internet of Things (IoT), i.e. devices with low processing power that require ultra-low energy consumption, whose partners include the US Air Force. WalnutDSA [7] is a digital signature algorithm developed by SecureRF that was presented at the NIST Lightweight Cryptography Workshop in 2016. SecureRF has collaborated with Intel [8] to develop an implementation of WalnutDSA

on secure field-programmable gate arrays (FPGAs). Thus, WalnutDSA's importance as a cryptosystem today is established, as corporations and government agencies push for security in a post-quantum world.

1.1 Our Contribution

We provide a universal forgery attack on WalnutDSA. Our attack does not require a signing oracle: in fact, having access to a small set of random message-signature pairs suffice. In principle, the security of WalnutDSA is based on the difficulty of reversing E-Multiplication and the cloaked conjugacy search problem [7, Problems 1, 2], but we go around this by reducing the problem of forging a WalnutDSA signature to an instance of the factorization problem in a non-abelian group (given a group element $g \in G$ and a generating set Γ for G , find a word w over Γ such that $w = g$). While this problem is plausibly hard in general, we give an efficient algorithm for solving the particular instance occurring in this context. Given a couple of valid signatures on random messages, our attack can produce a new signature on an arbitrary message in approximately two minutes. We also discuss countermeasures to prevent this attack.

Responsible Disclosure Process. Since WalnutDSA is advertised as a security product by SecureRF, we notified its authors of our findings before making them available to the public. We informed them by email on October 17th 2017 with full details of our attack. They acknowledged the effectiveness of our attack on October 19th 2017, and we agreed to postpone our publication until November 26th 2017.

Two countermeasures are discussed here, namely checking the signature length and increasing the parameters. SecureRF have communicated to us that they have always had a limit on signature lengths in their product offerings, and that the increase in parameter sizes we suggest may still allow for many applications in devices with limited computing power. These two countermeasures can prevent our attack for now. As we briefly argue in Sect. 5 below, improved versions of the attack might be able to defeat them, but we leave these to further work.

In reaction to our attack, SecureRF have also developed a new version of WalnutDSA using two private keys (instead of conjugation), such that Proposition 4 of this paper fails to apply.

1.2 Related Work

Ben-Zvi et al. [9] provide a complete attack on a version of SecureRF's Algebraic Eraser scheme, a public key encryption protocol also based on E-Multiplication. Other attacks on the Algebraic Eraser include those by Myasnikov and Ushakov [10], which is a length-based attack on SecureRF's specific realisation of the general scheme, and by Kalka et al. [11], which is a cryptanalysis for arbitrary parameter sizes.

Other important work includes Garside's and Birman et al. [12, 13] on solving the conjugacy search problem in braid groups using Summit Sets, the Garside normal form [12] and Dehornoy Handle Reduction [14].

Other instances of factorization problems in non-abelian groups have been solved previously, in both cryptographic contexts [15–17] and in mathematical literature [18]. The algorithms we develop in this paper for factorization in $\text{GL}_N(\mathbb{F}_q)$ belongs to the family of subgroup attacks [19].

1.3 Outline

In Sect. 2, we provide the definition of security for signature schemes, and introduce the factorization problem as well as some preliminary results about braid groups. In Sect. 3, we introduce the WalnutDSA protocol. In Sect. 4, we provide a cryptanalysis of WalnutDSA by first reducing the problem to a factorization problem in $\text{GL}_N(\mathbb{F}_q)$ (Sect. 4.1) and then solving it (Sect. 4.2). In Sect. 5, we describe possible countermeasures to prevent the attack. We conclude the paper in Sect. 6.

2 Preliminaries

2.1 Security Definition

The standard security definition for signatures is *existential unforgeability under chosen message attacks* [20, Introduction]. An adversary can ask for polynomially many signatures of messages of its choice to a signing oracle. The attack is then considered successful if the attacker is able to produce a valid pair of message and signature for a message different from those queried to the oracle. We will show that the version of WalnutDSA proposed in [7] is not resistant to this kind of attack and propose a modification to the scheme that fixes this weakness.

Definition 1. A signature scheme $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ is said to be existentially unforgeable under adaptive chosen-message attacks (or secure, for short) if for all probabilistic polynomial time adversaries \mathcal{A} with access to $\text{Sign}_{\text{SK}}(\cdot)$,

$$\left| \Pr \left[\begin{array}{l} (\text{PK}, \text{SK}) \leftarrow \text{Gen}(1^\lambda); s_i \leftarrow \text{Sign}_{\text{SK}}(m_i) \text{ for } 1 \leq i \leq k; \\ (m, s) \leftarrow \mathcal{A}(\text{PK}, (m_i)_{i=1}^k, (s_i)_{i=1}^k) : \\ \text{Verify}_{\text{PK}}(m, s) = 1 \text{ and } m \notin \mathcal{M} \end{array} \right] \right| \leq \text{negl}(\lambda).$$

where $\mathcal{M} = \{m_1, \dots, m_k\}$ is the set of messages queried by \mathcal{A} to the oracle, and $k = \#\mathcal{M}$ is polynomial in the security parameter λ .

For our cryptanalysis, the m_i can actually be random messages, leading to a stronger attack.

2.2 Braid Groups

For $N \geq 2$, the braid group [5] on N strands, denoted B_N , is a group with presentation

$$B_N = \left\langle b_1, \dots, b_{N-1} \mid \begin{array}{l} b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1} \\ b_i b_j = b_j b_i \text{ for } |i - j| \geq 2 \end{array} \right\rangle, \tag{1}$$

where the b_i are called *Artin generators*. There are other presentations for the braid group, but unless otherwise stated, we will use the definition provided in (1) and “generators” will refer to the Artin generators. Geometrically, the elements of a braid group are the equivalence classes of N strands under ambient isotopy, and the group operation is concatenation of the N strands. More precisely, the generator b_i corresponds to the $(i + 1)$ -th strand crossing over the i -th strand. Note that there is a natural homomorphism from B_N onto the symmetric group S_N : if $\beta = b_{i_1} \cdots b_{i_k}$, then the permutation induced by β is precisely

$$\prod_{j=1}^k (i_j, i_j + 1),$$

where $(i_j, i_j + 1)$ is the standard transposition in S_N .

Notation. Let $\mathfrak{p}: B_N \rightarrow S_N$ be the above map, which sends a braid to its induced permutation.

Braids that induce trivial permutations are called *pure braids*. The set of pure braids is exactly the kernel of the homomorphism \mathfrak{p} , hence it forms a normal subgroup of B_N . We will denote this subgroup by PB_N .

Garside Normal Form. A normal form of an element in a group is a canonical way to represent the element. One known normal form for braid groups is *Garside normal form*. The details can be found in Appendix A. We can compute the Garside normal form of a braid with complexity $O(|W|^2 N \log N)$ where $|W|$ is the length of the word in Artin generators [21]. Such a normal form is important for WalnutDSA, but the cryptanalysis we provide in Sect. 4 is independent of the choice of it.

The Colored Burau Representation. Let q be an arbitrary prime power, and let \mathbb{F}_q be the finite field with q elements. Let $\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}]$ be the ring of Laurent polynomials with coefficients in \mathbb{F}_q . Note that there is a natural action of S_N on $\text{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$, where a permutation acts on a Laurent polynomial by permuting its variables. In other words, we have an action $f \mapsto \sigma f$ where $f(t_1, \dots, t_N)$ is mapped to $f(t_{\sigma(1)}, \dots, t_{\sigma(N)})$. Similarly, a permutation may act on a matrix M in $\text{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$ entrywise, and we will denote the image of M under this action as ${}^\sigma M$.

Proposition 1. *There exists a group homomorphism, called the colored Burau representation [7],*

$$\Phi: B_N \rightarrow \text{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}]) \rtimes S_N,$$

where \rtimes denotes the semidirect product.

Let \mathbf{m} be the projection of Φ on $\text{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$. Then Φ is defined as follows:

– For the generator $b_1 \in B_N$, define

$$\mathbf{m}(b_1) = \begin{pmatrix} -t_1 & 1 & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$

and

$$\mathbf{m}(b_1^{-1}) = \begin{pmatrix} -\frac{1}{t_2} & \frac{1}{t_2} & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}.$$

– For $2 \leq i < N$, define

$$\mathbf{m}(b_i) = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & t_i & -t_i & 1 \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$

where the $-t_i$ occurs in the i -th row. Also define

$$\mathbf{m}(b_i^{-1}) = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & -\frac{1}{t_{i+1}} & \frac{1}{t_{i+1}} \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}.$$

– Define

$$\Phi(b_i) := (\mathbf{m}(b_i), \mathbf{p}(b_i)).$$

– Given generators $b_i^{\pm 1}, b_j^{\pm 1}$, we define $\Phi(b_i^{\pm 1}b_j^{\pm 1})$ to be

$$(\mathbf{m}(b_i^{\pm 1}), \mathbf{p}(b_i)) \cdot (\mathbf{m}(b_j^{\pm 1}), \mathbf{p}(b_j)) = \left(\mathbf{m}(b_i^{\pm 1}) \cdot (\mathbf{p}(b_i)\mathbf{m}(b_j^{\pm 1})), \mathbf{p}(b_i)\mathbf{p}(b_j) \right).$$

For a general braid β , we extend this definition inductively to define $\Phi(\beta)$.

Note that Φ and \mathbf{p} are homomorphisms, but \mathbf{m} is not a homomorphism in general. However, the following lemma shows that its restriction to pure braids is a homomorphism.

Lemma 1. *Let $\phi : PB_N \rightarrow GL_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$ be the restriction map of \mathbf{m} to PB_N . This map is a group homomorphism.*

Proof. Let β_1, β_2 be pure braids. Then, if Id_{S_N} is the identity permutation,

$$\begin{aligned} \phi(\beta_1\beta_2) &= \mathbf{m}(\beta_1\beta_2) \\ &= \mathbf{m}(\beta_1) \cdot (\text{Id}_{S_N}\mathbf{m}(\beta_2)) \\ &= \mathbf{m}(\beta_1)\mathbf{m}(\beta_2) = \phi(\beta_1)\phi(\beta_2), \end{aligned}$$

and so ϕ is indeed a homomorphism. □

Previous Cryptosystems Based on Braid Groups. A problem that is generally difficult to solve in non-abelian groups is the conjugacy search problem (CSP), i.e. given conjugate elements $u, w \in B_N$, find $v \in B_N$ such that $w = v^{-1}uv$. This motivated the development of several cryptosystems based on the CSP in braid groups, some of which are given in [5]. Techniques such as summit sets [13, 22, 23], length-based attacks [24–26], and linear representations [27–29], have been developed to attack the CSP in braid groups however, and so those cryptosystems have been rendered impractical. The design of WalnutDSA uses a variant of the CSP, the cloaked conjugacy search problem, to avoid these attacks.

2.3 Factorization Problem in Non-Abelian Groups

Factorization Problem in Groups. Let G be a group, let $\Gamma = \{g_1, \dots, g_\gamma\}$ be a generating set for G , and let $h \in G$. Find a “small” integer L and sequences $(m_1, \dots, m_L) \in \{1, \dots, \gamma\}^L$ and $(\epsilon_1, \dots, \epsilon_L) \in \{\pm 1\}^L$ such that

$$h = \prod_{i=1}^L g_{m_i}^{\epsilon_i}.$$

Depending on the context, “small” may refer to a concrete practical size, or it may mean polynomial in $\log |G|$. The existence of products of size polynomial in $\log |G|$ for any finite simple non-abelian group, any generating set, and any element h was conjectured by Babai and Seress [30]. This conjecture has attracted considerable attention from the mathematics community in the last fifteen years, and has now been proven for many important groups [31, 32].

The potential hardness of the factorization problem for non-abelian groups underlies the security of Cayley hash functions [33]. The problem was solved in the particular cases of the Zémor [34, 35], Tillich-Zémor [15, 17, 36], and Charles-Goren-Lauter [16, 37, 38] hash functions, and to a large extent in the case of symmetric and alternating groups [18], but it is still considered a potentially hard problem in general. Over cyclic groups, this problem is known to be equivalent to the discrete logarithm problem when removing the constraint on L [39]. We refer to [19] for a more extensive discussion of the factorization problem and its connection with Babai’s conjecture.

The instance of the factorization problem that appears in our attack is over $GL_N(\mathbb{F}_q)$, the general linear group of rank N over the finite field \mathbb{F}_q . Our solution for it exploits the particular subgroup structure of this group.

3 WalnutDSA

WalnutDSATM is a digital signature scheme proposed by Anshel et al. in [7], based on braid groups, E-MultiplicationTM and cloaked conjugacy.

3.1 E-Multiplication

Let B_N be the braid group on N braids, let q be a prime power and let \mathbb{F}_q^\times denote the non-zero elements of the finite field \mathbb{F}_q . Define a sequence of “T-values”:

$$\tau = (\tau_1, \tau_2, \dots, \tau_N) \in (\mathbb{F}_q^\times)^N.$$

Given the T-values, we can evaluate any Laurent polynomial $f \in \mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}]$ to produce an element of \mathbb{F}_q :

$$f \downarrow_\tau := f(\tau_1, \dots, \tau_N).$$

We can similarly evaluate any matrix M in $GL_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$ entrywise to produce a matrix $M \downarrow_\tau$ in $GL_N(\mathbb{F}_q)$.

E-Multiplication [40] is a right action, denoted by \star , of the colored Burau group $GL_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}]) \rtimes S_N$ on $GL_N(\mathbb{F}_q) \times S_N$. In other words, it takes two ordered pairs

$$\begin{aligned} (M, \sigma_0) &\in GL_N(\mathbb{F}_q) \times S_N, \\ (\mathbf{m}(\beta), \mathbf{p}(\beta)) &\in GL_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}]) \rtimes S_N, \end{aligned}$$

and produces another ordered pair

$$(M', \sigma') = (M, \sigma_0) \star (\mathbf{m}(\beta), \mathbf{p}(\beta))$$

in $GL_N(\mathbb{F}_q) \times S_N$.

E-Multiplication is defined inductively. For a single generator b_i ,

$$(M, \sigma_0) \star (\mathbf{m}(b_i), \mathbf{p}(b_i)) := \left(M \cdot \sigma_0(\mathbf{m}(b_i)) \downarrow_\tau, \sigma_0 \cdot \mathbf{p}(b_i) \right).$$

For a general braid $\beta = b_{i_1}^{\epsilon_1} \cdots b_{i_k}^{\epsilon_k}$,

$$(M, \sigma_0) \star (\mathbf{m}(\beta), \mathbf{p}(\beta)) = (M, \sigma_0) \star (\mathbf{m}(b_{i_1}^{\epsilon_1}), \mathbf{p}(b_{i_1}^{\epsilon_1})) \star \cdots \star (\mathbf{m}(b_{i_k}^{\epsilon_k}), \mathbf{p}(b_{i_k}^{\epsilon_k})),$$

where the successive E-Multiplications are performed left to right. This is well-defined, as it is independent of how we write β in terms of the generators [7, Sect. 3].

Lemma 2. *For any pure braid β , any permutation σ , and any $\tau \in (\mathbb{F}_q^\times)^N$, $\left((\sigma \mathbf{m}(s_i)) \downarrow_\tau \right)^{-1} = (\sigma \mathbf{m}(s_i^{-1})) \downarrow_\tau$.*

Proof. Let $M \in \text{GL}_N(\mathbb{F}_q)$ and let $\sigma \in S_N$. Then,

$$(M, \sigma) = (M, \sigma) \star (s_i \cdot s_i^{-1}) = (M \cdot^\sigma \mathbf{m}(s_i) \downarrow_\tau \cdot^\sigma \mathbf{m}(s_i^{-1}) \downarrow_\tau, \sigma),$$

which implies

$$\left((\sigma \mathbf{m}(s_i)) \downarrow_\tau \right)^{-1} = (\sigma \mathbf{m}(s_i^{-1})) \downarrow_\tau.$$

□

Notation. We will follow the notation in [7] and write

$$(M, \sigma_0) \star \beta$$

instead of $(M, \sigma_0) \star (\mathbf{m}(\beta), \mathbf{p}(\beta))$ for a braid $\beta \in B_N$.

Notation. For $\xi = (M, \sigma)$ in $\text{GL}_N(\mathbb{F}_q) \times S_n$, let $\mathbf{m}(\xi)$ denote the matrix part of ξ , i.e. $\mathbf{m}(\xi) = M$.

3.2 Key Generation

Before the signer generates the private-/public-key pair, some public parameters are fixed:

- An integer N and the associated braid group B_N ;
- A rewriting algorithm $\mathcal{R} : B_N \rightarrow B_N$, such as the Garside normal form;
- A prime power q defining a finite field \mathbb{F}_q of q elements;
- Two integers $1 < a < b < N$;
- T-values $\tau = (\tau_1, \tau_2, \dots, \tau_N) \in (\mathbb{F}_q^\times)^N$ with $\tau_a = \tau_b = 1$;
- An encoding function $\mathcal{E} : \{0, 1\}^* \rightarrow B_N$ taking messages to braids.

The signer then chooses a random freely-reduced braid $\text{SK} \in B_N$ (of the desired length to prevent brute force attacks from being effective) to be the private-key, and calculates the public-key as

$$\text{PK} = (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK}.$$

Notation. We follow the notation in [7] and write $\text{Pub}(\beta) := (\text{Id}_N, \text{Id}_{S_N}) \star \beta$ for a braid $\beta \in B_N$.

In [7], it is recommended to use $N \geq 8$ and $q \geq 32$ for the public parameters.

3.3 Message Encoding

To sign a message $m \in \{0, 1\}^*$ using WalnutDSA, it must first be encoded as a braid $\mathcal{E}(m) \in B_N$. WalnutDSA achieves this by encoding messages as pure braids: given a message m , it is first hashed using a cryptographically secure hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^{4\kappa}$, where $\kappa \geq 1$. The paper [7] does not provide a formal definition of “cryptographically secure”, but we believe that the intended meaning is that of a “random oracle” [41], and in this paper we will treat the hash function as such. The bitstring $H(m)$ is then encoded as a pure braid by noting that the $N - 1$ braids

$$\begin{aligned} g_{(N-1),N} &= b_{N-1}^2, \\ g_{(N-2),N} &= b_{N-1} \cdot b_{N-2}^2 \cdot b_{N-1}^{-1}, \\ &\vdots \\ g_{1,N} &= b_{N-1} b_{N-2} \cdots b_2 \cdot b_1^2 b_2^{-1} b_3^{-1} \cdots b_{N-1}^{-1} \end{aligned}$$

are pure braids that freely generate a subgroup of B_N [42]. Fix four of these generators, say $g_{k_1,N}, g_{k_2,N}, g_{k_3,N}, g_{k_4,N}$ for $1 \leq k_i \leq N - 1$, and define

$$C = \langle g_{k_1,N}, g_{k_2,N}, g_{k_3,N}, g_{k_4,N} \rangle \subset PB_N.$$

Each 4-bit block of $H(m)$ can then be mapped to a unique power of one of these generators: the first two bits determine the generator $g_{k_i,N}$ to use, while the last two bits determine the power $1 \leq i \leq 4$ to raise the generator to. The encoded message $\mathcal{E}(m) \in C$ is then defined to be the freely reduced product of the κ powers of the $g_{k_i,N}$ obtained via the above process.

3.4 Cloaking Elements

WalnutDSA defines and uses “cloaking elements” to avoid being reduced to the conjugacy search problem, reducing instead to the cloaked conjugacy search problem. A braid $\beta \in B_N$ is said to be a cloaking element of $(M, \sigma) \in \text{GL}_N(\mathbb{F}_q) \times S_N$ if $(M, \sigma) \star \beta = (M, \sigma)$. The set of cloaking elements of (M, σ) is then the stabilizer of (M, σ) under the E-Multiplication action, and so forms a subgroup of B_N .

Lemma 3. *Any cloaking element is a pure braid.*

Proof. Let $\beta \in B_N$ be a cloaking element of $(M, \sigma) \in \text{GL}_N(\mathbb{F}_q) \times S_N$. Then

$$(M, \sigma) = (M, \sigma) \star \beta = \left(M \cdot \sigma(\mathfrak{m}(\beta)) \downarrow_{\tau}, \sigma \cdot \mathfrak{p}(\beta) \right),$$

which implies that $\mathfrak{p}(\beta) = \text{Id}_{S_N}$. □

The authors of WalnutDSA provide a method of generating cloaking elements [7, Proposition 4.2], which we recap here for the reader’s convenience.

Proposition 2. Fix integers $N \geq 2$ and $1 < a < b < N$. Assume that $\tau_a = \tau_b = 1$. Let $M \in \text{GL}_N(\mathbb{F}_q)$ and $\sigma \in S_N$. Then a cloaking element β of (M, σ) is given by $\beta = wb_i^2w^{-1}$ where b_i is any Artin generator and $w \in B_n$ is any braid such that the associated permutation $\mathbf{p}(w)$ satisfies

$$\mathbf{p}(w)(i) = \sigma^{-1}(a), \mathbf{p}(w)(i + 1) = \sigma^{-1}(b).$$

Remark 1. A detailed algorithm for constructing cloaking elements is not provided. In particular, no algorithm to generate w is given. Hence, in our implementation, we generate it in the following way:

Algorithm 1. Generating w

repeat

 Pick a random integer l such that $30 \leq l \leq 80$.

 Pick a random freely-reduced word w in the generators $\{b_1, \dots, b_7\}$ of length l .

until $\mathbf{p}(w)$ satisfies the condition in Proposition 2.

We stress that our attack works independently of the way cloaking elements β are generated.

3.5 Signing

Signing. To sign a message m , the signer does as follows:

1. Compute $\mathcal{E}(m)$ as in Sect. 3.3;
2. Generate cloaking elements v for $(\text{Id}_N, \text{Id}_{S_N})$ and v_1, v_2 for $(\text{Id}_N, \text{Id}_{S_N}) \star \text{SK}$;
3. Compute $s = \mathcal{R}(v_2 \cdot \text{SK}^{-1} \cdot v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1)$;
4. Output (m, s) , the final signature for the message.

The cloaking elements are necessary to preclude the possibility of recovering for SK by solving the CSP (any solution to the CSP is sufficient), since both s and $\mathcal{E}(m)$ are publicly available (the latter after some computation).

Proposition 3. For any message m , its signature

$$s = \mathcal{R}(v_2 \cdot \text{SK}^{-1} \cdot v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1)$$

is a pure braid.

Proof. Recall that $\mathcal{E}(m)$ is a product of pure braids and is, therefore, a pure braid. Moreover, by Lemma 3, v, v_1 and v_2 are pure braids. Hence, the induced permutation $\mathbf{p}(s)$ of s is:

$$\begin{aligned} \mathbf{p}(s) &= \mathbf{p}(v_2 \cdot \text{SK}^{-1} \cdot v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1) \\ &= \text{Id}_{S_N} \cdot \mathbf{p}(\text{SK}^{-1}) \cdot \text{Id}_{S_N} \cdot \text{Id}_{S_N} \cdot \mathbf{p}(\text{SK}) \cdot \text{Id}_{S_N} \\ &= \text{Id}_{S_N}. \end{aligned}$$

□

3.6 Verifying

Verifying. To verify a signature (m, s) , the verifier does as follows:

1. Compute $\mathcal{E}(m)$;
2. Compute $\text{Pub}(\mathcal{E}(m)) = (\text{Id}_N, \text{Id}_{S_N}) \star \mathcal{E}(m)$.

The signature is then valid if and only if the verification equation

$$\mathbf{m}(\text{PK} \star s) = \mathbf{m}\left(\text{Pub}(\mathcal{E}(m))\right) \cdot \mathbf{m}(\text{PK})$$

holds.

Lemma 4. *A message-signature pair (m, s) , generated as in Sect. 3.5 satisfies the verification process.*

Proof. We have that

$$\begin{aligned} \text{PK} \star s &= (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK} \star s \\ &= (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK} \star (v_2 \cdot \text{SK}^{-1} \cdot v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1) \\ &\stackrel{(1)}{=} (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK} \star (\text{SK}^{-1} \cdot v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1) \\ &= (\text{Id}_N, \text{Id}_{S_N}) \star (v \cdot \mathcal{E}(m) \cdot \text{SK} \cdot v_1) \\ &\stackrel{(2)}{=} (\text{Id}_N, \text{Id}_{S_N}) \star (\mathcal{E}(m) \cdot \text{SK} \cdot v_1), \end{aligned}$$

where

- (1) holds since v_2 cloaks $\text{PK} = (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK}$;
- (2) holds since v cloaks $(\text{Id}_N, \text{Id}_{S_N})$.

Looking at the matrix parts of the above equality, we see that

$$\begin{aligned} \mathbf{m}(\text{PK} \star s) &= \mathbf{m}\left((\text{Id}_N, \text{Id}_{S_N}) \star (\mathcal{E}(m) \cdot \text{SK} \cdot v_1)\right) \\ &\stackrel{(3)}{=} \mathbf{m}\left((\text{Id}_N, \text{Id}_{S_N}) \star \mathcal{E}(m)\right) \cdot \mathbf{m}\left((\text{Id}_N, \text{Id}_{S_N}) \star (\text{SK} \cdot v_1)\right) \\ &\stackrel{(4)}{=} \mathbf{m}\left((\text{Id}_N, \text{Id}_{S_N}) \star \mathcal{E}(m)\right) \cdot \mathbf{m}\left((\text{Id}_N, \text{Id}_{S_N}) \star \text{SK}\right) \\ &= \mathbf{m}\left(\text{Pub}(\mathcal{E}(m))\right) \cdot \mathbf{m}(\text{PK}), \end{aligned}$$

where

- (3) holds since $\mathcal{E}(m)$ is a pure braid
- (4) holds since v_1 cloaks $\text{PK} = (\text{Id}_N, \text{Id}_{S_N}) \star \text{SK}$.

□

4 Practical Cryptanalysis of WalnutDSA

In this section we present a universal forgery attack on WalnutDSA. The structure of the section is as follows: in Sect. 4.1, we show that an attacker can produce a signature for a new message if they are able to solve a factorization problem over $GL_N(\mathbb{F}_q)$. In Sect. 4.2, we present an algorithm solving this factorization problem by exploiting the subgroup structure of $GL_N(\mathbb{F}_q)$, and in Sect. 4.3, we describe a meet-in-the-middle approach which reduces the complexity of this attack. In Sect. 4.4, we analyze the complexity of our attack and provide some experimental results. Finally, we discuss further improvements to our attack in Sect. 4.5.

4.1 Reduction to the Factorization Problem

Let I be a finite indexing set. For each $i \in I$, let m_i be a message and s_i be its signature generated as in Sect. 3.5. Define the set $\mathcal{M} = \{(m_i, s_i) : i \in I\}$. Recall that for a braid β , we define

$$\text{Pub}(\beta) = (\text{Id}_N, \text{Id}_{S_N}) \star \beta,$$

where Id_N is the identity matrix and Id_{S_N} is the identity permutation.

Proposition 4. *Let m be an arbitrary message. Let $g_i = \mathbf{m}(\text{Pub}(\mathcal{E}(m_i)))$ for each $i \in I$ and let $h = \mathbf{m}(\text{Pub}(\mathcal{E}(m)))$. Suppose*

$$h = \prod_{j=1}^L g_{i_j}^{\epsilon_{i_j}} \quad \text{where } i_j \in I, \epsilon_{i_j} \in \{\pm 1\} \text{ and } L \in \mathbb{N}.$$

Then $s = \prod_{j=1}^L s_{i_j}^{\epsilon_{i_j}}$, the concatenation of the corresponding braids $s_{i_j}^{\epsilon_{i_j}}$, is a valid signature for m .

Proof. Each pair in \mathcal{M} satisfies the verification equation:

$$\mathbf{m}(\text{PK} \star s_i) = \mathbf{m}(\text{Pub}(\mathcal{E}(m_i))) \cdot \mathbf{m}(\text{PK}).$$

Writing σ as $\mathbf{p}(\text{PK})$ and M as $\mathbf{m}(\text{PK})$, the above equation is equivalent to

$$(\sigma \mathbf{m}(s_i)) \downarrow_{\tau} = M^{-1} \cdot g_i \cdot M, \tag{2}$$

where $\tau = (\tau_1, \dots, \tau_N)$ is the sequence of T-values. Also, by Proposition 3, each $s_i^{\epsilon_i}$ is a pure braid, and so Lemma 2 applies. Hence, by taking the inverse of (2), we obtain

$$(\sigma \mathbf{m}(s_i^{-1})) \downarrow_{\tau} = \left((\sigma \mathbf{m}(s_i)) \downarrow_{\tau} \right)^{-1} = M^{-1} \cdot g_i^{-1} \cdot M.$$

and so

$$(\sigma \mathbf{m}(s_i^{\epsilon_i})) \downarrow_{\tau} = M^{-1} \cdot g_i^{\epsilon_i} \cdot M \tag{3}$$

By Lemma 1,

$$m(s) = m\left(\prod_{j=1}^L s_{i_j}^{\epsilon_{i_j}}\right) = \prod_{j=1}^L m(s_{i_j}^{\epsilon_{i_j}}),$$

and hence,

$$\begin{aligned} (\sigma m(s)) \downarrow_\tau &= \left(\sigma\left(\prod_{j=1}^L m(s_{i_j}^{\epsilon_{i_j}})\right)\right) \downarrow_\tau = \left(\prod_{j=1}^L \sigma m(s_{i_j}^{\epsilon_{i_j}})\right) \downarrow_\tau \\ &= \prod_{j=1}^L (\sigma m(s_{i_j}^{\epsilon_{i_j}})) \downarrow_\tau = \prod_{j=1}^L (M^{-1} \cdot g_{i_j}^{\epsilon_{i_j}} \cdot M) \\ &= M^{-1} \cdot \left(\prod_{j=1}^L g_{i_j}^{\epsilon_{i_j}}\right) \cdot M = M^{-1} \cdot h \cdot M. \end{aligned}$$

Therefore s is a valid signature for m , as the above equation is equivalent to

$$m(\text{PK} \star s) = m(\text{Pub}(\mathcal{E}(m))) \cdot m(\text{PK}),$$

the verification equation for (m, s) . □

4.2 Solution to the Factorization Problem

Let $\Gamma = \{g_i \mid i \in I\}$. Following our discussion in Sect. 4.1, we want to express h as a short word over Γ . We first define the following chain of subgroups:

Definition 2. For $k \in \{1, \dots, 2N - 2\}$, let

$$G_k = \{M \in \text{GL}_N(\mathbb{F}_q) \mid M_{N,N} = 1 \text{ and } M_{i,j} = 0 \text{ for } (i,j) \in A_{k_1} \cup A_{k_2}\},$$

where

$$\begin{aligned} A_{k_1} &= \left\{ (i, j) \mid \left(N - \left\lceil \frac{k}{2} \right\rceil\right) \leq i \leq N, i \neq j \right\}, \\ A_{k_2} &= \left\{ (i, j) \mid \left(N - \left\lfloor \frac{k}{2} \right\rfloor\right) \leq j \leq N, i \neq j \right\}. \end{aligned}$$

That is, for even k ,

$$G_k = \left\{ \left(\left(\begin{array}{c|ccc} A & 0 & \cdots & 0 \\ \hline 0 & \lambda_{\frac{k}{2}-1} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & & \lambda_1 & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array} \right) \middle| A \in \text{Mat}_{N-\frac{k}{2}, N-\frac{k}{2}}(\mathbb{F}_q) \right\} \cap \text{GL}_N(\mathbb{F}_q),$$

and for odd k ,

$$G_k = \left\{ \left(\begin{array}{c|ccc} A & * & 0 & \cdots & 0 \\ \hline 0 & \lambda_{\frac{k-1}{2}} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & & \lambda_1 & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array} \right) \mid A \in \text{Mat}_{N-\frac{k+1}{2}, N-\frac{k+1}{2}}(\mathbb{F}_q) \right\} \cap \text{GL}_N(\mathbb{F}_q),$$

where $*$ is a column of length $N - \frac{k+1}{2}$ and $\lambda_i \in \mathbb{F}_q^\times$ for $i \in \{1, \dots, \lfloor \frac{k-1}{2} \rfloor\}$.

Remark 2. Checking whether $g \in \text{GL}_N(\mathbb{F}_q)$ is in G_k for any k is straightforward given the characteristic shape of the matrices in each group.

Lemma 5. *For any braid $\beta \in B_N$, $\mathfrak{m}(\text{Pub}(\beta)) \in G_1$.*

Proof. Let G'_1 be the subgroup of $\text{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \dots, t_N^{\pm 1}])$ consisting of matrices with their last row all zeroes except for the last entry, which is equal to 1. For each $i \in \{1, \dots, N - 1\}$, $\mathfrak{m}(b_i) \in G'_1$. Therefore, $\mathfrak{m}(\beta) \in G'_1$ and hence, $\mathfrak{m}(\text{Pub}(\beta)) = \mathfrak{m}(\beta) \downarrow_\tau \in G_1$. \square

We also make use of the following assumption:

Assumption 1. *For any k , a small set of random elements of G_k generates G_k with high probability.*

This assumption is supported by [43] and our experiments.

Our algorithm aims to solve an instance of a factorization problem over G_1 . This is done in $2N - 2$ stages. The first $2N - 3$ stages are inductive: in stage k , we reduce the problem in G_k to an instance of the problem over the next subgroup G_{k+1} . At the end of stage $2N - 3$, we have reduced the original problem to factorization problem over G_{2N-2} , the diagonal subgroup. In the last stage of the algorithm, we reduce the factorization problem in G_{2N-2} to an easy case of the discrete logarithm problem over \mathbb{F}_q and a system of linear equations.

Let $\gamma_1 := |\mathcal{M}|$, let $\Gamma_1 := \Gamma = \{g_i^{(1)} : 1 \leq i \leq \gamma_1\}$, and let $h_1 := h$. Further, for $2 \leq k \leq 2N - 2$, let γ_k be a positive integer and $L_k := \left\lceil \log_{\gamma_k} \frac{\gamma_{k+1} |G_k|}{|G_{k+1}|} \right\rceil$. We will aim to produce γ_{k+1} elements of G_{k+1} in stage k , and we hope that these elements will generate G_{k+1} , which we will need to reduce the factorization problem into the next subgroup. The integer L_k captures some information about the number of elements we need to consider from G_k before we find γ_{k+1} elements of G_{k+1} : in our algorithm, the elements from G_k that we will consider will be words of some fixed length \mathcal{L}_k over some generating set of size γ_k ; by considering the relative sizes of G_k and G_{k+1} , it then follows that \mathcal{L}_k should be L_k .

Inductive Stages. In stage k (for $k \in \{1, \dots, 2N - 3\}$), we will find a set $\Gamma_{k+1} := \{g_i^{(k+1)} : 1 \leq i \leq \gamma_{k+1}\} \subset G_{k+1}$ and an element $h_{k+1} \in G_{k+1}$, where $g_i^{(k+1)}$ are words over Γ_k and h_{k+1} is a product of h_k with a word over Γ_k .

Algorithm 2. From Γ_k and h_k , we find Γ_{k+1} and h_{k+1}

repeat

 Generate products of the form:

$$p = \prod_{j=1}^{L_k} g_{i_j}^{(k)} \quad \text{where } 1 \leq i_j \leq \gamma_k$$

if $p \in G_{k+1}$ **then**

 Add p to Γ_{k+1}

else if h_{k+1} has not yet been defined **then**

if $p \in h_k G_{k+1}$ **then**

 Define $h_{k+1} = p^{-1} \cdot h_k$

end if

end if

until $|\Gamma_{k+1}| = \gamma_{k+1}$ and h_{k+1} is defined

Following Assumption 1, we expect that for large enough γ_k , Γ_k will be a generating set for G_k . We therefore expect to be able to find γ_{k+1} elements in $G_{k+1} \subset G_k$ given enough iterations of the loop. Moreover, $h_k G_{k+1} \subset G_k$, and so we expect to be able to find h_{k+1} as well.

Remark 3. We see from the above algorithm that for all $k \in \{1, \dots, 2N - 3\}$, we can write h_{k+1} as

$$h_{k+1} = \prod_j (g_{i_j}^{(k)})^{-1} \cdot h_k \quad \text{for } 1 \leq i_j \leq \gamma_k.$$

Moreover, we can write any element in Γ_{k+1} as a product of elements in Γ_k . Hence, we can recursively write h_{k+1} as a product of a word over $\Gamma_1 = \Gamma$ with $h_1 = h$, i.e. we can express h_{k+1} as

$$h_{k+1} = \left(\prod_j g_{i_j}^{\epsilon_{i_j}} \right) \cdot h \quad \text{for } 1 \leq i_j \leq \gamma_1. \tag{4}$$

In particular, we can express each element $g_i^{(2N-2)} \in \Gamma_{2N-2}$ as a word over Γ

$$g_i^{(2N-2)} = \left(\prod_j g_{i_j}^{\epsilon_{i_j}} \right) \quad \text{for } 1 \leq i_j \leq \gamma_1. \tag{5}$$

Final Stage. At the end of stage $2N - 3$, we will have a set

$$\Gamma_{2N-2} = \left\{ g_i^{(2N-2)} : 1 \leq i \leq \gamma_{2N-2} \right\} \subset G_{2N-2}$$

and an element $h_{2N-2} \in G_{2N-2}$. Note that G_{2N-2} is the subgroup of diagonal matrices, and so all of the above elements are diagonal matrices as well.

We want to express h_{2N-2} as a word over Γ_{2N-2} . Since G_{2N-2} is abelian, this is equivalent to finding exponents $v_1, \dots, v_{\gamma_{2N-2}} \in \mathbb{Z}$ such that

$$h_{2N-2} = \prod_{i=1}^{\gamma_{2N-2}} \left(g_i^{(2N-2)} \right)^{v_i}. \tag{6}$$

Equally, (4) and (5) then allow us to rewrite the above equation as

$$h = \prod_j g_{i_j}^{\epsilon_{i_j}},$$

an expression for h as a word over Γ , given that we can find the exponents v_i . We describe how to find these exponents next.

Note that all the matrices on both sides of (6) are diagonal matrices. For each $i \in \{0, \dots, \gamma_{2N-2}\}$, let $c_i = (\lambda_{i_1}, \dots, \lambda_{i_{N-1}}, 1)$ be the sequence of diagonal entries in $g_i^{(2N-2)}$, and let $c := (\mu_1, \dots, \mu_{N-1}, 1)$ be the diagonal entries in h_{2N-2} . Further, let δ be a generator of \mathbb{F}_q^\times . By solving the discrete logarithm problem over \mathbb{F}_q^\times (which is straightforward for small q), for each $i \in \{1, \dots, \gamma_{2N-2}\}$, and each $j \in \{1, \dots, N - 1\}$, we can find e_{i_j} and u_j such that:

$$\begin{aligned} \delta^{e_{i_j}} &= \lambda_{i_j}, \\ \delta^{u_j} &= \mu_j, \end{aligned}$$

i.e., we are able to write all non-zero entries of the matrices in (6) as powers of δ . Finding the exponents v_i is then reduced to solving a system of linear equations over \mathbb{Z}_{q-1} . More explicitly, for each $i \in \{1, \dots, \gamma_{2N-2}\}$, define $c'_i = (e_{i_1}, \dots, e_{i_{N-1}}, 1)$. Also, let $c' = (u_1, \dots, u_{N-1}, 1)$ and let $D = (c'_1, \dots, c'_{\gamma_{2N-2}})$, i.e., the matrix with i^{th} column equal to c'_i . So (6) above is equivalent to the system of linear equations

$$D \cdot v = c' \tag{7}$$

which can be solved with standard linear algebra techniques.

4.3 Meet-in-the-Middle Approach

We can improve the recursive step of our attack as follows: instead of computing products of length L_k until we hit an element of G_{k+1} , we compute pairs of products each of length $\lfloor \frac{L_k}{2} \rfloor$ and then check for pairs which lie in the same coset of G_{k+1} . This meet-in-the-middle approach will lead to a square root improvement on the complexity. In order to use this approach, we need an efficient method to

check whether two elements are in the same coset of G_{k+1} . The following lemma provides such a method.

Lemma 6. *Let G_k for $k \in \{1, \dots, 2N - 2\}$ be the subgroups in Definition 2, and let $p, p' \in G_k$. Then*

- For odd k , $p' \in pG_{k+1}$ if and only if the $(N - \frac{k+1}{2} + 1)^{th}$ columns of p and p' are multiples of each other.
- For even k , $p' \in G_{k+1}p'$ if and only if the $(N - \frac{k}{2})^{th}$ rows of p and p' are multiples of each other.

Proof. Let k be odd, let h be any matrix in G_{k+1} , and let $r = N - \frac{k+1}{2} + 1$. Note that the r^{th} column of h is zero except for the entry $h_{r,r} \in \mathbb{F}_q^\times$. Finally, let $p, p' \in G_k$.

Assume that $p' \in pG_{k+1}$, and so there exists $g \in G_{k+1}$ for which $p' = pg$. Let $p_{i,j}$ be the $(i, j)^{th}$ entry of p and let $\lambda_r := g_{r,r}$. Then the entries of the r^{th} column of p' are:

$$p'_{i,r} = \sum_{j=1}^N p_{i,j}g_{j,r} = p_{i,r} \cdot \lambda_r \quad \text{for } 1 \leq i \leq N$$

and hence the r^{th} columns of p and p' are multiples of each other.

Conversely, let c_r be the r^{th} column of p and c'_r be the r^{th} column of p' , and assume $c'_r = \lambda \cdot c_r$ for some $\lambda \in \mathbb{F}_q^\times$. Let $\pi = p^{-1} \cdot p'$. Then the entries of the r^{th} column of π are

$$\begin{aligned} \pi_{i,r} &= \sum_{j=1}^N (p^{-1})_{i,j} \cdot p'_{j,r} = \sum_{j=1}^N (p^{-1})_{i,j} \cdot \lambda p_{j,r} \\ &= \lambda \sum_{j=1}^N (p^{-1})_{i,j} \cdot p_{j,r} = \lambda \cdot (\text{Id}_N)_{i,r} \\ &= \lambda \cdot \delta_{ir} \end{aligned}$$

where δ_{ir} is the Kronecker delta. This implies that the r^{th} column of π is zero everywhere except at the $(r, r)^{th}$ entry. Since $\pi \in G_k$, this implies $\pi \in G_{k+1}$ and hence $p' \in pG_{k+1}$.

The case for even k is similar. □

Using the above lemma, we are able to construct an improved version of Algorithm 2:

Algorithm 3. From Γ_k and h_k , we find Γ_{k+1} and h_{k+1}

Generate all products of the form:

$$p_1 = \prod_{j=1}^{L_k/2} g_{i_j}^{(k)} \text{ where } 1 \leq i_j \leq \gamma_k$$

repeat

 Pick a product p_2 of the form above

Case 1: k is odd

if for some p_1 , we have $p_2 \in p_1^{-1}G_{k+1}$ **then**

 Add $p = p_1p_2$ to Γ_{k+1}

else if h_{k+1} has not been defined yet **then**

if for some p_1 , we have $p_2 \in p_1^{-1}h_kG_{k+1}$ **then**

 Define $h_{k+1} = p_2^{-1}p_1^{-1} \cdot h_k = p^{-1} \cdot h_k$

end if

end if

Case 2: k is even

if for some p_1 , we have $p_2 \in G_{k+1}p_2^{-1}$ **then**

 Add $p = p_1p_2$ to Γ_{k+1}

else if h_{k+1} has not been defined yet **then**

if for some p_1 , we have $h_k^{-1}p_1 \in G_{k+1}p_2^{-1}$ **then**

 Define $h_{k+1} = p_2^{-1}p_1^{-1} \cdot h_k = p^{-1} \cdot h_k$

end if

end if

until $|\Gamma_{k+1}| = \gamma_{k+1}$ and h_{k+1} is defined

4.4 Complexity Analysis and Experiments

Time Complexity. We observe that the complexity of the algorithm is dominated by the complexity of finding each Γ_{k+1} : the last step involves solving a discrete logarithm problem over a small field and a small linear system modulo $q - 1$. Moreover, the cost of finding an element h_{k+1} is essentially the same as the cost of finding one element of Γ_{k+1} .

Lemma 7. *The size of G_k is as follows:*

- For k even, $|G_k| = (q - 1)^{\binom{k}{2}-1} \cdot |\text{GL}_{N-\frac{k}{2}}(\mathbb{F}_q)|$.
- For k odd, $|G_k| = (q - 1)^{\lfloor \frac{k}{2} \rfloor} \cdot q^{N-\lfloor \frac{k}{2} \rfloor-1} \cdot |\text{GL}_{N-\lfloor \frac{k}{2} \rfloor-1}(\mathbb{F}_q)|$.

Proof. For k even, the block diagonal structure of G_k consists of an invertible matrix of size $N - \frac{k}{2}$ and $\frac{k}{2}$ entries on the diagonal. The bottommost such entry is 1, and the other diagonal entries can be any of the nonzero elements in \mathbb{F}_q , and so we obtain the formula above. For k odd, the block diagonal structure of G_k consists of an invertible matrix of size $N - \lfloor \frac{k}{2} \rfloor$ with a zero bottom row except for the last entry, and $\lfloor \frac{k}{2} \rfloor$ other entries on the diagonal. Note that $\lfloor \frac{k}{2} \rfloor - 1$ of the diagonal entries can be any nonzero element in \mathbb{F}_q while the bottommost

entry is 1. The invertible matrix of size $N - \lfloor \frac{k}{2} \rfloor$ consists of any element in $\text{GL}_{N - \lfloor \frac{k}{2} \rfloor - 1}(\mathbb{F}_q)$ on the upper diagonal, any nonzero entry from \mathbb{F}_q for the bottom right entry, and a value in \mathbb{F}_q for the rest of the entries in the last column. From this we obtain the formula above. \square

Lemma 8. $\frac{|G_k|}{|G_{k+1}|} \approx q^{N-1 - \lfloor \frac{k}{2} \rfloor}$

Proof. This follows immediately from the previous lemma. \square

If we pick a random element of G_k , the probability that it will also be in G_{k+1} is therefore approximately $1/q^{N-1 - \lfloor \frac{k}{2} \rfloor}$. In our algorithm, we make the assumption that random products of elements in Γ_k produces random elements in G_{k+1} , and so we expect that we will be able to obtain one element of Γ_{k+1} after considering $q^{N-1 - \lfloor \frac{k}{2} \rfloor}$ random products. By using the meet-in-the-middle approach described earlier, we reduce the expected number of products we need to consider by $q^{(N-1 - \lfloor \frac{k}{2} \rfloor)/2}$. Since we need to generate $|\Gamma_{k+1} \cup \{h_{k+1}\}| = \gamma_{k+1}$ new elements, the expected number of products we need to consider is bounded by $\gamma_{k+1} \cdot q^{(N-1 - \lfloor \frac{k}{2} \rfloor)/2}$. The total number of products our algorithm needs to consider is therefore

$$\sum_{k=1}^{2N-3} \gamma_{k+1} \cdot q^{(N-1 - \lfloor \frac{k}{2} \rfloor)/2}.$$

If we further assume that $\gamma_k = \gamma$ is constant, the above simplifies to

$$\gamma \cdot \sum_{k=1}^{2N-3} q^{(N-1 - \lfloor \frac{k}{2} \rfloor)/2} = 2 \cdot \gamma \cdot \sum_{l=0}^{N-2} q^{\frac{N-1-l}{2}} \approx 2 \cdot \gamma \cdot q^{\frac{N-1}{2}}.$$

Thus, the complexity of the attack is exponential in N and $\log q$.

Memory Complexity. The final stage of the algorithm requires a negligible amount of memory. For the inductive stages, in stage k of the algorithm, we need to store up to $q^{\frac{1}{2}(N-1 - \lfloor \frac{k}{2} \rfloor)}$ square matrices of size $N \times N$, each entry being in \mathbb{F}_q , so we will need $\log_2(q) \cdot N^2 q^{\frac{1}{2}(N-1 - \lfloor \frac{k}{2} \rfloor)}$ bits of memory for each stage. However, we do not need to keep the matrices from stage k when proceeding to stage $k + 1$ (except to store the relatively small number of matrices of Γ_{k+1} and h_{k+1}), and so the total amount of memory required for the entire algorithm is the maximum amount of memory required by each stage, which is $\log_2(q) \cdot N^2 q^{\frac{N-1}{2}}$. Memory costs can be removed entirely using standard cycle-finding and distinguished point techniques [44, 45].

Length Complexity. We now analyze the length of the forged signature that we obtain.

Note that the length of any element in Γ_{k+1} , as a word over elements of Γ_k , is given by L_k . Also, our algorithm expresses h_{k+1} as the product of h_k with L_K

elements of Γ_k . Unfolding this recurrence, we see that h_{2N-2} is the product of h with α elements of Γ , where

$$\begin{aligned} \alpha &= \sum_{k=1}^{2N-3} \prod_{j=1}^k L_j = \sum_{k=1}^{2N-3} \prod_{j=1}^k \log_{\gamma_j} \left(\frac{|G_j|}{|G_{j+1}|} \gamma_{j+1} \right) \\ &= \sum_{k=1}^{2N-3} \prod_{j=1}^k \log_{\gamma_j} \left(q^{N-1-\lfloor \frac{j}{2} \rfloor} \cdot \gamma_{j+1} \right) \\ &\approx \prod_{j=1}^{2N-3} \log_{\gamma_j} \left(q^{N-1-\lfloor \frac{j}{2} \rfloor} \cdot \gamma_{j+1} \right), \end{aligned}$$

since the last summand dominates the sum. Similarly, we see that each $g_i^{(2N-2)}$ is a product of $\approx \alpha$ elements of Γ .

If we further assume that $\gamma_k = \gamma$ is constant, the above formula simplifies to

$$\begin{aligned} \alpha &\approx \prod_{j=1}^{2N-3} \left(1 + \left(N - 1 - \lfloor \frac{j}{2} \rfloor \right) \log_{\gamma} q \right) \\ &\approx (\log_{\gamma} q)^{2N-3} ((N-1)!)((N-2)!). \end{aligned}$$

In the final step of the algorithm, we find a relation (6)

$$h_{2N-2} = \prod_{i=1}^{\gamma_{2N-2}} \left(g_i^{(2N-2)} \right)^{v_i}.$$

Since the v_i come from the solution to a system of linear equations over \mathbb{Z}_{q-1} , we know that $v_i < q-1$. Also, since the space we are working over in our system of linear Eq. 7 has dimension $N-1$, it follows that we need at most $N-1$ terms in the product above. Putting this all together, we see that h is then the product of $(1+(N-1)(q-1))\alpha \approx Nq\alpha$ elements of Γ , and so our forged signature is of length $\approx lNq\alpha$, where l is the length of the WalnutDSA signatures in \mathcal{M} .

Experimental Results. We have implemented our factorization algorithm in Magma [46] and tested it experimentally (the code is available from Christophe Petit’s webpage). The only parameters of our algorithm are the values of L_k , which we can control via γ_k . Note that increasing γ_k decreases the length of our forged signature but increases the running time of our algorithm. In our experiments, we first assumed that we are able to obtain ten legitimate message-signature pairs. We then chose γ_k such that L_k is large enough for us to find the relations for all h_k . This allowed us to obtain a signature of length 2^{35} times the length of a legitimate signature in approximately two minutes. To reduce the length of the forged signature, we increased γ_k such that $\gamma_k \approx 200000$ for $k > 3$. This allowed us to obtain signatures of length 2^{25} times the length of a legitimate signature in five minutes.

4.5 Practical Improvements

In this section we present two improvements on our attack.

Shorter Subgroup Chain. The subgroup chain we used above was chosen to have small subgroup indices $[G_k : G_{k+1}]$ in order to minimize computation time at each step. However, the first few stages of the algorithm contribute to the majority of the running time, whereas all stages contribute significantly to the total length of the signatures we produce.

To reduce signature lengths without affecting the computation time significantly, one can replace the above subgroup chain by another chain. An example of such a chain could have the same first five subgroups (at a cost of roughly $q^{3.5}$, q^3 , q^3 , $q^{2.5}$ and $q^{2.5}$ respectively), but then instead of considering a subgroup where the lower diagonal entries in the last four rows are zeroes (at a cost q^2), consider a subgroup where the lower diagonal entries in the last five rows are zeroes (at a cost of $q^{3.5}$), then a subgroup where the upper diagonal entries in the last five rows are also zeroes (at a cost of $q^{3.5}$), and finally considering the diagonal subgroup (at a cost of q^3). In that case, the factorization length can be approximated by

$$Nq \prod_k c_k \log_{\gamma_k} q$$

where $(c_1, c_2, \dots, c_8) = (7, 6, 6, 5, 5, 7, 7, 6)$, which for $\gamma_k = 256$ gives a signature size approximately 2^{11} times that of a normal signature size, while retaining the time complexity of roughly $q^{3.5}$.

Dealing with Non-Generating Sets. We have not been able to prove that the elements we construct in our recursive step are indeed generators for the next subgroup. We expect that this is the case with a high probability on the initial matrix choices when choosing product lengths as above, and this was verified for all recursive steps in our experimental tests.

The diagonal matrices generated for the last stage, however, may not generate the whole diagonal group when the number of generators constructed at each step is very small. We observed this experimentally when using $\gamma_k = 2$ in all but the last inductive stage, and can explain it intuitively as follows. Let $\Gamma_k := \{A^{(k)}, B^{(k)}\}$. At each stage, the diagonal entries in the diagonal part (in block diagonal form) of $A^{(k)}$ and $B^{(k)}$ can be approximated as random elements in \mathbb{F}_q^\times . Consider any pair of indices $((i_1, i_1), (i_2, i_2))$ in the diagonal part of the matrix, and consider the 2-dimensional vectors $(A_{i_1, i_1}^{(k)}, A_{i_2, i_2}^{(k)})$ and $(B_{i_1, i_1}^{(k)}, B_{i_2, i_2}^{(k)})$. It is a necessary condition for these two matrices to generate the whole subgroup, that there is no linear dependence between the two vectors obtained by taking entrywise logarithms of the above vectors. For a fixed pair of indices (i_1, i_1) and (i_2, i_2) , this happens with probability $\frac{q-2}{q-1}$. In the later inductive stages, the diagonal part of the matrices are larger, and hence the probability that all pairs of the logarithm vectors are linearly independent decreases. Moreover, any linear

dependence occurring in one stage will be preserved in subsequent stages. It is therefore intuitively plausible that Γ_{2N-2} may not generate G_{2N-2} when γ_k is very small. We leave a more complete analysis of this to further work.

In our experiments, it was easy to choose γ_k large enough such that all stages would produce a sufficient number of generators for the following subgroup, including that of diagonal subgroup G_{2N-2} . We note also that in the event that Γ_{2N-2} does not generate G_{2N-2} , one can simply set $h_1 = \text{Id}_n$ and relaunch the whole factorization algorithm: this will produce a new set of diagonal matrices Γ'_{2N-2} that, together with Γ_{2N-2} , is likely to generate the G_{2N-2} . This therefore allows our attack to succeed with high probability even when we only have access to two WalnutDSA message-signature pairs.

5 Discussion and Further Work

Due to its algebraic structure, WalnutDSA is inherently vulnerable to malleability attacks. The use of a cryptographic hash function in the message encoding process is intended to remove this inherent malleability, in the same way as Full Domain Hash removes the inherent malleability in the RSA signature algorithm. Our attack, however, goes around this protection mechanism by reducing the cryptanalysis of WalnutDSA to an instance of a factorization problem in the group $\text{GL}_N(\mathbb{F}_q)$.

We briefly discuss two countermeasures against this attack, namely increasing the parameter sizes and checking the signature lengths.

5.1 Increasing the Parameters

In order to defeat our attack, one can choose to increase the parameters of WalnutDSA such that the complexity of our attack is increased to $\sim 2^{100}$. As shown in Sect. 4.4, the complexity of our attack can be estimated by $\gamma \cdot q^{\frac{N-1}{2}}$. One can therefore choose to increase the value of q and N such that $q^{\frac{N-1}{2}} \approx 2^{100}$, by choosing $q = 2^{16}$ and $N = 14$ for example.

5.2 Checking Signature Length

Recall that our forged signature s is obtained from concatenating existing signatures. The length of s depends primarily on the length of the products L_k considered in Algorithm 3. As discussed in Sects. 4.4 and 4.5, larger values for $\gamma_k = |\Gamma_k|$ and a different choice of subgroup chain can achieve shorter forged signature lengths at the cost of higher time and memory complexity. Our best attempt produced a forged signature 2^{25} times larger than the original WalnutDSA signatures.

Observe that the length of a legitimate signature (one produced according to WalnutDSA) depends on the length of sk, $\mathcal{E}(m)$, and the cloaking elements. Even though these lengths are not fixed, we expect them to be within certain bounds, which will depend on the implementation of the protocol. However, in

principle, the length of s should greatly exceed these bounds. Therefore, we suggest that the length of both cloaking elements and private keys be bounded above, so that the length of a WalnutDSA signature is always less than some constant \mathcal{L} . Any signature of length greater than \mathcal{L} should then be rejected.

5.3 Limitations of the Countermeasures

We do not know, however, whether s could be shortened to fit the new imposed bounds. Methods such as Dehornoy's handle reduction [14] could potentially reduce the length of our forged signatures sufficiently in a non-negligible fraction of instances.

We stress that more efficient algorithms for solving the factorization problem in $\text{GL}_N(\mathbb{F}_q)$ may also exist. One may expect factorizations as small as $\log_{|\mathcal{M}|} |\text{GL}_N(\mathbb{F}_q)| = \log_{|\mathcal{M}|} q^{N^2 - N - 1}$ to exist, where \mathcal{M} is the set of WalnutDSA message-signature pairs one has access to. If an efficient algorithm to compute short factorizations exists, the increase in parameters q and N needed to achieve a sufficient level of security would then make WalnutDSA unsuitable for embedded devices. Moreover, with $|\mathcal{M}|$ large enough, the forged signatures will only be a small constant factor larger than legitimate signatures, and hence determining a suitable bound \mathcal{L} to apply our second countermeasure may be challenging.

Finally, we observe that our work has not considered the hard problems underlying the WalnutDSA protocol, that of reversing E-Multiplication and the cloaked conjugacy search problem. The study of these problems, along with the effectiveness of the above countermeasures, will be of interest for further work.

6 Conclusion

In this paper we provided a practical cryptanalysis of WalnutDSA. Given a couple of random valid message-signature pairs, our attack is able to produce new signatures on arbitrary messages in approximately two minutes. We also discuss countermeasures to our attack, including a simple modification of the verification algorithm.

Acknowledgements. Giacomo Micheli was supported by the Swiss National Science Foundation grant number 171248. Daniel Hart and Guillermo Pascual Perez were both kindly supported by EPSRC Vacation Bursaries.

A The Garside Normal Form

We follow the presentation in [5]. Define a *positive braid*, which is an element of B_N that can be written as a product of positive powers of the generators. Let B_N^+ denote the set of positive braids. One example of a positive braid is the *fundamental braid* $\Delta_N \in B_N$:

$$\Delta_N = (b_1 \cdots b_{N-1})(b_1 \cdots b_{N-2}) \cdots (b_1 b_2)(b_1).$$

Geometrically, Δ_N is the braid in which any two strands cross positively *exactly* once.

We now define a partial order on B_N : for $A, B \in B_N$, write $A \preceq B$ if there exists $C \in B_N^+$ such that $B = AC$. With this definition, we say that $P \in B_N$ is a *permutation braid* if $\varepsilon \preceq P \preceq \Delta_N$, where ε is the empty braid. Geometrically, a permutation braid is a braid in which any two strands cross positively *at most* once.

Let P be a permutation braid. Then the *starting set* of P is

$$S(P) = \{i \mid P = b_i P' \text{ for some } P' \in B_N^+\}$$

and the *finishing set* of P is

$$F(P) = \{j \mid P = P' b_j \text{ for some } P' \in B_N^+\}.$$

Furthermore, if A is any positive braid, its *left-weighted decomposition* into permutation braids is

$$A = P_1 \cdots P_m$$

where $S(P_{i+1}) \subset F(P_i)$ for any i .

References

1. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 1994 Proceedings of 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE (1994)
2. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Deep Space Netw. Prog. Rep. **44**, 114–116 (1978)
3. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_5
4. Ding, J., Yang, B.Y.: Multivariate public key cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 193–241. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_6
5. Garber, D.: Braid Group Cryptography. CoRR abs/0711.3941 (2007). <http://arxiv.org/abs/0711.3941>
6. SecureRF. <https://www.securerf.com/>
7. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: WalnutDSATM: a quantum-resistant digital signature algorithm. Cryptology ePrint Archive, Report 2017/058 (2017). <http://eprint.iacr.org/2017/058>
8. SecureRF and Intel collaboration delivers future-proof FPGA security solutions. <https://www.iot-now.com/2017/09/28/67603-securerf-intel-collaboration-delivers-future-proof-fpga-security-solutions/>
9. Ben-Zvi, A., Blackburn, S.R., Tsaban, B.: A practical cryptanalysis of the algebraic eraser. Cryptology ePrint Archive, Report 2015/1102 (2015). <http://eprint.iacr.org/2015/1102>
10. Myasnikov, A.D., Ushakov, A.: Cryptanalysis of Anshel-Anshel-Goldfeld-Lemieux key agreement protocol. Groups Complex. Cryptol. **1**(1), 63–75 (2009)

11. Kalka, A., Teicher, M., Tsaban, B.: Short expressions of permutations as products and cryptanalysis of the algebraic eraser. *Adv. Appl. Math.* **49**(1), 57–76 (2012)
12. Garside, F.A.: The braid group and other groups. *Q. J. Math.* **20**(1), 235–254 (1969)
13. Birman, J., Gebhardt, V., González-Meneses, J.: Conjugacy in Garside groups I: cyclings, powers and rigidity. *Groups. Geom. Dyn.* **1**, 221–279 (2007)
14. Dehornoy, P.: A fast method for comparing braids. *Adv. Math.* **125**(2), 200–235 (1997)
15. Tillich, J.P., Zémor, G.: Hashing with SL_2 . In: CRYPTO, pp. 40–49 (1994)
16. Petit, C., Lauter, K., Quisquater, J.-J.: Full cryptanalysis of LPS and morgenstern hash functions. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 263–277. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85855-3_18
17. Petit, C., Quisquater, J.-J.: Preimages for the Tillich-Zémor hash function. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 282–301. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_20
18. Babai, L., Hayes, T.: Near-independence of permutations and an almost sure polynomial bound on the diameter of the symmetric group. In: SODA, pp. 1057–1066 (2005)
19. Petit, C., Quisquater, J.-J.: Rubik’s for cryptographers. *Not. Am. Math. Soc.* **60**, 733–739 (2013)
20. Katz, J., Lindell, Y.: *Introduction to Modern Cryptography*. 2nd edn. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press/Taylor & Francis Group, Boca Raton (2014)
21. Epstein, D.B.A., Paterson, M.S., Cannon, J.W., Holt, D.F., Levy, S.V., Thurston, W.P.: *Word Processing in Groups*. A. K. Peters Ltd., Natick (1992)
22. Elrifai, E.A., Morton, H.: Algorithms for positive braids. *Q. J. Math.* **45**(4), 479–497 (1994)
23. Gebhardt, V.: A new approach to the conjugacy problem in Garside groups. *J. Algebra* **292**(1), 282–302 (2005). *Computational Algebra*
24. Hughes, J., Tannenbaum, A.: Length-Based Attacks for Certain Group Based Encryption Rewriting Systems. CoRR cs.CR/0306032 (2003)
25. Garber, D., Kaplan, S., Teicher, M., Tsaban, B., Vishne, U.: Probabilistic solutions of equations in the braid group. *Adv. Appl. Math.* **35**(3), 323–334 (2005)
26. Myasnikov, A.D., Ushakov, A.: Length based attack and braid groups: cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 76–88. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_6
27. Hughes, J.: A linear algebraic attack on the AAFG1 braid group cryptosystem. In: Batten, L., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 176–189. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45450-0_15
28. Lee, S.J., Lee, E.: Potential weaknesses of the commutator key agreement protocol based on braid groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 14–28. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_2
29. Cheon, J.H., Jun, B.: A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 212–225. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_13
30. Babai, L., Seress, Á.: On the diameter of permutation groups. *Eur. J. Comb.* **13**(4), 231–243 (1992)

31. Helfgott, H.A.: Growth and generation in $SL_2(Z/pZ)$. *Ann. Math.* **167**(2), 601–623 (2008)
32. Pyber, L., Szabó, E.: Growth in finite simple groups of Lie type. *J. Am. Math. Soc.* **29**(1), 95–146 (2016)
33. Petit, C., Quisquater, J.: Cayley hash functions. In: van Tilborg, H.C.A., Jajodia, S. (eds.) *Encyclopedia of Cryptography and Security*, vol. 2, pp. 183–184. Springer, Heidelberg (2011). https://doi.org/10.1007/978-1-4419-5906-5_126
34. Zémor, G.: Hash functions and Cayley graphs. *Des. Codes Crypt.* **4**(4), 381–394 (1994)
35. Tillich, J.-P., Zémor, G.: Group-theoretic hash functions. In: Cohen, G., Litsyn, S., Lobstein, A., Zémor, G. (eds.) *Algebraic Coding 1993*. LNCS, vol. 781, pp. 90–110. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57843-9_12
36. Grassl, M., Ilic, I., Magliveras, S.S., Steinwandt, R.: Cryptanalysis of the Tillich-Zémor hash function. *J. Cryptol.* **24**(1), 148–156 (2011)
37. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. *J. Cryptol.* **22**(1), 93–113 (2009)
38. Tillich, J.-P., Zémor, G.: Collisions for the LPS expander graph hash function. In: Smart, N. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 254–269. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_15
39. 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
40. Anshel, I., Anshel, M., Goldfeld, D., Lemieux, S.: Key agreement, the algebraic eraserTM, and lightweight cryptography. In: *Algebraic Methods in Cryptography*. Contemporary Mathematics, vol. 418, pp. 1–34. American Mathematical Society, Providence (2006)
41. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols, pp. 62–73. ACM Press (1993)
42. Birman, J.S.: Braids, Links, and Mapping Class Groups: *Annals of Mathematics Studies*, vol. 82. Princeton University Press, Princeton (1975)
43. Waterhouse, W.C.: Two generators for the general linear groups over finite fields. *Linear Multilinear Algebra* **24**(4), 227–230 (1989)
44. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. *J. Cryptol.* **12**, 1–28 (1999)
45. Quisquater, J.-J., Delescaille, J.-P.: How easy is collision search? Application to DES. In: Quisquater, J.-J., Vandewalle, J. (eds.) *EUROCRYPT 1989*. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_43
46. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. *J. Symb. Comput.* **24**(3–4), 235–265 (1997). *Computational algebra and number theory* (London, 1993)



Speed-Ups and Time–Memory Trade-Offs for Tuple Lattice Sieving

Gottfried Herold¹, Elena Kirshanova^{1(✉)}, and Thijs Laarhoven²

¹ Laboratoire LIP, ENS de Lyon, Lyon, France

`{gottfried.herold,elena.kirshanova}@ens-lyon.fr`

² Eindhoven University of Technology, Eindhoven, The Netherlands
`mail@thijs.com`

Abstract. In this work we study speed-ups and time–space trade-offs for solving the shortest vector problem (SVP) on Euclidean lattices based on tuple lattice sieving.

Our results extend and improve upon previous work of Bai–Laarhoven–Stehlé [ANTS’16] and Herold–Kirshanova [PKC’17], with better complexities for arbitrary tuple sizes and offering tunable time–memory trade-offs. The trade-offs we obtain stem from the generalization and combination of two algorithmic techniques: the configuration framework introduced by Herold–Kirshanova, and the spherical locality-sensitive filters of Becker–Ducas–Gama–Laarhoven [SODA’16].

When the available memory scales quasi-linearly with the list size, we show that with triple sieving we can solve SVP in dimension n in time $2^{0.3588n+o(n)}$ and space $2^{0.1887n+o(n)}$, improving upon the previous best triple sieve time complexity of $2^{0.3717n+o(n)}$ of Herold–Kirshanova. Using more memory we obtain better asymptotic time complexities. For instance, we obtain a triple sieve requiring only $2^{0.3300n+o(n)}$ time and $2^{0.2075n+o(n)}$ memory to solve SVP in dimension n . This improves upon the best double Gauss sieve of Becker–Ducas–Gama–Laarhoven, which runs in $2^{0.3685n+o(n)}$ time when using the same amount of space.

Keywords: Lattice-based cryptography

Shortest vector problem (SVP) · Nearest neighbor algorithms

Lattice sieving

1 Introduction

Lattice-based cryptography. Over the past few decades, lattice-based cryptography has emerged as a prime candidate for developing efficient, versatile, and (potentially) quantum-resistant cryptographic primitives; e.g. [Reg05, ADPS16]. The security of these primitives relies on the hardness of certain lattice problems, such as finding short lattice vectors. The fastest known method for solving many hard lattice problems is to use (a variant of) the BKZ lattice basis reduction algorithm [Sch87, SE94], which internally uses an algorithm for solving the

so-called Shortest Vector Problem (SVP) in lower-dimensional lattices: given a description of a lattice, the Shortest Vector Problem asks to find a shortest non-zero vector in this lattice. These SVP calls determine the complexity of BKZ, and hence an accurate assessment of the SVP hardness directly leads to sharper security estimates and tighter parameter choices for lattice-based primitives.

Algorithms for solving SVP. Currently, state-of-the-art algorithms for solving exact SVP can be classified into two groups, based on their asymptotic time and memory complexities in terms of the lattice dimension n : (1) algorithms requiring super-exponential time ($2^{\omega(n)}$) and $\text{poly}(n)$ space; and (2) algorithms requiring both exponential time and space ($2^{\Theta(n)}$). The former includes a family of so-called lattice enumeration algorithms [Kan83, FP85, GNR10], which currently perform best in practice and are used inside BKZ [Sch87, CN11]. The latter class of algorithms includes lattice sieving [AKS01, NV08, MV10], Voronoi-based approaches [AEVZ02, MV10, Laa16] and other techniques [BGJ14, ADRS15]. Due to the superior asymptotic scaling, these latter techniques will inevitably outperform enumeration in sufficiently high dimensions, but the large memory requirement remains a major obstacle in making these algorithms practical.

Heuristic SVP algorithms. In practice, only enumeration and sieving are currently competitive for solving SVP in high dimensions, and the fastest variants of both algorithms are based on *heuristic analyses*: by making certain natural (but unproven) assumptions about average-case behavior of these algorithms, one can (1) improve considerably upon worst-case complexity bounds, thus narrowing the gap between experimental and theoretical results; and (2) apply new techniques, supported by heuristics, to make these algorithms even more viable in practice. For enumeration, heuristic analyses of *pruning* [GNR10, AN17] have contributed immensely to finding the best pruning techniques, and making these algorithms as practical as they are today [LRBN]. Similarly, heuristic assumptions for sieving [NV08, MV10, Laa15a, BDGL16] have made these algorithms much more practical than their best provable counterparts [PS09, ADRS15].

Heuristic sieving methods. In 2008, Nguyen and Vidick [NV08] were the first to show that lattice sieving may be practical, proving that under certain heuristic assumptions, SVP can be solved in time $2^{0.415n+o(n)}$ and space $2^{0.208n+o(n)}$. Micciancio and Voulgaris [MV10] later described the so-called GaussSieve, which is expected to have similar asymptotic complexities as the Nguyen–Vidick sieve, but is several orders of magnitude faster in practice. Afterwards, a long line of work focused on locality-sensitive techniques succeeded in further decreasing the runtime exponent [WLTB11, ZPH13, BGJ14, Laa15a, LdW15, BL16].

Asymptotically the fastest known method for solving SVP is due to Becker et al. [BDGL16]. Using locality sensitive filters, they give a time–memory trade-off for SVP in time and space $2^{0.292n+o(n)}$, or in time $2^{0.368n+o(n)}$ when using only $2^{0.208n+o(n)}$ memory. A variant can even solve SVP in time $2^{0.292n+o(n)}$ retaining a memory complexity of $2^{0.208n+o(n)}$, but that variant is not compatible with

the Gauss Sieve (as opposed to the Nguyen–Vidick sieve) and behaves worse in practice.

Tuple lattice sieving. In 2016, Bai et al. [BLS16] showed that one can also obtain trade-offs for heuristic sieving methods in the other direction: reducing the memory requirement at the cost of more time. For instance, with a *triple sieve* they showed that one can solve SVP in time $2^{0.481n+o(n)}$, using $2^{0.189n+o(n)}$ space. Various open questions from [BLS16] were later answered by Herold and Kirshanova [HK17], who proved some of the conjectures from [BLS16], and greatly reduced the time complexity of the triple sieve to $2^{0.372n+o(n)}$. An open question remained whether these complexities were optimal, and whether it would be possible to obtain efficient time–memory trade-offs to interpolate between classical ‘double’ sieving methods and tuple lattice sieving.

1.1 Contributions

Results. In this work, we study both how to further speed up tuple lattice sieving, and how to obtain the best time–memory trade-offs for solving SVP with sieving¹. Our contributions include the following main results:

1. For triple sieving, we obtain a time complexity of $2^{0.3588n+o(n)}$ with a memory complexity of $2^{0.1887n+o(n)}$. This improves upon the previous best asymptotic time complexity of $2^{0.3717n+o(n)}$ of Herold and Kirshanova [HK17], and both the time *and* memory are better than the Gauss sieve algorithm of Becker et al. [BDGL16], which runs in time $2^{0.3685n+o(n)}$ and memory $2^{0.2075n+o(n)}$.
2. For triple sieving with arbitrary time–memory trade-offs, we obtain the trade-off curve depicted in Fig. 1, showing that our triple sieve theoretically outperforms the best double Gauss sieve up to a memory complexity of $2^{0.2437n+o(n)}$ (the intersection point of yellow and blue curves). For instance, with equal memory $2^{0.2075n+o(n)}$ as a double sieve, we can solve SVP in time $2^{0.3300n+o(n)}$, compared to the previous best $2^{0.3685n+o(n)}$ [BDGL16].
3. For larger tuple sizes (i.e., $k \geq 3$), in the regime when the space complexity is restricted to the input-sizes as considered by Bai et al. [BLS16] and Herold and Kirshanova [HK17], we improve upon all previous results. These new asymptotics are given in Table 3 on page 25.
4. Our experiments on lattices of dimensions 60 to 80 demonstrate the practicality of these algorithms, and highlight possible future directions for further optimizations of tuple lattice sieving.

Techniques. To obtain these improved time–memory trade-offs for tuple lattice sieving, this paper presents the following technical contributions:

¹ All our results are also applicable when we solve the closest vector problem (CVP) via sieving as was done in [Laa16]. Asymptotic complexities for CVP are the same as for SVP.

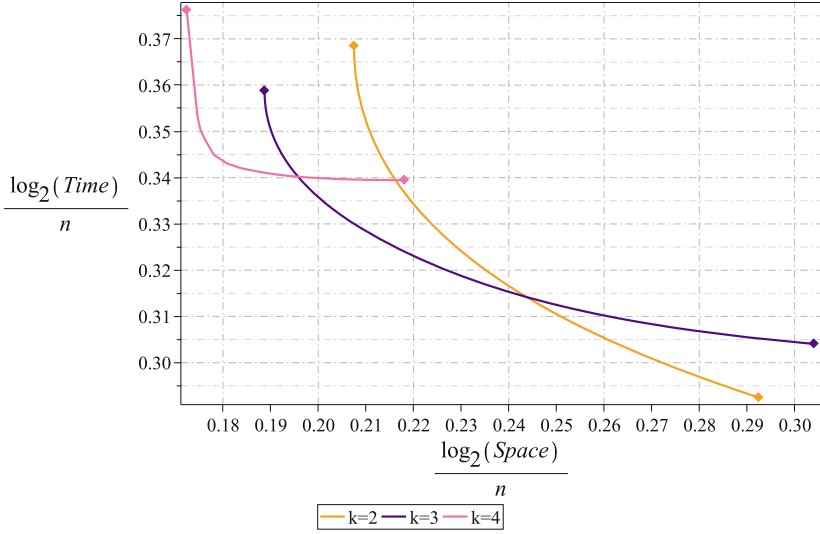


Fig. 1. Trade-offs for $k = 2, 3, 4$. Memory-exponents are on the X -axis, time-exponents are on the Y -axis. That means that for $x = m, y = t$, an algorithm will be of time-complexity $2^{t \cdot n + o(n)}$ and of memory-complexity $2^{m \cdot n + o(n)}$. Left-most points represent time and memory complexities for k -tuple sieving optimized for memory, right-most points represent complexities optimized for time.

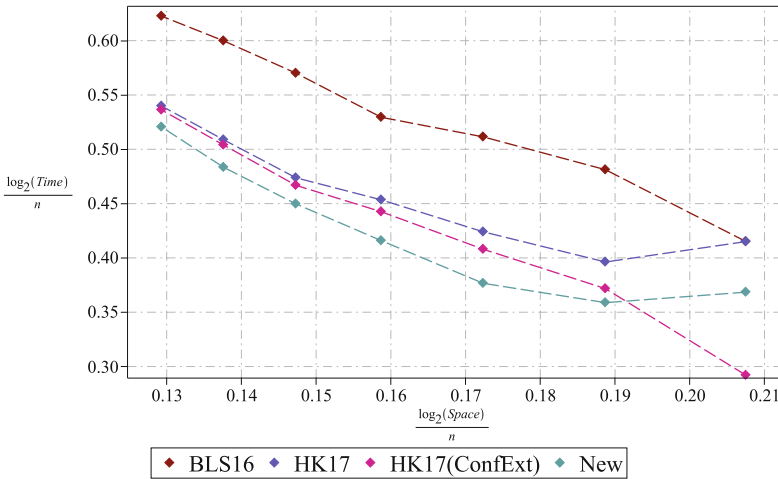


Fig. 2. Our runtime improvements for k -tuple lattice sieving over previous works for $3 \leq k \leq 8$ when we optimize for *memory*. For the $k = 2$ case, note that the results from [HK17] that use ConfExt with $T = 2^{0.292n + o(n)}$ cannot be applied to the Gauss-Sieve, but only to the NV-Sieve, which performs much worse in practice. Our results work for the Gauss-Sieve. See Sect. 4.2 for details.

1. We generalize the configuration search approach, first initiated by Herold and Kirshanova [HK17], to obtain optimized time–space trade-offs for tuple lattice sieving (Sect. 3).
2. We generalize the Locality-Sensitive Filters (LSF) framework of Becker et al. [BDGL16], and apply the results to tuple lattice sieving (Sect. 4). As an independent side result, we obtain explicit asymptotics for LSF for the approximate near neighbor problem on the sphere.²
3. We combine both techniques to obtain further improved asymptotic results compared to only using either technique (Sect. 5).

The remainder of the introduction is devoted to a high-level explanation of these techniques, and how they relate to previous work. We first introduce the approximate k -list problem. It serves as a useful abstraction of the tuple lattice sieving problem for which our results are *provable* – the results only become heuristic when applying them to tuple lattice sieving.

1.2 Approximate k -list Problem

The approximate k -list problem is the central computational problem studied in this paper. We denote by $S^{n-1} \subset \mathbb{R}^n$ the $(n - 1)$ -dimensional unit sphere. We use soft- $\tilde{\mathcal{O}}$ notation, e.g. $\tilde{\mathcal{O}}(2^n)$ means that we suppress sub-exponential factors.

Definition 1 (Approximate k -list problem). *Given k lists of i.i.d. uniformly random vectors $L_1, \dots, L_k \subset S^{n-1}$ and a target norm $t \in \mathbb{R}$, we are asked to find k -tuples $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in L_1 \times \dots \times L_k$ such that $\|\mathbf{x}_1 + \dots + \mathbf{x}_k\| \leq t$.*

We do not necessarily require to output all the solutions.

This problem captures the main subroutine of lattice sieving algorithms and allows us to describe precise, *provable* statements without any heuristic assumptions: all our results for the approximate k -list problem in the remainder of this paper are *unconditional*. In these application to lattice sieving, the lists will be identical (i.e., $|L_1| = \dots = |L_k|$) and the number of such k -tuples required to be output will be $\tilde{\mathcal{O}}(|L_1|)$.

To translate our results about the approximate k -list problem to lattice sieving, one needs to make additional heuristic assumptions, such as that the lists of lattice points appearing in sieving algorithms can be thought of as i.i.d. uniform vectors on a sphere (or a thin spherical shell). This essentially means that we do not ‘see’ the discrete structure of the lattice when we zoom out far enough, i.e. when the list contains very long lattice vectors. When the vectors in the list become short, we inevitably start noticing this discrete structure, and this heuristic assumption becomes invalid. Although experimental evidence suggests that these heuristic assumptions quite accurately capture the behavior of lattice

² The main difference with the works [ALRW17, Chr17], published after a preliminary version of some of these results [Laa15b], is that those papers focused on the case of list sizes scaling subexponentially in the dimension. Due to the application to lattice sieving, here we exclusively focus on exponential list sizes.

sieving algorithms on random lattices, the results in the context of lattice sieving can only be proven under these additional (unproven) assumptions.

Under these heuristic assumptions, any algorithm for the approximate k -list problem with $t < 1$ and $|L_1| = \dots = |L_k| = |L_{\text{out}}|$ will give an algorithm for SVP with the same complexity (up to polynomial factors). We dedicate Sect. 6 to such a SVP algorithm.

1.3 Generalized Configuration Search

By a concentration result on the distribution of scalar products of $\mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbb{S}^{n-1}$ previously shown in [HK17], the approximate k -list problem (Definition 1) can be reduced to the following configuration problem:

Definition 2 (Configuration problem). *Given k lists of i.i.d. uniform vectors $L_1, \dots, L_k \subset \mathbb{S}^{n-1}$ and a target configuration given by $C_{i,j}$'s, find a $1 - o(1)$ -fraction of k -tuples $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in L_1 \times \dots \times L_k$ s.t. all pairs $(\mathbf{x}_i, \mathbf{x}_j)$ in a tuple satisfy given inner-product constraints: $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \approx C_{i,j}$.*

We consider this problem only for k and $C_{i,j}$'s fixed. The approximation sign \approx in Definition 2 above is shorthand for $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle - C_{i,j}| \leq \varepsilon$ for some small $\varepsilon > 0$, as we deal with real values. This approximation will affect our asymptotical results as $\tilde{O}(2^{cn+\nu(\varepsilon)n})$ for some $\nu(\varepsilon)$ that tends to 0 as we let $\varepsilon \rightarrow 0$. Eventually, $\nu(\varepsilon)$ will be hidden in the \tilde{O} -notation and we usually omit it.

We arrange these constraints into a $k \times k$ real matrix C – the Gram matrix of the \mathbf{x}_i 's – which we call a *configuration*. The connection to the k -list problem becomes immediate once we notice that a k -tuple $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ with $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \approx C_{i,j}, \forall i, j$, produces a sum vector whose norm satisfies $\|\sum_i \mathbf{x}_i\|^2 = \sum_{i,j} C_{i,j}$.

Consequently, solving the configuration problem for an appropriate configuration C with $\sum_{i,j} C_{i,j} \leq t^2$ will output a list of solutions to the Approximate k -list problem. The result [HK17, Theorem 1] shows that this will in fact return almost all solutions for a certain choice of $C_{i,j}$, i.e., k -tuples that form short sums are concentrated around one specific set of inner product constraints $C_{i,j}$.

The main advantage of the configuration problem is that it puts *pair-wise* constraints on solutions, which significantly speeds up the search. Moreover, C determines the expected number of solutions via a simple but very useful fact: the probability that a uniform i.i.d. tuple $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ satisfies C is for fixed C, k given by (up to $\text{poly}(n)$ factors) $\det(C)^{n/2}$ [HK17, Theorem 2]. Hence, if our goal is to output $|L|$ tuples, given $|L|^k$ possible k -tuples from the input, we must have $|L|^k \cdot \det(C)^{n/2} = |L|$. Now there are two ways to manipulate this equation. We can either fix the configuration C and obtain a bound on $|L|$ (this is precisely what [HK17] does), or vary C and deduce the required list-sizes for a given C . The latter option has the advantage that certain choices of C may result in a faster search for tuples. In this work, we investigate the second approach and present the trade-offs obtained by varying C .

Let us first detail on trade-offs obtained by varying the target configuration C . In [HK17], lattice sieving was optimized for memory with the optimum

attained at to the so-called *balanced* configuration C (a configuration is called balanced if $C_{i,j} = -1/k, \forall i \neq j$). Such a configuration maximizes $\det(C)$, which in turn maximizes the number of k tuples that satisfy C . Hence, the balanced configuration minimizes the size of input lists (remember, in lattice-sieving we require the number of returned tuples be asymptotically equal to the input list size) and, hence, gives one of two extreme points of the trade-off.

Now assume we change the target configuration C . As the result, the number of returned tuples will be exponentially smaller than in the balanced case (as the value for $\det(C)$ decreases, the probability that a random k -tuple satisfies C decays by a factor exponential in n). To maintain the requirement on the size of the output, we need to increase the input lists. However, the *search* for tuples that satisfy C becomes faster for some choices of C . A choice for C with the fastest search gives another extreme point of the trade-off. In Sect. 3, we analyze the algorithm for the configuration problem and explain why certain C 's result in faster algorithms. For small k , we give explicit time–memory trade-off curves.

1.4 Generalized Locality-Sensitive Filters

Our second contribution improves the running time for the configuration search using a near neighbor method called spherical locality-sensitive filtering (LSF), first introduced in the context of lattice sieving in [BDGL16]. This method was later shown to be optimal for other applications in [ALRW17, Chr17].

LSF is an algorithm which receives on input a (typically large) set of points, a so-called query point usually not from the set, and a target distance d . It returns all points from the set that are within target distance d from the query point (the metric can be any, in our case, it will be angular). The aim of LSF is to answer many such queries fast by cleverly preprocessing this set so that the time of preprocessing is amortized among many query points. Depending on the choice of preprocessing, LSF may actually require more memory than the size of the input set. This expensive preprocessing results in faster query complexity, and the whole algorithm can be optimized (either for time or for memory) when we know how many query points we have.

In the application to tuple lattice sieving, we make use of the locality-sensitive filtering technique of [BDGL16] to speed up the configuration search routine. Time–memory trade-offs offered by LSF naturally translate to time–memory trade-offs for configuration search and, hence, for sieving.

There are several ways we can make use of LSF. First, we can apply LSF to the balanced configuration search and remain in the minimal memory regime (i.e., the memory bound for the LSF data structure is upper-bounded by the input list sizes). Interestingly, even in such a restricted regime we can speed up the configuration search and, in turn, asymptotically improve lattice sieving. Secondly, we allow LSF to use more memory while keeping the target configuration balanced. This has the potential to speed up the query cost leading to a faster configuration search. We can indeed improve k -tuple sieving in this regime for $k = 2, 3, 4$. In Sect. 4 we give exact figures in both aforementioned LSF scenarios.

1.5 Combining Both Techniques

Finally, we can combine LSF with changing the target configuration C . We search for an optimal C taking into account that we exploit LSF as a subroutine in the search for tuples satisfying C . Essentially, if we write out all the requirements on input/output list sizes and running time formulas, the search for such optimal C becomes a high-dimensional optimization problem (the number of variables to optimize for is of order k^2). Here, ‘optimal’ C may either mean the configuration that minimizes time, or memory, or time given a bound on memory. When we optimize C for time, we obtain exponents given in Table 1. Interestingly, for $k = 2, 3$, k -tuple sieve achieves the ‘time=memory’ regime.

Table 1. Asymptotic complexities when using **LSF** and **arbitrary configurations**, when **optimizing for time**. I.e., we put no restriction on memory, but using more memory than stated in the table does not lead to a faster algorithm.

Tuple size (k)	2	3	4	5	6
Time	0.2925	0.3041	0.3395	0.3459	0.4064
Space	0.2925	0.3041	0.2181	0.2553	0.2435

To obtain trade-off curves for $k = 2, 3, 4$, we set-up an optimization problem for several memory bounds and find the corresponding solutions for time. The curves presented in Fig. 1 are the best curve-fit for the obtained (memory, time) points. The right-most diamond-shaped points on the curves represent k -tuple algorithms when optimized for time (Table 1). The left-most are the points when we use the smallest possible amount of memory for this tuple size (see Table 3 in Sect. 4 for exact numbers). The curves do not extend further to the left since the k -tuple search will not succeed in finding tuples if the starting lists will be below a certain bound.

Already for $k = 4$, the optimization problem contains 27 variables and non-linear inequalities, so we did not attempt to give full trade-off curves for higher k ’s. We explain the constraints for such an optimization problem later in Sect. 5 and provide access to the Maple program we used. Extreme points for larger k ’s are given in Tables 1 and 3 in Sect. 4.

1.6 Open Problems

Although this work combines and optimizes two of the most prominent techniques for tuple lattice sieving, some open questions for future work remain, which we state below:

- As explained in [Laa16], sieving (with LSF) can also be used to solve CVPP (the closest vector problem *with preprocessing*) with time–memory trade-offs depending on the size of the preprocessed list and the LSF parameters. Tuple sieving would provide additional trade-off parameters in terms of the tuple size and the configuration to search for, potentially leading to better asymptotic time–memory trade-offs for CVPP.

- Similar to [LMvdP15], it should be possible to obtain asymptotic quantum speed-ups for sieving using Grover’s algorithm. Working out these quantum trade-offs (and potentially finding other non-trivial applications of quantum algorithms to tuple sieving) is left for future work.
- An important open problem remains to study what happens when k is super-constant in n . Unfortunately, our analysis crucially relies on k being fixed, as various subexponential terms depend on k .

2 Preliminaries

We denote vectors by bold letters, e.g., \mathbf{x} . In this paper we consider the ℓ_2 -norm and denote the length of a vector \mathbf{x} by $\|\mathbf{x}\|$. We denote by $S^{n-1} \subset \mathbb{R}^n$ the $(n-1)$ -dimensional unit sphere. For any square matrix $C \in \mathbb{R}^{k \times k}$ and $I \subset \{1, \dots, k\}$, we denote by $C[I]$ the $|I| \times |I|$ submatrix of C obtained by restricting to the rows and columns indexed by I .

Lattices. Given a basis $B = \{\mathbf{b}_1, \dots, \mathbf{b}_d\} \subset \mathbb{R}^n$ of linearly independent vectors, the lattice generated by B , denoted $\mathcal{L}(B)$, is given by $\mathcal{L}(B) := \{\sum_{i=1}^d \lambda_i \mathbf{b}_i : \lambda_i \in \mathbb{Z}\}$. For simplicity of exposition, we assume $d = n$, i.e. the lattices considered are full rank. One of the central computational problems in the theory of lattices is the shortest vector problem (SVP): given a basis B , find a shortest non-zero vector in $\mathcal{L}(B)$. The length of this vector, known as the first successive minimum, is denoted $\lambda_1(\mathcal{L}(B))$.

The fastest (heuristic) algorithms for solving SVP in high dimensions are based on lattice sieving, originally described in [AKS01] and later improved in e.g. [NV08, PS09, MV10, Laa15a, LdW15, BL16, BDGL16]. These algorithms start by sampling an exponentially large list L of $2^{\ell n}$ (long) lattice vectors. The points from L are then iteratively combined to form shorter and shorter lattice points as $\mathbf{x}_{\text{new}} = \mathbf{x}_1 \pm \mathbf{x}_2 \pm \dots \pm \mathbf{x}_k$ for some k .³ The complexity is determined by the cost to find k -tuples whose combination produces shorter vectors.

In order to improve and analyze the cost of sieving algorithms, we consider the following problem, adapted from [HK17], generalizing Definition 1.

Definition 3 (Approximate k -list problem [HK17]). *Given k lists L_1, \dots, L_k of respective exponential sizes $2^{\ell_1 n}, \dots, 2^{\ell_k n}$ whose elements are i.i.d. uniformly random vectors from S^{n-1} , the approximate k -list problem consists of finding $2^{\ell_{\text{out}} n}$ k -tuples $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in L_1 \times \dots \times L_k$ that satisfy $\|\mathbf{x}_1 + \dots + \mathbf{x}_k\| \leq t$, using at most $M = \tilde{O}(2^{mn})$ memory.*

We are interested in the asymptotic complexity for $n \rightarrow \infty$ with all other parameters fixed. Note that the number of solutions to the problem is concentrated around its expected number and since we only want to solve the problem with high probability, we will work with expected list sizes throughout. See Appendix A in the full version [HKL] for a justification. We only consider cases

³ For the approximate k -list problem, we stick to all + signs. This limitation is for analysis only, does not affect asymptotics and is easy to solve in practice.

where $m \geq \ell_{\text{out}}$ and where the expected number of solutions is at least $\tilde{\Omega}(2^{\ell_{\text{out}}n})$. Part of our improvements over [HK17] comes from the fact that we consider the case where the total number of solutions to the approximate k -list problem is exponentially larger than $2^{\ell_{\text{out}}n}$. By only requiring to find an exponentially small fraction of solutions, we can focus on solutions that are easier to find.

In the applications to sieving, we care about the case where $\ell_1 = \dots = \ell_k = \ell_{\text{out}}$ and $t = 1$. Allowing different ℓ_i is mostly done for notational consistency with Definition 5 below, where different ℓ_i 's naturally appear as subproblem in a recursive analysis of our algorithm.

2.1 Configurations and Concentration Results

One of the main technical tools introduced by [HK17] is so-called configurations. We repeat their definitions and results here, adapted to our case.

Definition 4 (Configuration [HK17, Definition 2]). *The configuration $C = \text{Conf}(\mathbf{x}_1, \dots, \mathbf{x}_k)$ of k points $\mathbf{x}_1, \dots, \mathbf{x}_k$ from the n -sphere is defined to be the Gram matrix of the \mathbf{x}_i , i.e. $C_{i,j} := \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.*

The connection between the approximate k -list problem and configurations is as follows: the configuration of a k -tuple $\mathbf{x}_1, \dots, \mathbf{x}_k$ determines the length $\|\sum_i \mathbf{x}_i\|$:

$$\left\| \sum_i \mathbf{x}_i \right\|^2 = \sum_{i,j} C_{i,j} = \mathbf{1}^t C \mathbf{1}, \tag{1}$$

where $\mathbf{1}$ is a column vector of 1's. So a k -tuple is a solution to the approximate k -list problem if and only if their configuration satisfies $\mathbf{1}^t C \mathbf{1} \leq t^2$. Let

$$\begin{aligned} \mathcal{C} &= \{C \in \mathbb{R}^{k \times k} \mid C \text{ symmetric positive semi-definite, } C_{i,i} = 1\} \\ \mathcal{C}_{\text{good}} &= \{C \in \mathcal{C} \mid \mathbf{1}^t C \mathbf{1} \leq t^2\} \end{aligned}$$

be the set of all possible configuration resp. of the configurations of solutions. We call the latter *good* configurations.

Following [HK17], rather than only looking for k -tuples that satisfy $\|\sum_i \mathbf{x}_i\| \leq t$, we look for k -tuples that additionally satisfy a constraint on their configuration as in the following problem (generalizing Definition 2).

Definition 5 (Configuration problem [HK17]). *Let $k \in \mathbb{N}$, let $m > 0$, let $\varepsilon > 0$, and suppose we are given a target configuration $C \in \mathcal{C}$. Given k lists L_1, \dots, L_k of respective exponential sizes $2^{\ell_1 n}, \dots, 2^{\ell_k n}$ whose elements are i.i.d. uniform from S^{n-1} , the k -list configuration problem asks to find a $1-o(1)$ fraction of all solutions using at most $M = \tilde{O}(2^{mn})$ memory, where a solution is a k -tuple $\mathbf{x}_1, \dots, \mathbf{x}_k$ with $\mathbf{x}_i \in L_i$ such that $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle - C_{i,j}| \leq \varepsilon$ for all i, j .*

Clearly, solving the configuration problem for any good configuration C yields solutions to the approximate k -list problem. If the number of solutions to the

configuration problem is large enough, this is then sufficient to solve the approximate k -list problem. The number of expected solutions for a given configuration C can be easily determined from the distribution of $\text{Conf}(\mathbf{x}_1, \dots, \mathbf{x}_k)$ for i.i.d. uniform $\mathbf{x}_i \in S^{n-1}$. For this we have

Theorem 1 (Distribution of configurations [HK17]). *Let $\mathbf{x}_1, \dots, \mathbf{x}_k$ be independent uniformly distributed from S^{n-1} with $n > k$. Then their configuration $C = \text{Conf}(\mathbf{x}_1, \dots, \mathbf{x}_k)$ follows a distribution with density function*

$$\mu = W_{n,k} \cdot \det(C)^{\frac{1}{2}(n-k)} dC_{1,2} \dots dC_{n-1,n}, \tag{2}$$

where $W_{n,k} = \mathcal{O}_k(n^{\frac{1}{4}(k^2-k)})$ is an (explicitly known) normalization constant that only depends on n and k .

This implies that we need to solve the configuration problem for any good configuration such that $\prod_i |L_i| \cdot (\det C)^{n/2} = \tilde{\Omega}(2^{\ell_{\text{out}}n})$.

In [HK17], the authors ask to find *essentially all* solutions to the approximate k -list problem. For this, they solve the configuration problem for a particular target configuration \bar{C} , such that the solutions to the configuration problem for \bar{C} comprise a $1 - o(1)$ -fraction of the solutions to the approximate k -list problem. \bar{C} has the property that all non-diagonal entries are equal; such configurations are called *balanced*. In the case $t = 1$, we have $\bar{C}_{i,j} = -\frac{1}{k}$ for $i \neq j$.

By contrast, we are fine with only finding *enough* solutions to the approximate k -list problem. This relaxation allows us to consider other, non-balanced, configurations.

2.2 Transformation

In our application, we will have to deal with lists L whose elements are not uniform from S^{n-1} . Instead, the elements $\mathbf{x} \in L$ have prescribed scalar products $\langle \mathbf{v}_i, \mathbf{x} \rangle$ with some points $\mathbf{v}_1, \dots, \mathbf{v}_r$.

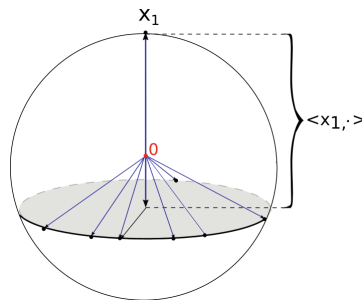


Fig. 3. Taking a vector $\mathbf{x}_1 \in L_1$ and applying filtering to $L_2, \dots, L_k \subset S^{n-1}$ w.r.t. \mathbf{x}_1 fixes the distances between \mathbf{x}_1 and all the vectors from L_2, \dots, L_k that ‘survive’ the filtering. All filtered vectors (i.e., vectors from $L_2^{(1)}, \dots, L_k^{(1)}$) can be considered as vectors from S^{n-2} – a scaled sphere of one dimension less.

The effect of these restriction is that the elements $\mathbf{x} \in L$ are from some (shifted, scaled) sphere of reduced dimension S^{n-1-r} , cf. Fig. 3. We can apply a transformation (TRANSFORM in Algorithm 1 below), eliminating the shift and rescaling. This reduces the situation back to the uniform case, allowing us to use recursion. Note that we have to adjust the target scalar products to account for the shift and scalings. A formal statement with formulas how to adjust the scalar products is given by Lemma 3 in Appendix B in the full version [HKL].

Algorithm 1. Recursive algorithm for the configuration problem

Input: L_1, \dots, L_k – lists of vectors from S^{n-1} , $C_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \in \mathbb{R}^{k \times k}$ – Gram matrix, $\varepsilon > 0$ – fudge-factor.
Output: L_{out} – list of k -tuples $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in L_1 \times \dots \times L_k$, s.t. $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle - C_{ij}| \leq \varepsilon$ for all i, j .

```

1: function SOLVECONFIGURATIONPROBLEM( $L_1, \dots, L_k, \varepsilon, C_{i,j} \in \mathbb{R}^{k \times k}$ )
2:    $L_{\text{out}} \leftarrow \emptyset$ 
3:   if  $k = 1$  then
4:      $L_{\text{out}} \leftarrow L_1$ 
5:   else
6:     for all  $\mathbf{x}_1 \in L_1$  do
7:       for all  $j = 2 \dots k$  do
8:          $L_j^f \leftarrow \text{FILTER}(\mathbf{x}_1, L_j, C_{1,j}, \varepsilon)$  ▷ Create all filtered lists
9:          $L_j', C' \leftarrow \text{TRANSFORM}(\mathbf{x}_1, L_j^f, C, i)$  ▷ Remove the dependency on  $\mathbf{x}_1$ 
10:         $\hat{L} \leftarrow \text{SOLVECONFIGURATIONPROBLEM}(L_2', \dots, L_k', \varepsilon, C_{[2..k],[2..k]})$ 
11:        for all  $\mathbf{x} \in \hat{L}$  do
12:           $L_{\text{out}} \leftarrow L_{\text{out}} \cup \{(\mathbf{x}_1, \mathbf{x})\}$  ▷ Append  $\mathbf{x}$  to all tuples that contain  $\mathbf{x}_1$ 
13:   return  $L_{\text{out}}$ 

```

```

1: function FILTER( $\mathbf{v}, L, c, \varepsilon$ )
2:    $L' \leftarrow \emptyset$ 
3:   for all  $\mathbf{x} \in L$  do
4:     if  $|\langle \mathbf{x}, \mathbf{v} \rangle - c| \leq \varepsilon$  then
5:        $L' \leftarrow L' \cup \{\mathbf{x}\}$ 
6:   return  $L'$ 

```

```

1: function TRANSFORM( $\mathbf{v}, L, C, i$ ) ▷ Changes the  $i^{\text{th}}$  row of  $C$ 
2:    $L' \leftarrow \emptyset$ 
3:   Transform all  $C_{i,j}$  to  $C'_{j,k}$  for  $j, k > i$  as follows:

```

$$C'_{j,k} = \frac{1}{\sqrt{(1 - C_{i-1,j}^2)(1 - C_{i-1,k}^2)}}(C_{j,k} - C_{i-1,j} \cdot C_{i-1,k})$$

(see Lemma 3 in Appendix B of the full version for justification).

```

4:   for all  $\mathbf{x} \in L$  do
5:      $\mathbf{x}^\perp \leftarrow \mathbf{x} - \langle \mathbf{x}, \mathbf{v} \rangle \mathbf{v}$ 
6:      $L' \leftarrow L' \cup \left\{ \frac{\mathbf{x}^\perp}{\|\mathbf{x}^\perp\|} \right\}$ 
7:   return  $L', C'$ 

```

3 Generalized Configuration Search

Now we present our algorithm for the Configuration problem (Definition 5). We depict the algorithm in Fig. 4. Its recursive version is given in Algorithm 1.

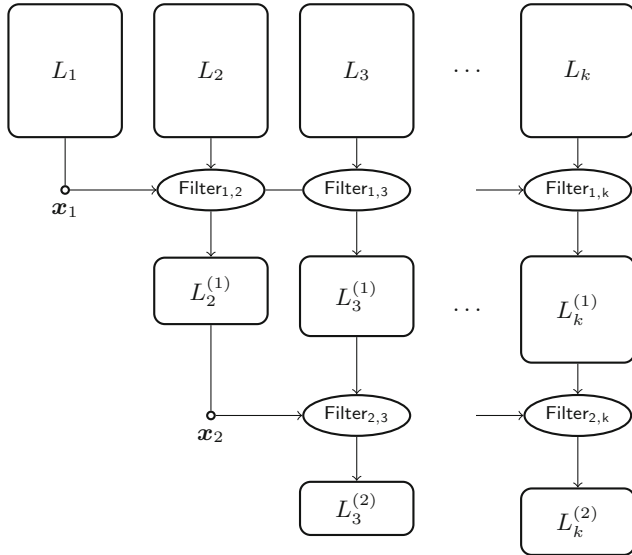


Fig. 4. Our algorithm for the Configuration problem. Procedures $\text{Filter}_{i,j}$ receive on input a vector (e.g. \mathbf{x}_1), a list of vectors (e.g. L_2), and a real number $C_{i,j}$ - the target inner product. It creates another shorter list (e.g. $L_2^{(1)}$) that contains all vectors from the input list whose inner product with the input vector is within some small ε from the target inner product. Time–memory trade-offs are achieved by taking different values $C_{i,j}$'s for different i, j . In particular, an asymptotically faster k -list algorithm can be obtained by more ‘aggressive’ filtering to the left-most lists on each level. In other words, for fixed i , the value $C_{i,j}$ is the largest (in absolute value) for $j = i + 1$.

The algorithm receives on input k lists L_1, \dots, L_k of exponential (and potentially different) sizes and the target configuration $C \in \mathbb{R}^{k \times k}$. It outputs a list L_{out} of tuples $\mathbf{x}_1, \dots, \mathbf{x}_k \in L_1 \times \dots \times L_k$ s.t. $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle - C_{i,j}| \leq \varepsilon$ for all i, j .

The algorithm processes the lists from left to right (see Fig. 4). Namely, for each $\mathbf{x}_1 \in L_1$, it creates lists $L_2^{(1)}, \dots, L_k^{(1)}$ by applying a filtering procedure w.r.t. \mathbf{x}_1 to these $k - 1$ lists L_2, \dots, L_k . This filtering takes as input a vector, a list L_j , and target inner-product $C_{1,j}$ for all $j \geq 2$. Its output $L_j^{(1)}$ contains all the vectors \mathbf{x}_j from L_j that satisfy $|\langle \mathbf{x}_1, \mathbf{x}_j \rangle - C_{1,j}| \leq \varepsilon$ for some small fixed $\varepsilon > 0$. Note the upper index of $L_j^{(1)}$: it denotes the number of filterings applied to the original list L_j .

Applying filtering to all $k - 1$ lists, we obtain $k - 1$ new lists that contain vectors with a fixed angular distance to \mathbf{x}_1 (see Fig. 3). Now our task is to find

all the $k - 1$ tuples $(\mathbf{x}_2, \dots, \mathbf{x}_k) \in L_2^{(1)}, \dots, L_k^{(1)}$ that satisfy $C[2 \dots k]$. Note that this is almost an instance of the $(k - 1)$ configuration problem, except for the distribution of the filtered elements. To circumvent this issue we apply to all elements of filtered lists the transformation described in Lemma 2 (Appendix B, full version [HKL]), where our fixed \mathbf{x}_1 plays the role of \mathbf{v} .

It is easy to see that the time needed for the transformation is equal to the size of filtered lists and thus, asymptotically irrelevant. Finally, we can apply the algorithm recursively to the $k - 1$ transformed lists.

Note that the transformation is merely a tool for analysis and the algorithm can easily be implemented without it. We only need it for the analysis of the LSF variant in Sect. 4.

3.1 Analysis

In this section we analyse the complexity of Algorithm 1. Speed-ups obtained with Nearest Neighbor techniques are discussed in the next section. In this ‘plain’ regime, the memory complexity is completely determined by the input list sizes. Applying filtering can only decrease the list sizes. Recall that k is assumed to be a fixed constant. Further, as our algorithm is exponential in n and we are only interested in asymptotics, in our analysis we ignore polynomial factors, i.e. computations of inner-products, summations, etc.

Our main objective is to compute the (expected) size of lists on each level (by the term ‘level’ we mean the number of filterings applied to a certain list or, equivalently, the depth of the recursion). We are interested in $|L_i^{(j)}|$ - the size of list L_i after application of j filterings. For the input lists, we set $L_i^{(0)} := L_i$.

Once we know how the list-sizes depend on the chosen configuration C , it is almost straightforward to conclude on the running time. Consider the j^{th} recursion-level. On this level, we have j points $(\mathbf{x}_1, \dots, \mathbf{x}_j)$ fixed during the previous recursive calls and the lists $L_{j+1}^{(j-1)}, \dots, L_k^{(j-1)}$ of (possibly) different sizes which we want to filter wrt. \mathbf{x}_j . Asymptotically, all filterings are done in time $\max_{j+1 \leq i \leq k} |L_i^{(j-1)}|$. This process will be recursively called as many times as many fixed tuples $(\mathbf{x}_1, \dots, \mathbf{x}_j)$ we have, namely, $\prod_{r=1}^j |L_r^{(r-1)}|$ times (i.e., the product of all left-most list-sizes). Hence, the cost to create *all* lists on level j can be expressed as

$$T_j = \prod_{r=1}^j |L_r^{(r-1)}| \cdot \max_{j+1 \leq i \leq k} \left\{ |L_i^{(j-1)}|, 1 \right\}. \tag{3}$$

In the above formula, considering the maximum between a filtered list and 1 takes care about the lists of size 0 or 1 (i.e., the exponent for the list-sizes might be negative). Assume on a certain level all filtered lists contain only one or no elements in expectation. Technically, to create the next level, we still need to check if these lists are indeed empty or not. This also means that the level above – the one that created these extremely short lists – takes more time than the subsequent level.

Finally, the total running time of the algorithm is determined by the level j for which T_j is maximal (we refer to such a level as dominant):

$$T = \max_{1 \leq j \leq k} \left[\prod_{r=1}^j |L_r^{(r-1)}| \cdot \max_{j+1 \leq i \leq k} |L_i^{(j-1)}| \right]. \quad (4)$$

Note that we do not need to take into account using lists of expected sizes ≤ 1 . As mentioned above, creating these lists takes more time than using them.

In the following lemma, we use the results of Theorem 1 to determine $|L_i^{(j)}|$ for $0 \leq j \leq i-1, 1 \leq i \leq k$. We denote by $C[1 \dots j, i]$ the $(j+1) \times (j+1)$ the submatrix of C formed by restricting its columns and rows to the set of indices $\{1, \dots, j, i\}$.

Lemma 1. *During a run of Algorithm 1 that receives on input a configuration $C \in \mathbb{R}^{k \times k}$ and lists L_1, \dots, L_k , the intermediate lists $L_i^{(j)}$ for $1 \leq i \leq k, i-1 \leq j \leq k$ are of expected sizes*

$$\mathbb{E}[|L_i^{(j)}|] = |L_i| \cdot \left(\frac{\det(C[1 \dots j, i])}{\det(C[1 \dots j])} \right)^{n/2}. \quad (5)$$

Proof. The list $L_i^{(j)}$ is created when we have the tuple $(\mathbf{x}_1, \dots, \mathbf{x}_j)$ fixed. The probability for such a tuple to appear is $\det(C[1 \dots j])^{n/2}$. Moreover, the probability that we ever consider a tuple $(\mathbf{x}_1, \dots, \mathbf{x}_j, \mathbf{x}_i)$ for any $\mathbf{x}_i \in L_i$ is given by $(\det(C[1 \dots j, i]))^{n/2}$. The result follows when we take the latter probability conditioned on the event that $(\mathbf{x}_1, \dots, \mathbf{x}_j)$ is fixed.

Using the result of Theorem 1 once again, we obtain the expected number of the returned tuples, i.e. the size of L_{out} .

Corollary 1. *Algorithm 1 receiving on input a configuration $C \in \mathbb{R}^{k \times k}$ and the lists L_1, \dots, L_k of respective sizes $2^{\ell_1 n}, \dots, 2^{\ell_k n}$, outputs $|L_{\text{out}}|$ solutions to the configuration problem, where*

$$\mathbb{E}[|L_{\text{out}}|] = 2^{(\sum_i^k \ell_i) \cdot n} \cdot (\det C)^{n/2}. \quad (6)$$

In particular, if all k input lists are of the same size $|L|$ and the output list size is required to be of size $|L|$ as well (the case of sieving), we must have

$$|L| = (\det C)^{-\frac{n}{2(k-1)}}. \quad (7)$$

Proof. The statement follows immediately from the total number of k -tuples and the fact that the probability that a random k -tuple has the desired configuration is (up to polynomial factors) $(\det C)^{n/2}$ (see Theorem 1).

We remark that our Algorithm 1 outputs all k -tuples that satisfy a given configuration C . This is helpful if one wants to solve the k -list problem using the output of Algorithm 1. All one needs to do is to provide the algorithm with the lists L_1, \dots, L_k and a configuration C , such that according to Eq. (6), we expect

to obtain $2^{\ell_{\text{out}}}$ k -tuples. Furthermore, we require $\mathbf{1}^t C \mathbf{1} \leq t^2$, so $C \in \mathcal{C}_{\text{good}}$ and every solution to the configuration problem is a solution to the k -list problem.

Note also that the k -list problem puts a bound M on the memory used by the algorithm. This directly translates to the bound on the input lists for the configuration problem (recall that filtering only shrinks the lists).

The above discussion combined with Lemma 1 yields the next theorem, which states the complexity of our algorithm for the k -list problem.

Theorem 2. *Algorithm 1 is expected to output $|L_{\text{out}}| = 2^{\ell_{\text{out}}}$ solutions to the k -list problem in time T provided that the input L_1, \dots, L_k and $C \in \mathcal{C}_{\text{good}}$ satisfy Eq. (6) and $\max_i |L_i| \leq M$, where T is (up to polynomial factors) equal to*

$$\max_{1 \leq j \leq k} \left[\prod_{r=1}^j |L_r| \left(\frac{\det C[1 \dots r]}{\det C[1 \dots r-1]} \right)^{\frac{n}{2}} \cdot \max_{j+1 \leq i \leq k} |L_i| \left(\frac{\det(C[1 \dots j-1, i])}{\det(C[1 \dots j-1])} \right)^{\frac{n}{2}} \right] \quad (8)$$

Note that we miss exponentially many solutions to the k -list problem, yet for the configuration search, we expect to obtain all the tuples that satisfy the given target configuration C . This loss can be compensated by increased sizes of input lists. Since the running time T (see Eq. (8)) depends on C and on the input list-sizes in a rather intricate manner, we do not simplify the formula for T , but rather discuss an interesting choice for input parameters in case $k = 3$.

3.2 Case of Interest: $k = 3$

The case $k = 3$ is the most relevant one if we want to apply our 3-list algorithm to SVP sieving algorithms (see Sect. 6 for a discussion on the k -Gauss sieve algorithm for SVP). In [HK17], the k -sieve algorithm was proved to be the most time-efficient when $k = 3$ among all the non-LSF based algorithms. In particular, a 3-sieve was shown to be faster than non-LSF 2-sieve.

To be more concrete, the 3-list problem (the main subroutine of the 3-Gauss sieve) can be solved using the *balanced* configuration search (i.e., $C_{i,j} = -1/3$ for $i \neq j$) in time $T_{\text{bal}} = 2^{0.396n}$ requiring memory $M = 2^{0.1887n}$. Up to $\text{poly}(n)$ factors these numbers are also the complexities of the 3-Gauss sieve algorithm. The main question we ask is whether we can reduce the time T at the expense of the memory M ?

If we take a closer look at the time complexity of the 3-Configuration search for the balanced C , we notice that among the two levels – on the first level we filter wrt. $\mathbf{x}_1 \in L_1$, on the second wrt. $\mathbf{x}_2 \in L_2^{(1)}$ – the second one dominates. In other words, if we expand Eq. (8), we obtain $T = \max\{T_1 = 2^{0.377n}, T_2 = 2^{0.396n}\} = T_2$. We denote by T_i the time needed to create all lists on the i^{th} level (see Eq. (3)). Hence, there is potential to improve T by putting more work on the first level hoping to reduce the second one.

This is precisely what our optimization for T, M , and configuration C suggests: (1) increase the input list sizes from $2^{0.1887n}$ to $2^{0.1895n}$, and (2) use more ‘aggressive’ filtering on the first level to balance out the two levels. We spend

more time creating $L_2^{(1)}$ and $L_3^{(1)}$ as the input lists became larger, but the filtered $L_2^{(1)}, L_3^{(1)}$ are shorter and contain more ‘promising’ pairs. With the configuration C given below, the time complexity of the 3-Configuration search becomes $T = \max\{T_1 = T_2\} = 2^{0.3789n}$ with input and output list sizes equal to $2^{0.1895n}$. We remark that in our optimization we put the constraint that $|L_{\text{out}}| = |L_1|$.

$$C = \begin{pmatrix} 1 & -0.3508 & -0.3508 \\ -0.3508 & 1 & -0.2984 \\ -0.3508 & -0.2984 & 1 \end{pmatrix}.$$

In fact, for $k = 3, \dots, 8$, when we optimize for time (cf. Table 2), the same phenomenon occurs: the time spent on each level is the same. We conjecture that such a feature of the optimum continues for larger values of k , but we did not attempt to prove it.

3.3 Trade-off Curves

Now we demonstrate time-memory trade-offs for larger values of k . In Fig. 5 we plot the trade-off curves obtained by changing of the target configuration C . For each $3 \leq k \leq 7$, there are two extreme cases on each curve: the left-most points describe the algorithm that uses balanced configuration. Here, the memory is as low as possible. The other endpoint gives the best possible time complexity achievable by a change of the target configuration (the right-most points on the plot). Adding more memory will not improve the running time. Table 2 gives explicit values for the time-optimized points.

Table 2. Asymptotic complexities for **arbitrary configurations** when **optimizing for time**. These are the right-most points for each k on the time–memory trade-off curve from Fig. 5.

Tuple size (k)	2	3	4	5	6	7	8
Time	0.4150	0.3789	0.3702	0.3707	0.3716	0.3722	0.3725
Space	0.2075	0.1895	0.1851	0.1853	0.1858	0.1861	0.1862

4 Generalized Locality-Sensitive Filters

In our Algorithm 1, a central sub-problem that arises is that of efficient filtering: given c , \mathbf{x}_1 and a list L_j , find all $\mathbf{x}_j \in L_j$ such that $\langle \mathbf{x}_j, \mathbf{x}_1 \rangle \approx c$, where c is determined by the target configuration. Note that, by using Lemma 3 (Appendix B, full version), we may assume that the points in L_j are uniform from some sphere of dimension $n - o(n)$. To simplify notation, we ignore the $o(n)$ -term, since it

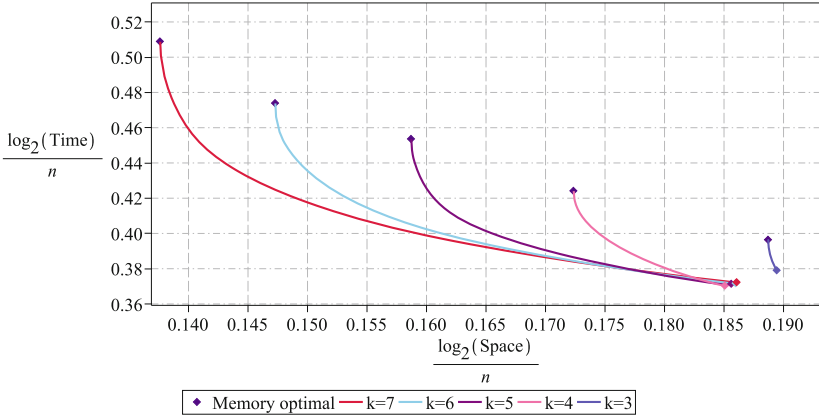


Fig. 5. Time–memory trade-offs for $k = 3, \dots, 7$ and for **arbitrary configurations**. Similar to Fig. 1, the complexities are on a logarithmic scale. The left-most points give complexities for the configuration search in case of balanced configurations. The right-most point indicates the lowest time. Increasing memory even further will not result in improved running times.

will not affect the asymptotics. For notational reasons⁴, we shall assume $c \geq 0$; the case $c \leq 0$ is equivalent by changing \mathbf{x}_1 to $-\mathbf{x}_1$.

So far, we solved the problem of finding all such \mathbf{x}_j in time $|L_j|$. However, better algorithms exist if we preprocess L_j . This leads to the following problem:

Definition 6 (Near neighbor on the sphere). *Let L consist of N points drawn uniformly at random from S^{n-1} , and let $c \geq 0$. The c -near neighbor problem is to preprocess L such that, given a query vector $\mathbf{q} \in S^{n-1}$, one can quickly find points $\mathbf{p} \in L$ with $\langle \mathbf{p}, \mathbf{q} \rangle \geq c$.*

Depending on the magnitude of N there is a distinction between the near neighbor problem for *sparse* data sets ($N = 2^{o(n)}$) and for *dense* data sets ($N = 2^{\Theta(n)}$). In many applications of near neighbor searching one is interested in sparse data sets, and various lower bounds matching upper bounds have been derived for this regime [OWZ14, AIL+15, ALRW17, Chr17]. For our purposes, we are only interested in *dense* data sets.

The target scalar product c corresponds to a target angular distance $\phi = \arccos c$ and we want to find points in a spherical cap centered around \mathbf{q} with angular distance ϕ . With $c \geq 0$, we have $\phi < \frac{1}{2}\pi$, so we are really concerned with near neighbors. In order to simplify incorporating our results for LSF into the configurations framework, we analyze and express everything in terms of scalar products rather than angular distances. This allows us to use Eq. (2) to express the involved complexities.

⁴ The literature is concerned with finding points that are close to each other (corresponding to $c > 0$) rather than points that are far apart (corresponding to $c < 0$). Note that in our applications, we typically would obtain $c < 0$.

Let us denote the data structure resulting from the preprocessing phase by \mathcal{D} . In algorithms for the near neighbor problem, we are interested in time and memory complexities of both preprocessing and of querying and also in the size of \mathcal{D} . Spending more resources on preprocessing and using more space will generally reduce the time complexity of querying.

In our applications, we may want to modify the list L by adding/removing vectors in between queries, without having to completely rebuild all of \mathcal{D} . So we assume that \mathcal{D} is built up by processing the $\mathbf{x} \in L$ one element at a time and updating \mathcal{D} . We therefore analyze the preprocessing cost in terms of the cost to update \mathcal{D} when L changes by one element.

Spherical locality-sensitive filters. To solve the near neighbor problem on the sphere, [BDGL16] introduced spherical locality-sensitive filters, inspired by e.g. the spherical cap LSH of [AINR14]. The idea is to create a data structure \mathcal{D} of many *filter buckets*, where each bucket $B_{\mathbf{u}}$ contains all vectors from L which are α -close to a filter vector \mathbf{u} , where \mathbf{u} is typically not from L , but drawn uniformly at random from S^{n-1} . Here, two vectors \mathbf{u}, \mathbf{p} are considered α -close iff $\langle \mathbf{u}, \mathbf{p} \rangle \geq \alpha$. Let F be the set of all such chosen \mathbf{u} 's. Ideally, one generates $|F| \gg 1$ of these buckets, each with $\mathbf{u} \in F$ chosen independently and uniformly at random from S^{n-1} . We build up the data structure by processing the elements $\mathbf{x} \in L$ one by one and updating \mathcal{D} each time, as mentioned above. This means that for each update, we need to find all $\mathbf{u} \in F$ that are α -close to a given \mathbf{x} .

If we then want to find points $\mathbf{x} \in L$ that are c -close to a given query point \mathbf{q} , we first find all $\mathbf{u} \in F$ that are β -close to \mathbf{x} for some β . Then we search among those buckets $B_{\mathbf{u}}$ for points $\mathbf{x} \in B_{\mathbf{u}}$ that are c -close to \mathbf{x} . The idea here is that points in $B_{\mathbf{u}}$ for \mathbf{u} close to \mathbf{q} have a higher chance to be close to \mathbf{q} than a random point from L . The algorithm is detailed in Algorithm 2.

Structured filters. In the above idea, one has to find all $\mathbf{u} \in F$ that are close to a given point \mathbf{x} . A naive implementation of this would incur a cost of $|F|$, which would lead to an impractically large overhead. To surmount this problem, a small amount of *structure* is added to the filter vectors \mathbf{u} , making them dependent: small enough so that their joint distribution is sufficiently close to $|F|$ independent random vectors, but large enough to ensure that finding the filter vectors that are close to \mathbf{x} can be done in time (up to lower-order terms) proportional to the number of such close filters vectors. This is the best one can hope for. This technique was later called “tensoring” in [Chr17], and replaced with a tree-based data structure in [ALRW17]. For further details regarding this technique we refer the reader to [BDGL16]; below, we will simply assume that filter vectors are essentially independent, and that we can solve the problem of finding all $\mathbf{u} \in F$ close to given point with a time complexity given by the number of solutions.

Complexity. Let us now analyze the complexity of our spherical locality-sensitive filters and set the parameters α and β . For this we have the following theorem:

Algorithm 2. Algorithm for spherical locality-sensitive filteringParameters: α, β, c, F \mathcal{D} and L are kept as global state, modified by functions below. $\mathcal{D} = \{B_u \mid u \in F\}$.INSERT and REMOVE modify L and \mathcal{D} . We assume that \mathcal{D} and L are constructed by calling INSERT repeatedly. Initially, all B_u and L are empty.Query(q) outputs $x \in L$ that are c -close to q .

```

1: function INSERT( $x$ )                                ▷ Add  $x$  to  $L$  and update  $\mathcal{D}$ 
2:    $L \leftarrow L \cup \{x\}$ 
3:   for all  $u \in F$  s.t.  $\langle u, x \rangle \geq \alpha$  do
4:      $B_u \leftarrow B_u \cup \{x\}$ 

1: function REMOVE( $x$ )                                ▷ Remove  $x$  from  $L$  and update  $\mathcal{D}$ 
2:    $L \leftarrow L \setminus \{x\}$ 
3:   for all  $u \in F$  s.t.  $\langle u, x \rangle \geq \alpha$  do
4:      $B_u \leftarrow B_u \setminus \{x\}$ 

1: function QUERY( $q$ )                                  ▷ Find  $x \in L$  with  $\langle x, q \rangle \geq c$ 
2:   PointsFound  $\leftarrow \emptyset$ 
3:   for all  $u \in F$  s.t.  $\langle u, q \rangle \geq \beta$  do
4:     for all  $x \in B_u$  do
5:       if  $\langle x, q \rangle \geq c$  then
6:         PointsFound  $\leftarrow$  PointsFound  $\cup \{x\}$ 
7:   return PointsFound

```

Theorem 3 (Near neighbor trade-offs). Consider Algorithm 2 for fixed target scalar product $0 \leq c \leq 1$, fixed $0 \leq \alpha, \beta \leq 1$ and let L be a list of i.i.d. uniform points from S^{n-1} , with $|L|$ exponential in n . Then any pareto-optimal⁵ parameters satisfy the following restrictions:

$$\begin{aligned}
 |L| (1 - \alpha^2)^{n/2} &= 1 \\
 \alpha c &\leq \beta \leq \min\left\{\frac{\alpha}{c}, \frac{c}{\alpha}\right\}
 \end{aligned} \tag{9}$$

Assume these restrictions hold and that $|F|$ is chosen as small as possible while still being large enough to guarantee that on a given query, we find all c -close points except with superexponentially small error probability. Then the complexity of spherical locality-sensitive filtering is, up to subexponential factors, given by:

- **Bucket size:** The expected size of each bucket is 1.
- **Number of buckets:** The number of filter buckets is

$$|F| = \frac{(1 - c^2)^{n/2}}{(1 + 2c\alpha\beta - c^2 - \alpha^2 - \beta^2)^{n/2}}.$$

- **Update time:** For each $x \in L$, updating the data structure \mathcal{D} costs time $T_{\text{Update}} = |F| \cdot (1 - \alpha^2)^{n/2}$

⁵ This means that we cannot modify the parameters α, β in a way that would reduce either the preprocessing or the query cost without making the other cost larger.

- **Preprocessing time:** The total time to build \mathcal{D} is $T_{\text{Preprocess}} = |F|$.
- **Memory used:** The total memory required is $|F|$, used to store \mathcal{D} .
- **Query time:** After \mathcal{D} has been constructed, each query takes time $T_{\text{Query}} = |F| \cdot (1 - \beta^2)^{n/2}$.

Note that the formulas for the complexity in the theorem rely on Eqs. (9) to hold. As the proof of this theorem (mainly the conditions for pareto-optimality) is very technical, we defer it to Appendix D (see full version [HKL]). In the following Remark 1, we only discuss the meaning of the restrictions that are satisfied in the pareto-optimal case and sketch how to derive the formulas. After that, we discuss what the extreme cases for β mean in terms of time/memory trade-offs.

Remark 1. Using Theorem 1, for a given point \mathbf{u} and uniformly random $\mathbf{x} \in \mathbb{S}^{n-1}$, the probability that $\langle \mathbf{u}, \mathbf{x} \rangle \approx \alpha$ is given by $(1 - \alpha^2)^{n/2}$, ignoring subexponential factors throughout the discussion. So the expected size of the filter buckets is $|L|(1 - \alpha^2)^{n/2}$ and the first condition ensures that this size is 1.

If the expected bucket size was exponentially small, we would get a lot of empty buckets. Since F needs to have a structure that allows to find all $\mathbf{u} \in F$ close to a given point quickly, we cannot easily remove those empty buckets. Consequently, the per-query cost would be dominated by looking at (useless) empty buckets. It is strictly better (in both time and memory) to use fewer, but more full buckets in that case. Conversely, if the buckets are exponentially large, we should rather use more, but smaller buckets, making \mathcal{D} more fine-grained.

Now consider a “solution triple” $(\mathbf{x}, \mathbf{q}, \mathbf{u})$, where \mathbf{q} is a query point, $\mathbf{x} \in L$ is a solution for this query and $\mathbf{u} \in F$ is the center of the filter bucket used to find the solution. By definition, this means $\langle \mathbf{x}, \mathbf{q} \rangle \geq c$, $\langle \mathbf{x}, \mathbf{u} \rangle \geq \alpha$ and $\langle \mathbf{q}, \mathbf{u} \rangle \geq \beta$. The second set of conditions from Eq. (9) imply that these 3 inequalities are actually satisfied with (near-)equality whp. Geometrically, it means that the angular triangle formed by a triple $\mathbf{q}, \mathbf{x}, \mathbf{u}$ in a 2-dimensional \mathbb{S}^2 with these pairwise inner products has no obtuse angles.

The required size of $|F|$ is determined by the conditional probability $P = \frac{1}{|F|}$ that a triple $(\mathbf{x}, \mathbf{u}, \mathbf{q})$ has these pairwise scalar products, conditioned on $\langle \mathbf{x}, \mathbf{q} \rangle \approx c$. Using Theorem 1, this evaluates to

$$P = \Pr_{\mathbf{u} \in \mathbb{S}^{n-1}} [\langle \mathbf{x}, \mathbf{u} \rangle \approx \alpha, \langle \mathbf{q}, \mathbf{u} \rangle \approx \beta \mid \langle \mathbf{x}, \mathbf{q} \rangle \approx c] = \frac{\left(\det \begin{pmatrix} 1 & \alpha & \beta \\ \alpha & 1 & c \\ \beta & c & 1 \end{pmatrix}\right)^{n/2}}{\left(\det \begin{pmatrix} 1 & c \\ c & 1 \end{pmatrix}\right)^{n/2}}.$$

Taking the inverse gives $|F|$. The other formulas are obtained even more easily by applying Theorem 1: the probability that a point $x \in L$ is to be included in a bucket $B_{\mathbf{u}}$ is $(1 - \alpha^2)^{n/2}$, giving the update cost (using the structure of F). Similarly for the per-query cost, we look at $|F|(1 - \beta^2)^{n/2}$ buckets with $|L|(1 - \alpha^2)^{n/2}$ elements each. Using $|L|(1 - \alpha^2) = 1$ then gives all the formulas.

Note that Theorem 3 offers some trade-off. With $|L|$ and c given, the parameter α is determined by the condition $(1 - \alpha^2)^{n/2} |L| = 1$. The complexities can then be expressed via β , which we can choose between $c \cdot \alpha$ and either α/c or c/α .

For $\beta = c \cdot \alpha$, we have (up to subexponential terms) $|F| = \frac{1}{(1-\alpha^2)^{n/2}} = |L|$, so the update cost is subexponential. The memory complexity is only $\tilde{O}(|L|)$, which is clearly minimal. The query complexity is at $\tilde{O}(|L| (1 - c^2 \alpha^2)^{n/2})$. Increasing β increases both the time complexity for updates and the memory complexity for \mathcal{D} , whereas the query complexity decreases. If $\alpha \leq c$, we can increase β up to $\beta = \frac{c}{\alpha}$. At that point, we have subexponential expected query complexity. The total complexity of preprocessing is given by $\frac{1}{(1-\alpha^2/c^2)^{n/2}}$.

If $\alpha \geq c$, we may increase β up to $\frac{c}{\alpha}$. At that point, we obtain a query complexity of $|L| (1 - c^2)^{n/2}$. This is equal to the number of expected solutions to a query, hence the query complexity is optimal. The preprocessing cost is equal to $|L| (1 - c^2)^{n/2} (1 - \alpha^2/c^2)^{-n/2}$. Increasing β further will only make both the query and preprocessing costs worse. Note that, if the total number of queries is less than $\frac{1}{(1-\beta_{\max}^2)^{n/2}}$ with $\beta_{\max} = \min\{\frac{\alpha}{c}, \frac{c}{\alpha}\}$, it does not make sense to increase β up to β_{\max} even if we have arbitrary memory, as the total time of preprocessing will exceed the total time of all queries. In the special case where the number of queries $|Q|$ is equal to the list size $|L|$, preprocessing and total query costs are equal for $\alpha = \beta$. This latter choice corresponds to the case used in [BDGL16].

In the LSH literature, the quality of a locality sensitive hash function is usually measured in terms of update and query exponents ρ_u resp. ρ_q . Re-phrasing our results in these quantities, we obtain the following corollary:

Corollary 2. *Let $c > 0$, corresponding to an angular distance $\phi = \arccos c$, $0 < \phi < \frac{1}{2}\pi$ and consider the c -near neighbor problem on the n -dimensional unit sphere for dense data sets of size $N = 2^{\Theta(n)}$. Then, for any value of γ with $c \leq \gamma \leq \min\{\frac{1}{c}, \frac{c}{1-N^{-2/n}}\}$, spherical locality sensitive filtering solves this problem with update and query exponents given by:*

$$\rho_u = \log\left(\frac{\sin^2 \phi}{\sin^2 \phi - (1 - N^{-2/n})(1 - 2\gamma \cos \phi + \gamma^2)}\right) / \log(N^{2/n}) - 1$$

$$\rho_q = \log\left(\frac{(1 - \gamma^2 + \gamma^2 N^{-2/n}) \sin^2 \phi}{\sin^2 \phi - (1 - N^{-2/n})(1 - 2\gamma \cos \phi + \gamma^2)}\right) / \log(N^{2/n})$$

The data structure requires $N^{1+\rho_u+o(1)}$ memory, can be initialized in time $N^{1+\rho_u+o(1)}$ and allows for updates in time $N^{\rho_u+o(1)}$. Queries can be answered in time $N^{\rho_q+o(1)}$.

Proof. This is just a restatement of Theorem 3 with $\gamma := \beta/\alpha$, plugging in $\alpha^2 = 1 - N^{-2/n}$ and $\cos \phi = c$.

4.1 Application to the Configuration Problem

We now use spherical locality-sensitive filtering as a subroutine of Algorithm 1, separately replacing each naive $\text{FILTER}_{i,j}$ subroutine of Algorithm 1 resp. Fig. 4

by spherical locality-sensitive filtering. In this application, in each near neighbor problem the number of queries is equal to the list size. We caution that in Algorithm 1, the lists for which we have to solve a c -near neighbor problem were obtained from the initial lists by restricting scalar products with known points. This changes the distribution, which renders our results from Theorem 3 not directly applicable. In order to obtain a uniform distribution on a sphere, we use TRANSFORM in Algorithm 1, justified by Lemma 3 (Appendix B in the full version [HKL]), to perform an affine transformation on all our point. This transformation affects scalar products and, as a consequence, the parameter c in each near neighbor problem is not directly given by $\pm C_{i,j}$.

For the case of the k -sieve with the balanced configuration $C_{i,j} = -\frac{1}{k}$ for $i \neq j$, the parameter c used in the replacement of FILTER $_{i,j}$ is given by $\frac{1}{k+1-i}$, as a simple induction using Lemma 3 (full version) shows.

Using LSF changes the time complexities as follows: recall the time complexity is determined by the cost to create various sublists $L_j^{(i)}$ as in Sect. 3.1, where $L_j^{(i)}$ is obtained from the input list L_j by applying i filterings. These filterings and $L_j^{(i)}$ depend on the partial solution $\mathbf{x}_1, \dots, \mathbf{x}_i$. The cost $T_{i,j}$ to create all instances (over all choices of $\mathbf{x}_1, \dots, \mathbf{x}_i$) of $L_j^{(i)}$ is then given by (cf. Eq. (3))

$$T_{i,j} = \prod_{r=1}^{i-1} |L_r^{(r-1)}| \cdot \left(T_{\text{Preprocess},i,j} + |L_i^{(i-1)}| \cdot T_{\text{Query},i,j} \right),$$

where $T_{\text{Preprocess},i,j}$ and $T_{\text{Query},i,j}$ denote the time complexities for LSF when replacing FILTER $_{i,j}$. Here, $\mathbf{x}_i \in L_i^{(i-1)}$ takes the role of the query point.

Applying LSF to this configuration gives time/memory trade-offs depicted in Fig. 6. Optimizing for time yields Table 4 and for memory yields Table 3. Note that for $k \geq 5$, there is no real trade-off. The reason for this is as follows: for large k and balanced configuration, the running time is dominated by the creation of sublists $L_j^{(i)}$ with large i . At these levels, the list sizes $L_j^{(i)}$ have already shrunk considerably compared to the input list sizes. The memory required even for time-optimal LSF at these levels will be below the size of the input lists. Consequently, increasing the memory complexity will not help. We emphasize that for any $k > 2$, the memory-optimized algorithm has the same memory complexity as [HK17], but strictly better time complexity.

4.2 Comparison with Configuration Extension

There exists a more memory-efficient variant [BGJ14] of the LSF techniques than described above, which can reach optimal time without increasing the memory complexity. Roughly speaking, the idea is to immediately process the filter buckets after creating them and never storing them all. However, even ignoring subexponential overhead, this variant has two limitations: Firstly, we need to know all query points $\mathbf{q} \in Q$ in advance. This makes it unsuitable for the top level (which is the only level for $k = 2$) in the Gauss Sieve, because we build up the list L vector by vector. Secondly, in this variant we obtain the solutions

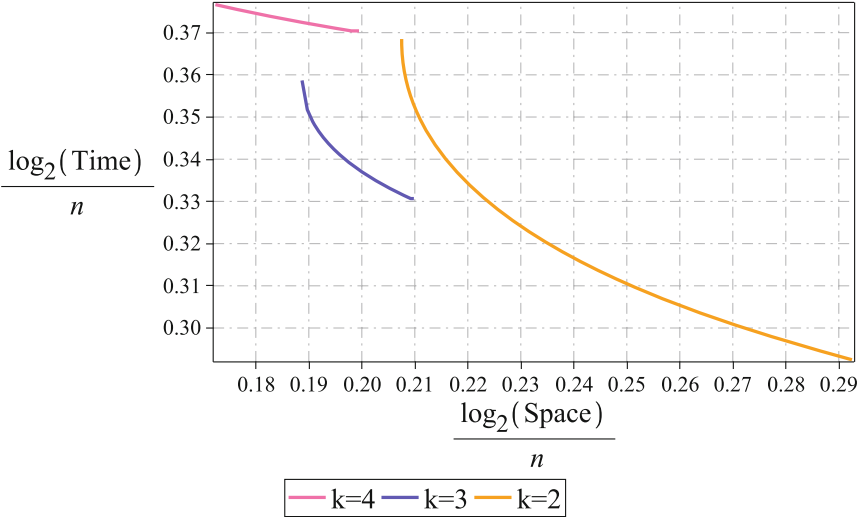


Fig. 6. Trade-offs for $k = 2, 3, 4$. Analogous to Figure 1, the axes are on a logarithmic scale. The exact values for the left-most points for each curve (optimized for memory) are given in Table 3. The right-most points (optimized for time) are given in Table 4.

Table 3. Asymptotic complexities when using **LSF** with the **balanced/any configuration**, when **optimizing for memory**: we keep the memory equal to the input list sizes. The running time for $k = 2$ with LSF is equal to the one obtained in [BDGL16]. Without LSF, the runtime exponent 0.4150 for $k = 2$ was first shown in [NV08]. Note that, when optimizing for memory, we implicitly restrict to the balanced configuration, because this minimizes the input list size.

Tuple size (k)	2	3	4	5	6	7	8
Time [HK17]	0.415	0.3962	0.4240	0.4534	0.4738	0.5088	0.5398
Time (with LSF)	0.3685	0.3588	0.3766	0.4159	0.4497	0.4834	0.5205
Space	0.2075	0.1887	0.1723	0.1587	0.1473	0.1376	0.1293

Table 4. Asymptotic complexities when using **LSF** with the **balanced configuration**, when **optimizing for time** (i.e. using the largest amount of memory that still leads to better time complexities). With this we can obtain improved time complexities for our k -list algorithm for $k = 2, 3, 4$. Starting from $k = 5$, giving more memory to LSF without changing the target configuration does not result in an asymptotically faster algorithm.

Tuple size (k)	2	3	4	5	6	7	8
Time (with LSF)	0.3685	0.3307	0.3707	0.4159	0.4497	0.4834	0.5205
Space	0.2075	0.2092	0.1980	0.1587	0.1473	0.1376	0.1293

(\mathbf{q}, \mathbf{x}) with $\mathbf{q} \in Q$, $\langle \mathbf{q}, \mathbf{x} \rangle \approx c$ in essentially random order. In particular, to obtain all solutions \mathbf{x} for *one* given query point \mathbf{q} , there is some overhead⁶. Note that, due to the structure of Algorithm 1, we really need the solutions for one query point after another (except possibly at the last level). So the memory-less variant does not seem well-suited for our algorithm for $k > 2$.

A generalization of memory-efficient LSF to $k > 2$ was described in [HK17] under the name configuration extension and applied to sieving. However, due to the second limitation described above, they achieve smaller speed-ups than we do for $k > 2$.

5 Combining both Techniques

In the previous sections we showed how to obtain time–memory trade-offs by either (1) changing the target configuration C (Sect. 3), or (2) using locality-sensitive filters for the balanced configuration (Sect. 4). An obvious next step would be to try to combine both techniques to obtain even better results, i.e. by solving the configuration problem for an *arbitrary* target configuration C using LSF as a subroutine. The memory complexity will be either dominated by the input list sizes (which are determined by C from the condition that the output list size equals the input list sizes) or by the filter buckets used for LSF.

To obtain time–space trade-offs for sieving, we optimize over all possible configurations C with $\sum_{i,j} C_{i,j} \leq 1$ and parameters $\beta_{i,j}$ used in each application of LSF. We used MAPLE [M] for the optimization. To obtain the complete trade-off curve, we optimized for time while prescribing a bound on memory, and varying this memory bound to obtain points on the curve. Note that the memory-optimized points are the same as in Sect. 4.1, since we have to keep C balanced – in a memory-restricted regime, LSF only gives the same improvement as combining LSF with arbitrary target configurations. However, as soon as the memory bound is slightly larger, we already observe that this combination of both techniques leads to results strictly better than ones produced by using neither or only one of the two techniques.

The resulting trade-off curves are depicted in Fig. 1 for $k = 2, 3, 4$. The same curves for $k = 3, 4$ are also depicted in Fig. 7, where the trade-offs are compared to using either a modified configuration (without LSF) or LSF (with a balanced configuration). The time-optimal points for fixed k are given in Table 1.

In general, for small k we can gain a lot by using LSF, whereas for large k , most of the improvement comes from changing the target configuration. Note also that the trade-offs obtained by combining both techniques are strictly superior to using only either improvement.

⁶ E.g. we could first output the solutions for *all* query points and sort these solutions wrt. the query point, but that requires storing the set of all solutions for all queries and increases the memory complexity.

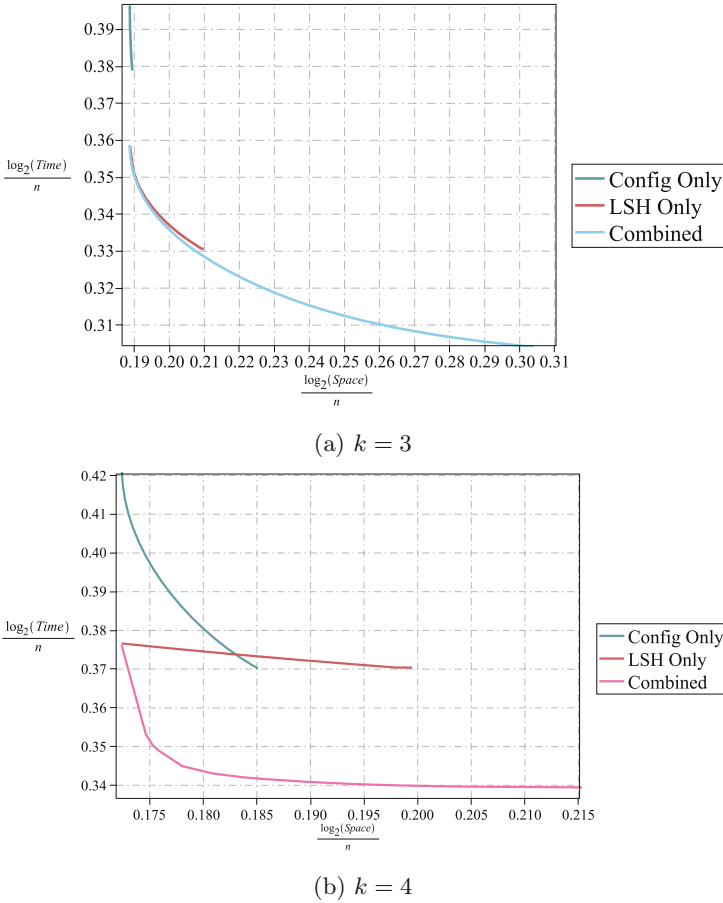


Fig. 7. Trade-offs for $k = 3, 4$ with different improvements: changing the target configuration, LSF, or both.

6 Tuple Gauss Sieve

So far we have been concerned with the algorithm for the configuration problem and how it can be used to solve the approximate k -list problem. Here we explain how our algorithm for the k -list problem can be used as a subroutine within k -tuple lattice sieving. We only give a high-level overview here. A more detailed description, including pseudo-code can be found in Appendix C in [HKL].

Lattice sieving algorithms have two flavors: the Nguyen-Vidick sieve [NV08] and the Gauss sieve [MV10]. Since in practice the latter outperforms the former, we concentrate on Gauss sieve. However, our approach can be used for both.

The algorithm receives on input a lattice represented by a basis $B \in \mathbb{R}^{n \times n}$ and outputs a vector $\mathbf{v} \in \mathcal{L}(B)$ s.t. $\|\mathbf{v}\| = \lambda_1(\mathcal{L}(B))$. During its run, the Gauss

Sieve needs to efficiently sample (typically long) points $\mathbf{v} \in \mathcal{L}(B)$, whose direction is nearly uniformly distributed.

After preprocessing the basis by an L^3 reduction, we can use Klein’s sampling procedure [Kle00], which outputs vectors of length $2^n \cdot \lambda_1(\mathcal{L}(B))$ in $\text{poly}(n)$ time.

In the Gauss Sieve, we keep a set S of vectors and try to perform as many k -reductions on S as possible. By this, we mean that we look for k -tuples $(\mathbf{x}_1, \dots, \mathbf{x}_k) \in S$ with $0 < \|\mathbf{x}_1 \pm \dots \pm \mathbf{x}_k\| < \max\{\|\mathbf{x}_1\|, \dots, \|\mathbf{x}_k\|\}$ and we replace the vector of maximal length by the (shorter) sum. To find such k -tuples, we use Algorithm 1 for the configuration problem for an appropriately chosen configuration C . If no k -tuples from S satisfy C , we enlarge S by sampling a new vector.

To avoid checking whether the same k -tuples satisfies C multiple times, we separate $S = L \cup Q$ into a list L and a queue Q . The list L contains the lattice vectors that we already checked: we maintain that no k -tuples from L satisfy C . The queue Q contains vectors that might still be part of a k -tuple satisfying C .

Due to our splitting of S , we may assume that one of the vectors in the k -tuples is from Q . In fact, we can just repeatedly call Algorithm 1 on lists $L_1 = \{\mathbf{p}\}$, $L_2 = \dots = L_k = L$ for $\mathbf{p} \in Q$ and perform k -reductions on the solutions.

Whenever we sample or modify a vector, we have to move it into Q ; if no more reduction is possible with $L_1 = \{\mathbf{p}\}$, $L_2 = \dots = L_k$, we move \mathbf{p} from Q into L . If Q is empty, this signals that we have to sample a new vector.

Since the length of the vectors in S keeps decreasing, we hope to eventually find the shortest vector. We stop the search when the length of the shortest element of the list (i.e., the shortest lattice vector found) is equal to the first successive minimum, $\lambda_1(B)$. Since we usually do not know the value of $\lambda_1(B)$ exactly, we use some heuristics: in practice [MLB15], we stop once we found a lot of k -tuples where the k -reduction would give a zero vector.

6.1 Gauss Sieve with $k = 3$ in Practice

We ran experiments with the 3-Gauss sieve algorithm with the aim to compare balanced and non-balanced configuration search. We used a Gauss-sieve implementation developed by Bai et al. in [BLS16] and by Herold and Kirshanova in [HK17].

As an example, we generated 5 random (in the sense of Goldstein-Mayer [GM06]) 70-dimensional lattices and preprocessed them with β -BKZreduction for $\beta = 10$. Our conclusions are the following:

- The unbalanced configuration C given above indeed puts more work to the (asymptotically) non-dominant first level than the balanced configuration does. We counted the number of inner products (applications of filterings) on the first and on the second levels for each of the 5 lattices. As the algorithm spends most of its time computing inner-products, we think this model reasonably reflects the running time.

Table 5. Running times of triple lattice sieve for 3 different target configurations $C_{i,j}$. All the figures are average values over 5 different Goldstein-Mayer lattices of dimension n . Columns ‘level 1’ show the number of inner-products computed on the first level of the algorithm (it corresponds to the quantity $|L|^2$), columns ‘level 2’ count the number of inner-product computed in the second level (in corresponds to $|L| \cdot |L^{(1)}|^2$). The third columns T shows actual running times in seconds.

n	$ C_{i,j} = 0.32$			$ C_{i,j} = 0.333$			$ C_{i,j} = 0.3508$		
	level 1	level 2	T/sec	level 1	level 2	T/sec	level 1	level 2	T/sec
60	$4.8 \cdot 10^8$	$1.7 \cdot 10^8$	$5.6 \cdot 10^2$	$5.8 \cdot 10^8$	$1.2 \cdot 10^8$	$6.0 \cdot 10^2$	$7.2 \cdot 10^8$	$2.1 \cdot 10^8$	$7.3 \cdot 10^2$
62	$9.4 \cdot 10^8$	$3.5 \cdot 10^8$	$1.3 \cdot 10^3$	$1.1 \cdot 10^9$	$6.7 \cdot 10^8$	$2.7 \cdot 10^3$	$1.4 \cdot 10^9$	$1.4 \cdot 10^8$	$3.4 \cdot 10^3$
64	$1.7 \cdot 10^9$	$6.8 \cdot 10^8$	$3.4 \cdot 10^3$	$2.1 \cdot 10^9$	$4.6 \cdot 10^8$	$5.0 \cdot 10^3$	$2.7 \cdot 10^9$	$2.7 \cdot 10^8$	$5.6 \cdot 10^3$
66	$3.0 \cdot 10^9$	$1.5 \cdot 10^9$	$5.2 \cdot 10^3$	$3.9 \cdot 10^9$	$8.7 \cdot 10^8$	$2.1 \cdot 10^4$	$5.1 \cdot 10^9$	$5.1 \cdot 10^8$	$1.5 \cdot 10^4$
68	$6.0 \cdot 10^9$	$2.5 \cdot 10^9$	$1.4 \cdot 10^4$	$7.3 \cdot 10^9$	$1.6 \cdot 10^9$	$1.1 \cdot 10^4$	$9.6 \cdot 10^9$	$9.5 \cdot 10^8$	$2.7 \cdot 10^4$
70	$1.1 \cdot 10^{10}$	$4.9 \cdot 10^9$	$3.0 \cdot 10^4$	$1.4 \cdot 10^{10}$	$3.2 \cdot 10^9$	$3.6 \cdot 10^4$	$1.8 \cdot 10^{10}$	$1.8 \cdot 10^9$	$4.9 \cdot 10^4$
72	$2.1 \cdot 10^{10}$	$9.4 \cdot 10^9$	$4.8 \cdot 10^4$	$2.6 \cdot 10^{10}$	$6.1 \cdot 10^9$	$7.2 \cdot 10^4$	–	–	–
74	$3.9 \cdot 10^{10}$	$1.8 \cdot 10^{10}$	$1.3 \cdot 10^5$	$4.7 \cdot 10^{10}$	$1.1 \cdot 10^{10}$	$1.4 \cdot 10^5$	$6.4 \cdot 10^{10}$	$6.2 \cdot 10^9$	$1.9 \cdot 10^5$
76	$7.0 \cdot 10^{10}$	$3.4 \cdot 10^{10}$	$2.7 \cdot 10^5$	$8.6 \cdot 10^{10}$	$2.1 \cdot 10^{10}$	$3.1 \cdot 10^5$	–	–	–

The average number of inner-product computations performed on the upper-level increases from $1.39 \cdot 10^{10}$ (balanced case) to $1.84 \cdot 10^{10}$ (unbalanced case), while on the lower (asymptotically dominant) level this number drops down from $3.23 \cdot 10^9$ to $1.82 \cdot 10^9$.

- As for the actual running times (when measured in seconds), 3-Gauss sieve instantiated with balanced configuration search outperforms the sieve with the unbalanced C given above on the dimensions up to 74. We explain this by the fact that the asymptotical behavior is simply not visible on such small dimensions. In fact, in our experiments the dominant level turns out to be the *upper* one (just compare the number of the inner-products computations in Table 5). So in practice one might want to reverse the balancing: put more more work on the lower level than on the upper by again, changing the target configuration. Indeed, if we relax the inner-product constraint to 0.32 (in absolute value), we gain a significant speed-up compared with balanced 1/3 and asymptotically optimal 0.3508.

Acknowledgements. We thank the anonymous reviewers of PKC 2018 for careful proof-reading and finding an error in one of our tables. Gottfried Herold and Elena Kirshanova are supported by ERC Starting Grant ERC-2013-StG-335086-LATTAC. Thijs Laarhoven is supported by ERC consolidator grant 617951.

References

- [ADPS16] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. In: USENIX Security Symposium, pp. 327–343 (2016)
- [ADRS15] Aggarwal, D., Dadush, D., Regev, O., Stephens-Davidowitz, N.: Solving the shortest vector problem in 2^n time via discrete Gaussian sampling. In: STOC, pp. 733–742 (2015)
- [AEVZ02] Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. *IEEE Trans. Inf. Theor.* **48**(8), 2201–2214 (2002)
- [AIL+15] Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal LSH for angular distance. In: NIPS, pp. 1225–1233 (2015)
- [AINR14] Andoni, A., Indyk, P., Nguyễn, H., Razenshteyn, I.: Beyond locality-sensitive hashing. In: SODA, pp. 1018–1028 (2014)
- [AKS01] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610 (2001)
- [ALRW17] Andoni, A., Laarhoven, T., Razenshteyn, I., Waingarten, E.: Optimal hashing-based time-space trade-offs for approximate near neighbors. In: SODA, pp. 47–66 (2017)
- [AN17] Aono, Y., Nguyen, P.Q.: Random sampling revisited: lattice enumeration with discrete pruning. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 65–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_3
- [BDGL16] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: SODA, pp. 10–24 (2016)
- [BGJ14] Becker, A., Gama, N., Joux, A.: A sieve algorithm based on overlattices. In: ANTS, pp. 49–70 (2014)
- [BL16] Becker, A., Laarhoven, T.: Efficient (ideal) lattice sieving using cross-polytope LSH. In: AFRICACRYPT, pp. 3–23 (2016)
- [BLS16] Bai, S., Laarhoven, T., Stehlé, D.: Tuple lattice sieving. In: ANTS, pp. 146–162 (2016)
- [Chr17] Christiani, T.: A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In: SODA, pp. 31–46 (2017)
- [CN11] Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_1
- [FP85] Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice. *Math. Comput.* **44**(170), 463–471 (1985)
- [GM06] Goldstein, D., Mayer, A.: On the equidistribution of hecke points. *Forum Math.* **15**(3), 165–189 (2006)
- [GNR10] Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_13
- [HK17] Herold, G., Kirshanova, E.: Improved algorithms for the approximate k -list problem in Euclidean norm. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 16–40. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54365-8_2

- [HKL] Herold, G., Kirshanova, E., Laarhoven, T.: Speed-ups and time-memory trade-offs for tuple lattice sieving (2017). eprint.iacr.org/2017/1228
- [Kan83] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: STOC, pp. 193–206 (1983)
- [Kle00] Klein, P.: Finding the closest lattice vector when it’s unusually close. In: SODA, pp. 937–941 (2000)
- [Laa15a] Laarhoven, T.: Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 3–22. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_1
- [Laa15b] Laarhoven, T.: Tradeoffs for nearest neighbors on the sphere. arXiv, pp. 1–16 (2015)
- [Laa16] Laarhoven, T.: Finding closest lattice vectors using approximate Voronoi cells. Cryptology ePrint Archive, Report 2016/888 (2016)
- [LdW15] Laarhoven, T., de Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 101–118. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22174-8_6
- [LMvdP15] Laarhoven, T., Mosca, M., van de Pol, J.: Finding shortest lattice vectors faster using quantum search. Des. Codes Crypt. **77**(2), 375–400 (2015)
- [LRBN] Lindner, R., Rückert, M., Baumann, P., Nobach, L.: SVP challenge generator. <https://latticechallenge.org/svp-challenge>
- [M] Standard Worksheet Interface, Maple 2016.0, build id 1113130, 17 February 2016
- [MLB15] Mariano, A., Laarhoven, T., Bischof, C.: Parallel (probable) lock-free Hash-Sieve: a practical sieving algorithm for the SVP. In: ICPP, pp. 590–599 (2015)
- [MV10] Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, pp. 1468–1480 (2010)
- [NV08] Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. Math. Cryptol. **2**, 181–207 (2008)
- [OWZ14] O’Donnell, R., Yi, W., Zhou, Y.: Optimal lower bounds for locality-sensitive hashing (except when q is tiny). ACM Trans. Comput. Theor. **6**(1), 5:1–5:13 (2014)
- [PS09] Pujol, X., Stehlé, D.: Solving the shortest lattice vector problem in time $2^{2.465n}$. Cryptology ePrint Archive, Report 2009/605, pp. 1–7 (2009)
- [Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)
- [Sch87] Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. **53**(2–3), 201–224 (1987)
- [SE94] Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**(2–3), 181–199 (1994)
- [WLTB11] Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In: ASIACCS, pp. 1–9 (2011)
- [ZPH13] Zhang, F., Pan, Y., Hu, G.: A three-level sieve algorithm for the shortest vector problem. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 29–47. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_2



Fast Lattice Basis Reduction Suitable for Massive Parallelization and Its Application to the Shortest Vector Problem

Tadanori Teruya^{1(✉)}, Kenji Kashiwabara^{1,2}, and Goichiro Hanaoka¹

¹ Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology,
AIST Tokyo Waterfront Bio-IT Research Building, 2-4-7 Aomi,
Koto-ku, Tokyo, Japan
tadanori.teruya@aist.go.jp

² Department of General Systems Studies, University of Tokyo, 3-8-1 Komaba,
Meguro-ku, Tokyo, Japan

Abstract. The hardness of the shortest vector problem for lattices is a fundamental assumption underpinning the security of many lattice-based cryptosystems, and therefore, it is important to evaluate its difficulty. Here, recent advances in studying the hardness of problems in large-scale lattice computing have pointed to need to study the design and methodology for exploiting the performance of massive parallel computing environments. In this paper, we propose a lattice basis reduction algorithm suitable for massive parallelization. Our parallelization strategy is an extension of the Fukase–Kashiwabara algorithm (J. Information Processing, Vol. 23, No. 1, 2015). In our algorithm, given a lattice basis as input, variants of the lattice basis are generated, and then each process reduces its lattice basis; at this time, the processes cooperate and share auxiliary information with each other to accelerate lattice basis reduction. In addition, we propose a new strategy based on our evaluation function of a lattice basis in order to decrease the sum of squared lengths of orthogonal basis vectors. We applied our algorithm to problem instances from the SVP Challenge. We solved a 150-dimension problem instance in about 394 days by using large clusters, and we also solved problem instances of dimensions 134, 138, 140, 142, 144, 146, and 148. Since the previous world record is the problem of dimension 132, these results demonstrate the effectiveness of our proposal.

Keywords: Lattice basis reduction · Parallelization
Shortest vector problem · SVP Challenge

1 Introduction

1.1 Background

Lattice-based cryptography is an attractive field of research, because it has produced useful cryptographic schemes, for example, public-key cryptosystems,

functional encryption schemes, and fully homomorphic encryption schemes, etc. [1, 2, 4, 7–12, 24, 26–29, 31, 44], and these systems are believed to resist quantum cryptanalysis. Their security has been proven under the computational hardness assumption, and this assumption is related to the difficulty in solving the shortest vector problem (SVP) on a lattice. Therefore, it is important to show evidence of and to estimate the hardness not only theoretically, but also practically.

There have been many studies reporting on algorithms to solve the SVP, and the typical approaches are lattice basis reduction and enumerating lattice vectors. Lattice basis reduction algorithms take a lattice basis as input and output another lattice basis for the same lattice whose basis vectors are relatively shorter than the input. Enumeration algorithms take a lattice basis as input and search for relatively shorter lattice vectors than the basis vectors of the input. To evaluate the computational hardness of SVP, it is important to improve the efficiency of algorithms and implement high-performance solvers; here too, there have been many studies [6, 13–17, 20, 21, 23, 30, 33, 34, 36–39, 46–49]. However, thanks to recent advances in computing technologies, exploiting the performance of massively parallel computing environment, for example, a cluster consists of many large computers, has become the most important strategy with which to evaluate the security of lattice cryptosystems, and a methodology to solve the SVP is needed for it.

1.2 Our Contribution

In this paper, we propose a new lattice basis reduction algorithm suitable for parallelization. Our proposal consists of a parallelization strategy and a reduction strategy.

Our parallelization strategy enables many parallel processes to reduce the lattice basis cooperatively, so it can exploit the performance of parallel computing environments in practice. This strategy is based on an idea proposed by Fukase and Kashiwabara in [20]; however, the parallel scalability of their algorithm is unclear with regard to massive parallelization. To achieve parallel scalability, our strategy consists of two methods. The first method is a procedure of generating different lattice bases for many processes. Our experiments show that this method enables the processes to generate a huge number of lattice vectors and that it works well even if the processes take the same lattice basis. The second method is to use global shared storage to keep information on the cooperative lattice basis reduction; this method enables many processes to share auxiliary information for accelerating the lattice basis reduction with a small synchronization overhead.

Our lattice basis reduction strategy enables us to efficiently obtain lattice bases with a small sum of squared lengths of orthogonal basis vectors, which is related to finding relatively short lattice vectors. Experiments and analyses are given by Fukase and Kashiwabara in [20], as explained in Sect. 3.2, and Aono and Nguyen in [5]. Fukase and Kashiwabara in [20] proposed a strategy to reduce this sum, but it is complicated. To efficiently obtain a lattice basis that has a

small sum of squared lengths, we propose a new strategy based on an evaluation function of lattice bases.

We applied our algorithm to problem instances in the SVP Challenge, which is hosted by Technische Universität Darmstadt [45]. We solved a problem instance of dimension 150 in about 394 days with four clusters, a world record. In addition, we solved problem instances of dimensions 134, 138, 140, 142, 144, 146, and 148.

1.3 Related Work

Kannan’s enumeration (ENUM) algorithm [33] is an exponential-time algorithm that outputs the shortest or nearly shortest lattice vector of given lattice. ENUM and its variants are used as the main routine or subroutine to solve the shortest lattice vector problem (SVP), as explained in Sect. 2.3. Thus, studying the ENUM algorithm is an important research field, and several improvements have been proposed [16, 17, 23, 30, 38].

The Lenstra–Lenstra–Lovász (LLL) algorithm [36] is a lattice basis reduction algorithm which takes a lattice basis and a parameter δ and produces a δ -LLL reduced basis of the same lattice. The LLL algorithm is the starting point of many lattice basis reduction algorithms and is a polynomial-time algorithm. Since the first basis vector of the output basis is a relatively short lattice vector in the lattice in practice, the LLL algorithm is also used as the main routine or subroutine of SVP algorithms. The Block Korkin–Zolotarev (BKZ) algorithm [49] is a block-wise generalization of the LLL algorithm, and it uses the ENUM algorithm as a subroutine. The BKZ algorithm generates relatively shorter basis vectors than the LLL algorithm does. Studying LLL, BKZ, and their variants is a popular topic of research and several improvements and extensions have been reported [6, 15, 21, 39].

Schnorr’s random sampling reduction (RSR) algorithm [47, 48] is a probabilistic lattice basis reduction algorithm. The RSR algorithm randomly generates lattice vectors. The simple sampling reduction (SSR) algorithm [13, 14, 37] is an extension of the RSR algorithm that analyzes the expected length of the lattice vectors to be generated. The authors of SSR also presented several theoretical analyses of the RSR and their algorithm.

Our algorithm is based on the Fukase–Kashiwabara (FK) algorithm [20]. The FK algorithm is an extension of the SSR algorithm. The authors of the FK algorithm refined the calculation for the expected length of lattice vectors to be generated and showed that the probability of finding short lattice vectors can be increased by reducing the sum of squared lengths of orthogonal basis vectors. A brief introduction to the FK algorithm is given in Sect. 3.2. Aono and Nguyen [5] presented a theoretical analysis of the RSR algorithm and its extensions. They also gave a probabilistic perspective and a comparison with [23].

With the aim of using parallel computation to solve the SVP, several researchers proposed parallelized algorithms [16, 17, 30, 34, 46]. In particular, parallelized ENUM algorithms on CPUs, GPUs, and FPGAs were investigated by [16, 17, 30]. A parallelized SSR algorithm for GPUs was proposed in [46]. A

design for an SVP solver that exploits large-scale computing environments was proposed by [34]. In [34], the authors integrated, tuned up, and parallel pruned the ENUM algorithm [23] into a multicore-CPU and GPU implementation [30], and they extended this implementation by using multiple CPUs and GPUs in the single program multiple data (SPMD) style [41].

Organization. Preliminaries are described in Sect. 2. We then give a brief introduction to sampling reduction and the previous studies [20, 37, 47, 48] in Sect. 3. An overview of our parallelization strategy is presented in Sect. 4, and our proposal is explained in detail in Sect. 5. Section 6 presents the results of applying our proposal to the SVP Challenge and we conclude in Sect. 7.

2 Preliminaries

2.1 Lattice

In this section, we introduce notations and definitions of the lattice.

The set of natural numbers $\{0, 1, \dots\}$ is denoted by \mathbb{N} , and the sets of integers and real numbers are denoted by \mathbb{Z} and \mathbb{R} , respectively. We denote an n -element vector (or sequence) as $\mathbf{x} = (x_1, \dots, x_n) = (x_i)_{i=1}^n$. Let $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{R}^n$ and $\mathbf{u} = (u_1, \dots, u_n) \in \mathbb{R}^n$ be two n -dimensional vectors on \mathbb{R} ; we define the inner product of \mathbf{v} and \mathbf{u} by $\langle \mathbf{v}, \mathbf{u} \rangle := \sum_{i=1}^n v_i u_i$. The length (Euclidean norm) of \mathbf{v} , denoted by $\|\mathbf{v}\|$, is $\sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$.

Let $\mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{Z}^n$ be n -dimensional linearly independent column (integral) vectors. The (full rank) lattice L spanned by a matrix $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is defined as the set $L = \mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$. Such a \mathbf{B} is called a lattice basis of L , and each $\mathbf{v} \in L$ is called a lattice vector of L . Every lattice has infinitely many lattice bases except $n = 1$; i.e., let U be an n -dimensional unimodular matrix over \mathbb{Z} ; then $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}U)$, so that $\mathbf{B}U$ is another basis of L if U is not the identity matrix. For a lattice basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$, the corresponding (Gram-Schmidt) orthogonal basis vectors $\mathbf{b}_1^*, \dots, \mathbf{b}_n^* \in \mathbb{R}^n$ are defined by $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{\mathbf{B},i,j} \mathbf{b}_j^*$, where $\mu_{\mathbf{B},i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$. The determinant of L is denoted by $\det(L)$, and note that $\det(L) = \prod_{i=1}^n \|\mathbf{b}_i^*\|$ for any basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of L .

Given a lattice basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ of a lattice L , the i -th orthogonal projection by \mathbf{B} , denoted by $\pi_{\mathbf{B},i} : \mathbb{R}^n \rightarrow \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})^\perp$, is defined as $\pi_{\mathbf{B},i}(\mathbf{v}) = \mathbf{v} - \sum_{j=1}^{i-1} v_j^* \mathbf{b}_j^*$, where $v_j^* = \langle \mathbf{v}, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2 \in \mathbb{R}$ (for $i = 1$, $\pi_{\mathbf{B},1}$ is the same as the identity map). Note that $\|\pi_{\mathbf{B},i}(\mathbf{v})\|^2 = \sum_{j=i+1}^n (\langle \mathbf{v}, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|)^2$. We define the i -th orthogonal complement by \mathbf{B} as $L_{\mathbf{B},i} := \pi_{\mathbf{B},i}(L) = \{\pi_{\mathbf{B},i}(\mathbf{v}) \mid \mathbf{v} \in L\}$. We call $\|\pi_{\mathbf{B},i}(\mathbf{v})\|$ the orthogonal length of \mathbf{v} on $L_{\mathbf{B},i}$; this is an important measurement in the context of lattice basis reduction.

2.2 Natural Number Representation

In this section, we explain the natural number representation (NNR) of a lattice vector. In Schnorr's algorithm [47], a lattice vector is represented by an

n -dimensional sequence consisting of 0s and 1s, where n is the dimension of the given lattice basis. The numbers 0 and 1 express the amount of aberration from the orthogonal point at each index, and one can calculate a lattice vector from the lattice basis and this sequence. Moreover, Buchmann and Ludwig [13, 14, 37] indicate that one can generalize 0, 1 to the natural numbers 0, 1, 2, ..., and calculate the expected value of squared lengths of the lattice vectors generated from a sequence of natural numbers. Fukase and Kashiwabara [20] call such a sequence of natural numbers a *natural number representation (NNR)*, and Aono and Nguyen [5] call it a *tag (defined on the natural numbers)*. One can determine a set of NNRs that has a small expected value of squared lengths for a given basis. The following definitions and statements are basically reprinted from [20], but similar descriptions are given in other papers [5, 13, 14, 37].

Definition 1 (Natural Number Representation). *Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ be a lattice basis of the lattice L . Given a lattice vector $\mathbf{v} = \sum_{i=1}^n \nu_i \mathbf{b}_i^* \in L$, the natural number representation (NNR) of \mathbf{v} on \mathbf{B} is a vector of n non-negative integers $(\mathbf{n}_1, \dots, \mathbf{n}_n) \in \mathbb{N}^n$ such that $-(\mathbf{n}_i + 1)/2 < \nu_i \leq -\mathbf{n}_i/2$ or $\mathbf{n}_i/2 < \nu_i \leq (\mathbf{n}_i + 1)/2$ for all $i = 1, \dots, n$.*

Theorem 1 [20]. *Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ be a lattice basis of the lattice L , and let $\mathbf{v} = \sum_{i=1}^n \nu_i \mathbf{b}_i^* \in L$. For any lattice vector $\mathbf{v} \in L$, the NNR of \mathbf{v} is uniquely determined, and the map $\mathbf{n}_\mathbf{B} : L \rightarrow \mathbb{N}^n$, which is given by $\mathbf{n}_\mathbf{B}(\mathbf{v}) := (\mathbf{n}_1, \dots, \mathbf{n}_n)$, is a bijection.*

Assumption 1 (Randomness Assumption [5, 20, 47]). *Let $\mathbf{v} = \sum_{j=1}^n \nu_j \mathbf{b}_j^*$ be a lattice vector, and \mathbf{v} has the NNR $\mathbf{n} = (\mathbf{n}_1, \dots, \mathbf{n}_n)$. The coefficients ν_j of \mathbf{v} are uniformly distributed in $(-\mathbf{n}_j + 1)/2, -\mathbf{n}_j/2]$ and $(\mathbf{n}_j/2, \mathbf{n}_j + 1)/2]$ and statistically independent with respect to j .*

The difference from the original randomness assumption [47, 48] is that the NNRs are generated by a procedure called the sampling algorithm and these NNRs belong to a subset of $\{0, 1\}^n$ defined as

$$\{(0, \dots, 0, \mathbf{n}_{n-u}, \dots, \mathbf{n}_{n-1}, 1) \mid \mathbf{n}_{n-u}, \dots, \mathbf{n}_{n-1} \in \{0, 1\}\},$$

where u is a positive integer parameter of the sampling algorithm.

Theorem 2 [20]. *Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ be a lattice basis of the lattice L , and let $\mathbf{n} = (\mathbf{n}_1, \dots, \mathbf{n}_n)$ be an NNR. The expectation of the squared length of the corresponding lattice vector \mathbf{v} of \mathbf{n} (namely, $\mathbf{n}_\mathbf{B}(\mathbf{v}) = (\mathbf{n}_1, \dots, \mathbf{n}_n)$) is:*

$$E[\|\mathbf{v}\|^2] = \frac{1}{12} \sum_{i=1}^n (3\mathbf{n}_i^2 + 3\mathbf{n}_i + 1) \|\mathbf{b}_i^*\|^2. \tag{1}$$

Theorem 2 states that one can calculate the expected value of the squared length of a generated lattice vector with given NNR in a given basis without calculating the coordinates of the lattice vector. For a relatively reduced basis (the output of the LLL algorithm, the BKZ algorithm, or other variants), the

sequence $(\|\mathbf{b}_1^*\|^2, \|\mathbf{b}_2^*\|^2, \dots, \|\mathbf{b}_n^*\|^2)$ tends to decrease with respect to the indices. Therefore, a short lattice vector tends to have natural number 0 in first consecutive indices of the NNR, and natural numbers 1 and 2 appear in last consecutive indices in the NNR for any relatively reduced basis.

2.3 Shortest Vector Problem and SVP Challenge

Given a lattice basis \mathbf{B} of a lattice L of dimension n , the *shortest vector problem (SVP)* consists in finding the shortest non-zero vector $\mathbf{v} \in L$. The SVP is NP-hard under randomized reduction [3]. It is hard to find the shortest lattice vector in the lattice when n is large. However, thanks to the *Gaussian heuristic* $\text{GH}(L)$, the length of the shortest lattice vector in L can be estimated as $\text{GH}(L) := (\Gamma(n/2 + 1) \cdot \det(L))^{1/n} / \sqrt{\pi}$, where Γ is the gamma function [32, 40]. The *root Hermite factor of \mathbf{v} and L* is defined by $(\|\mathbf{v}\| / \det(L)^{1/n})^{1/n}$. Given a lattice vector \mathbf{v} , the corresponding root Hermite factor is used to make comparative measurements among different lattices, because it is independent of the lattice basis [22, 42].

The security of lattice-based cryptosystems depends on the hardness of the lattice problem. Therefore, it is important to study efficient algorithms for solving the SVP. Here, a competition, called the *SVP Challenge* [45], was maintained by Technische Universität Darmstadt since 2010. The SVP Challenge gives a problem instance generator that produces SVP instances and hosts a ranking of registered lattice vectors as follows. A submitted lattice vector $\mathbf{v} \in L$ is registered if one of the following conditions is satisfied: no lattice vector is registered at the dimension of \mathbf{v} and $\|\mathbf{v}\| < 1.05 \cdot \text{GH}(L)$, or \mathbf{v} is shorter than any other lattice vector of the same dimension. In this paper, for a lattice L , we call a lattice vector $\mathbf{v} \in L$ a *solution of L* if $\|\mathbf{v}\| < 1.05 \cdot \text{GH}(L)$.

3 Brief Introduction to Sampling Reduction

Before explaining our algorithm, we briefly introduce the RSR algorithm and its extensions [20, 37, 47, 48].

3.1 Overview of Sampling Reduction

Here, we briefly introduce the RSR algorithm and its extensions [20, 37, 47, 48] and describe the parts that our algorithm shares with them.

First of all, we define the following notions and terminology.

Definition 2 (lexicographical order). Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ and $\mathbf{C} = (\mathbf{c}_1, \dots, \mathbf{c}_n)$ be two lattice bases of the same lattice. We say that \mathbf{B} is smaller than \mathbf{C} in the lexicographical order if and only if $\|\mathbf{b}_j^*\| < \|\mathbf{c}_j^*\|$, where j is the smallest index for which $\|\mathbf{b}_j^*\|$ and $\|\mathbf{c}_j^*\|$ are different.

Definition 3 (insertion index). Given a lattice vector \mathbf{v} and a real number δ with $0 < \delta \leq 1$, the δ -insertion index of \mathbf{v} for \mathbf{B} is defined by

$$\mathbf{h}_\delta(\mathbf{v}) = \min(\{i \mid \|\pi_{\mathbf{B},i}(\mathbf{v})\|^2 < \delta\|\mathbf{b}_i^*\|^2\} \cup \{\infty\}).$$

Note that the δ -insertion index is a parameterized notion of Schnorr [47, 48].

The RSR algorithm and its extensions need a lattice reduction algorithm to reduce a given lattice basis in lexicographical order, and the LLL and BKZ algorithms can be used for this purpose. In this paper, we define this operation as follows.

Definition 4. Let $\mathbf{v} \in L$ be a lattice vector with $\|\pi_{\mathbf{B},i}(\mathbf{v})\| < \delta\|\mathbf{b}_i^*\|$. We call the following operation δ -LLL reduction of \mathbf{B} at index i by \mathbf{v} : generate an $(n+1) \times n$ matrix $M = (\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{v}, \mathbf{b}_i, \dots, \mathbf{b}_n)$, compute the δ -LLL reduced (full rank) lattice basis \mathbf{B}' by using the LLL algorithm with input M , and output the resulting lattice basis.

As mentioned above, one can use another lattice reduction algorithm. In fact, the RSR algorithm [47, 48] uses the BKZ algorithm.

The RSR algorithm and its extensions consist of two major parts. The first part takes a lattice basis \mathbf{B} and a set of NNRs as input and generates a lattice vector that corresponds to an NNR and \mathbf{B} . It outputs a set of generated lattice vectors. In this paper, we call this operation *lattice vector generation*. The second part performs lattice basis reduction. If the result of the lattice vector generation is a lattice vector \mathbf{v} whose orthogonal length is shorter than a orthogonal basis vector of \mathbf{B} , in other words, whose δ -insertion index is finite, \mathbf{B} is reduced by using δ -LLL reduction of \mathbf{B} at index $\mathbf{h}_\delta(\mathbf{v})$ by \mathbf{v} . Note that the lexicographical order of the resulting lattice basis of this part is smaller than the input. To summarize the RSR algorithm and its extensions, alternately repeat the lattice vector generation and the lattice basis reduction to reduce a given lattice basis. To solve the SVP, these calculations are repeated until the length of the first basis vector of the obtained lattice basis is less than $1.05 \cdot \text{GH}(\mathcal{L}(\mathbf{B}))$, where \mathbf{B} is the given lattice basis.

To reduce the lattice basis or solve the SVP efficiently, it is important to choose a lattice vector as input for the lattice basis reduction from the result of the lattice vector generation. Each previous study has its own selection rule. In the RSR algorithm, a lattice vector \mathbf{v} with $\mathbf{h}_\delta(\mathbf{v}) \leq 10$ is chosen from the result of the lattice vector generation. In the SSR algorithm, a lattice vector \mathbf{v} with $\mathbf{h}_\delta(\mathbf{v}) = 1$ is chosen. Fukase and Kashiwabara [20] introduce a rule to choose a lattice vector, called restricting reduction, as explained in Sect. 3.2.

Ludwig [37] and Fukase and Kashiwabara [20] computed the expectation of the squared length of the generated lattice vectors in the lattice vector generation. According to the expectation value of squared length, their algorithm prepared lists of NNRs with 0, 1, and 2, and generated the corresponding lattice vectors from the lattice basis. Aono and Nguyen [5] presented an algorithm to produce a set of better NNRs (in their terms, tags).

3.2 Fukase–Kashiwabara Algorithm

As our algorithm is an extension of a parallelization strategy and uses the global shared storage proposed by Fukase and Kashiwabara [20], we will briefly introduce their proposal before explaining ours. Note that our algorithm does not use the restricting reduction of their algorithm. We refer the reader to [20] for the details of the FK algorithm.

Restricting Reduction. Ludwig [37] and Fukase and Kashiwabara [20] showed that one can calculate the expected value of the squared lengths of the generated lattice vectors for a lattice basis and a set of NNRs, and this expectation is proportional to the sum of the squared lengths of the orthogonal basis vectors. Therefore, a nice reduction strategy seems to reduce the lattice basis so as to decrease the sum of squared lengths, and hence, such a strategy is considerable. Fukase and Kashiwabara [20] proposed a method for decreasing the sum of squared lengths by introducing the restricting reduction index; this strategy is briefly explained below.

To decrease the sum of squared lengths quickly, some application of a lattice vector to the lattice basis is not good. Moreover, some application of a lattice vector which decreases the sum of squared lengths is not the fast way to reach a very small sum of squared lengths. To decrease the sum of squared lengths quickly, Fukase and Kashiwabara [20] used a restricting index. For an index ℓ , called the restricting reduction index, the lattice basis is reduced at an index ℓ' only after ℓ . In other words, the basis vectors before ℓ are fixed. Let ℓ be 0 at the beginning of the program. As the main loops are executed, the restricting index is gradually increased according to some rule. When the restricting index reaches a certain value, it is set to 0 again. By using the restricting index, the sum of squared lengths of orthogonal basis vectors is decreased from first consecutive indices.

Stock Vectors. Some lattice vectors may be useful even if they are not short enough for reducing the lattice basis. For example, after reduction of the lattice basis at index i by another lattice vector, a lattice vector \mathbf{v} may be applied at index $i + 1$ or greater. In addition, after changing the restricting index to 0, a lattice vector \mathbf{v} may be applied to the lattice basis if it was not used for reducing the lattice basis because of a previous high restriction index. Therefore, lattice vectors that are relatively shorter than the i -th orthogonal basis vector are stored in global shared storage, and each stored lattice vector is associated with an orthogonal complement $L_{B,i}$ (namely, the place of the stored vector is determined by the basis vectors $\mathbf{b}_1, \dots, \mathbf{b}_{i-1}$) in order to load and use it for reducing the lattice basis by all the parallel running processes. Fukase and Kashiwabara [20] call these stored vectors *stock vectors*.

Parallelization. A set of generated lattice vectors is determined from the lattice basis and set of NNRs. Therefore, even if the sets of NNRs are the same,

two different lattice bases generate different sets of lattice vectors that have few vectors in common. Fukase and Kashiwabara [20] utilized this heuristic approach with stock vectors to generate a lot of lattice vectors and cooperate with each parallel running process. Given a lattice basis, their algorithm generates different lattice bases except for first consecutive indices from 1 and distributes each generated basis to each process. In other words, the processes have lattice bases that have different basis vectors at last consecutive indices. Since these lattice bases have a common orthogonal complement at first consecutive indices, processes can share stock vectors associated with common basis vectors of first consecutive indices, and since they have different basis vectors at last consecutive indices, the generated lattice vectors could be different from each other.

Note that basis reduction is done by each process, and keeping first consecutive basis vectors the same is difficult. Thus, some mechanism is required to keep first consecutive basis vectors the same among all the processes. Fukase and Kashiwabara [20] proposed a method by storing basis vectors as the stock vectors.

4 Overview of Our Parallelization Strategy

First, we briefly explain parallelization and the problems that make it hard to exploit the performance of parallel computation. Then, we briefly describe our strategy.

4.1 Technical Hurdles and Brief Review of Parallelization Methodology

Several researchers have used parallel computation to solve the SVP. In particular, there are three types of parallel algorithm for solving it.

The first type is the divide-and-conquer approach for parallelizing the ENUM, pruned ENUM, and sampling algorithms of lattice vectors [16, 17, 23, 30, 34, 46]. Several researchers [16, 17, 30, 34] studied parallel ENUM algorithms based on this methodology. This type of parallelization can be used to speed up the search for the shortest or a relatively short lattice vector. However, it is hard to solve high-dimensional SVPs by using this type of parallelization, which take an exponentially long time in practice. Hence, this type of parallelization is out of the scope of this paper.

The second type is randomization of lattice bases to exploit the power of massively parallel computing environments. The authors of reference [34] proposed an implementation to solve high-dimensional SVPs by exploiting massively parallel running processes. Their implementation works as follows. For a given lattice basis, the method generates a lot of lattice bases by multiplications with random unimodular matrices. The generated lattice bases are distributed to the processes, and each process runs the BKZ and pruned parallel ENUM algorithms. This method reduces a lot of lattice bases simultaneously, so the probability of

finding a short lattice vector becomes higher. However, each process works independently; therefore, the information for accelerating lattice basis reduction is not shared among processes.

The third type of parallelization, proposed by Fukase and Kashiwabara [20], involves alteration of last consecutive basis vectors, which is briefly explained in Sect. 3.2. For a given lattice basis, their method generates a lot of lattice vectors from the given basis and a set of NNRs in order to find lattice vectors with a shorter length than the given orthogonal basis vectors. Their key idea consists of two heuristic approaches. Firstly, each process executes lattice vector generation, which takes the same set of NNRs and a different lattice basis as input in order to generate a different set of lattice vectors. To generate many different lattice bases, each process computes the δ -LLL reduction of a lattice basis of dimension n at last consecutive indices $\ell, \ell+1, \dots, n$ by using a different lattice vector. Note that different lattice bases might be outputted for different input lattice vectors. Secondly, if the processes have different lattice bases but same first consecutive basis vectors with indices $1, 2, \dots, \ell'$, then useful lattice vectors, which are contained in the same orthogonal complements (called stock vectors by Fukase and Kashiwabara), could be shared among the processes in order to accelerate the lattice basis reduction. Fukase and Kashiwabara proposed a parallelization strategy based on these heuristics. However, they reported experimental results with only 12 or fewer parallel processes. In fact, in our preliminary experiments of the parallelization strategy of Fukase and Kashiwabara [20], we observed that many parallel running processes to solve higher dimensional problems did not keep the same first consecutive basis vectors. Since current massively parallel computing environments have many more physical processing units (often, more than 100), we conclude that there is still room for scalable massive parallelization.

The parallelization strategy in our algorithm is an extension of [20]. The key idea of our method is explained next.

4.2 Key Idea of Our Parallelization Strategy

To reduce a lattice basis by utilizing the performance of a parallel computing environment effectively, one needs a scalable parallelization strategy that keeps first consecutive indices' basis vectors common but last consecutive indices' basis vectors different on the many lattice bases of the individual processes with a small synchronization penalty in practice. To accomplish this, we extend the parallelization strategy of Fukase and Kashiwabara [20]. Our strategy consists of two methodologies. The first enables each process to generate a lattice basis variant which has different last consecutive indices' basis vectors from those of other processes, and the second enables each process to share first consecutive indices' basis vectors. The rest of this section briefly explains our idea, and Sect. 5.2 explains it in detail.

Our lattice basis reduction with process-unique information (e.g., process ID, MPI rank, etc.) generates many lattice basis variants that have different last consecutive indices' basis vectors. In addition, each process generates a lot of lattice vectors to reduce the lattice basis by using a given lattice basis and a

set of NNRs. If there is a lattice vector whose orthogonal length is relatively shorter than the orthogonal basis vectors in first consecutive indices, the process stores it in the global shared storage of stock vectors, by using the method proposed by Fukase and Kashiwabara [20], as explained in Sect. 3.2. Thanks to the process-unique information, each process can use this information as a seed for generating different lattice bases even if each process takes the same lattice basis as input. During the reduction execution, the processes work independently except for the access to the global shared storage and do not communicate with each other. The most computationally expensive operation is generating a lot of lattice vectors, and this operation is calculated in each process independently. Hence, there is only a negligible synchronization penalty in practice.

As mentioned above regarding the second method, Fukase and Kashiwabara [20] suggested that cooperation among parallel running processes can be enabled by sharing first consecutive indices' basis vectors. However, maintaining this situation without incurring a painfully large synchronization cost is a difficult task when there are many processes. Fukase and Kashiwabara [20] proposed to use global shared storage for stock vectors, as explained in Sect. 3.2. However, it is unclear how their method can be used to keep common basis vectors in first consecutive indices among many processes. We propose a method to share the basis vectors in first consecutive indices by using additional global shared storage. In our algorithm, each process has its own lattice basis. In every loop, each process stores basis vectors in first consecutive indices of its own lattice basis in this extra global shared storage; then it loads lattice vectors from the storage and computes the smallest lattice basis in lexicographical order by using the stored lattice vectors and their own lattice bases. Note that only first consecutive indices' basis vectors are stored in the global storage. In this method, many processes cause accesses to the global shared storage, and synchronization is required. However, the synchronization penalty is practically smaller than that of communication among many processes, and as mentioned above, the most computationally expensive operation is generating lattice vectors.

5 Our Algorithm

The most important part of our proposal is the methodology of parallelization. We designed and implemented our parallelization methodology in the single program multiple data (SPMD) style [41]. Namely, each process executes the same program, but each process has a different internal state from each other. Our parallelization is an extension of the parallelization strategy and uses the global shared storage and stock vectors proposed by Fukase and Kashiwabara [20], as explained in Sect. 3.2. In addition, it contains a lattice basis reduction strategy based on our new evaluation function.

Our algorithm, listed in Algorithms 1 and 2, is based on the Fukase–Kashiwabara (FK) algorithm [20]; in particular, its lattice basis reduction is similar to that method, namely, by generating a lot of lattice vectors by using a lattice basis \mathbf{B} and a set of natural number representations (NNRs), as defined

<p>Input: A lattice basis \mathbf{B}, process-unique information <code>pui</code> (regarded as an integer), two sets S and S' of natural number representations, integers ℓ_{fc}, ℓ_{lc}, and ℓ_{link}, and real values δ_{stock}, δ, δ', δ'', and Θ.</p> <p>Output: A lattice basis \mathbf{B}.</p> <pre> 1 begin 2 loop begin 3 $\mathbf{B}' \leftarrow \mathbf{B}$; 4 Reduce \mathbf{B} by using Algorithm 2 with input parameters \mathbf{B}, <code>pui</code>, S', ℓ_{fc}, ℓ_{lc}, δ_{stock}, δ, δ' and δ'' (explained in Sect. 5.2); 5 Generate a set of lattice vectors V from \mathbf{B} and a set S of natural number representations; 6 foreach $v \in V$ do 7 if there exists an index $i = \mathbf{h}_{\delta_{\text{stock}}}(v)$ such that $1 \leq i \leq \ell_{\text{fc}}$ then 8 store v in the global shared storage of stock vectors (explained in Sect. 3.2); 9 end 10 Reduce \mathbf{B} by using our strategy with inputs lattice vectors from the global shared storage of stock vectors and the parameter Θ (explained in Sect. 5.1) ; 11 Store the basis vectors of indices from 1 to ℓ_{link} of \mathbf{B} in the global shared storage of link vectors (explained in Sect. 5.2); 12 Load the lattice vectors from the global shared storage of link vectors; then generate the smallest lattice basis \mathbf{B}'' in lexicographical order by using loaded link vectors and \mathbf{B}; then replace \mathbf{B} by \mathbf{B}'' (explained in Sect. 5.2); 13 if there exists a lattice vector v in the global shared storage of stock vectors and \mathbf{B} with $\ v\ < 1.05 \cdot \text{GH}(\mathcal{L}(\mathbf{B}))$ then output v and halt this process; 14 if $\mathbf{B}' = \mathbf{B}$ then halt this process and output \mathbf{B}; 15 end 16 end </pre>

Algorithm 1. Main routine of our algorithm

by Definition 1 in Sect. 2.2, and reducing \mathbf{B} to significantly decrease the sum of squared lengths of orthogonal basis vectors of \mathbf{B} by using the generated vectors.

Our method to generate a lot of lattice vectors from a lattice basis is similar to the FK algorithm. Note that Aono and Nguyen [5] presented an algorithm to produce a set of better NNRs (in their terms, tags), but we did not use it.

The major differences from the FK algorithm are as follows: a parallelization strategy, and a lattice basis reduction strategy based on our evaluation function in order to decrease the sum of squared lengths significantly for each process. In Sect. 5.1, we show our new lattice basis reduction strategy for each process. In Sect. 5.2, we describe our new parallelization strategy. We explain how to choose parameters of our proposed algorithm in Sect. 5.3.


```

Input: A lattice basis  $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ , the process-unique information pu
  (regarded as an integer), a set  $S'$  of natural number representations,
  integers  $\ell_{fc}$ ,  $\ell_{lc}$ , and real values  $\delta_{stock}$ ,  $\delta$ ,  $\delta'$ , and  $\delta''$ .
Output: A lattice basis  $\mathbf{B}$ .
1 begin
2    $\delta'_i \leftarrow \delta'$  for  $i = \ell_{lc} + 1, \dots, n$  where  $n$  is the dimension of given lattice;
3   loop begin
4     Generate a set of lattice vectors  $V$  from  $\mathbf{B}$  and a set  $S'$  of natural
5     number representations;
6     foreach  $v \in V$  do
7       if there exists an index  $i = \mathbf{h}_{\delta_{stock}}(v)$  such that  $1 \leq i \leq \ell_{fc}$  then
8         store  $v$  in the global shared storage of the stock vectors (explained
9         in Sect. 3.2);
10      end
11     Collect the lattice vectors  $V' = \{v_1, \dots, v_N\}$  that have a  $\delta$ -insertion
12     index at an index  $i$  with  $\ell_{lc} < i$  from  $V$ ;
13      $\mathbf{B}' \leftarrow \mathbf{B}$ ;
14     loop begin
15       Find the lattice vector  $v_i \in V'$  with  $j = \mathbf{h}_{\delta'_j}(v)$  such that
16        $\ell_{lc} < j \leq n$  and  $\|\mathbf{b}_j^*\|^2 - \|\pi_{B,j}(v_i)\|^2$  is maximum;
17       if  $v_i$  is not found in line 11 then go to line 18;
18       Reduce  $\mathbf{B}$  by  $\delta$ -LLL reduction at the index  $j$  by  $v_i$ ;
19       foreach  $k = \ell_{lc} + 1, \dots, j - 1$  do  $\delta'_k \leftarrow \delta'_k - \delta''$ ;
20       foreach  $k = j, \dots, n$  do  $\delta'_k \leftarrow \delta'$ ;
21       if  $(i + \text{pu}) \bmod N = 0$  then go to line 18;
22     end
23     if  $\mathbf{B} = \mathbf{B}'$  then terminate this subroutine and output  $\mathbf{B}$ ;
24   end
25 end

```

Algorithm 2. Subroutine of our algorithm to generate lattice basis variants

5.1 Basis Reduction Strategy Using Evaluation Function

As mentioned above, our algorithm is based on the FK algorithm [20]. FK algorithm adopted restricting index as its lattice basis reduction method. But our algorithm uses a following evaluation function of bases on lattice basis reduction. Our algorithm generates a lot of lattice vectors by using a lattice basis \mathbf{B} and a set of NNRS. Discovered relatively shorter lattice vectors, called stock vectors, are stored in global shared storage [20], as explained in Sect. 3.2. After generating these stock vectors, our algorithm uses them to reduce \mathbf{B} . Several stock vectors have the δ -insertion index at an index i for \mathbf{B} , which is defined in Sect. 2.1, so that there are several choices of stock vector that can reduce \mathbf{B} . Now we face the problem of deciding which stock vector is the best to reduce \mathbf{B} . An answer to this question is important for finding a solution efficiently.

However, based on our preliminary experiments with reducing higher dimensional lattice bases, this is quite difficult problem. In order to manage to resolve

this difficulty, we introduce the following evaluation function Eval and a positive real number parameter Θ less than 1 for the lattice basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$:

$$\text{Eval}(\mathbf{B}, \Theta) = \sum_{i=1}^n \Theta^i \|\mathbf{b}_i^*\|^2. \quad (2)$$

It uses the following strategy at line 9. Replace \mathbf{B} by a lattice basis $\overline{\mathbf{B}}$ which has the minimum $\text{Eval}(\overline{\mathbf{B}}, \Theta)$, where $\overline{\mathbf{B}}$ is computed by the δ -LLL reduction of \mathbf{B} at index i by \mathbf{v} , which is a lattice vector from the global shared storage of stock vectors with $\mathbf{h}_{\delta_{\text{stock}}}(\mathbf{v}) = i \leq \ell_{\text{fc}}$.

Now let us precisely describe what is difficulty and why we introduced Eval and Θ . According to [13, 14, 20, 37], the average squared lengths of lattice vectors generated by \mathbf{B} and the set of NNRs are proportional to the sum of squared lengths of orthogonal basis vectors of \mathbf{B} , explained in Sect. 2.2. In this way, the SVP can be solved by searching for shorter lattice vectors and using the found lattice vector to reduce the lattice basis. At this time, lattice basis reduction with a shorter lattice vector yields a reduced lattice basis in the context of the lexicographical order, which is defined in Sect. 2.1; however, the sum of the squared lengths of orthogonal basis vectors of the resulting lattice basis becomes much larger in practice. Namely, there is a trade-off between two reduction strategies (decreasing the lexicographical order and the sum of squared lengths of orthogonal basis vectors). To resolve this dilemma, a method for optimally choosing between these strategies is needed. However, as mentioned above, it seems to be quite difficult because, to the best of our knowledge, basic theories allowing this have not been developed in this research area. In order to manage this issue, we decide to combine both strategies and this is the reason why we introduced Eval and Θ .

The evaluation function Eval attempts to combine the lexicographical order and the order defined by the sum of squared lengths with an adjustable parameter Θ . For a lattice basis \mathbf{B} , if Θ is close to 1, then $\text{Eval}(\mathbf{B}, \Theta)$ considers that decreasing the sum of squared lengths of the orthogonal basis vectors is more important, otherwise decreasing the lexicographical order is more important. Eval balances the two orders and yields a flexible and efficient basis reduction strategy for solving the SVP.

5.2 Parallelization Strategy

Now, we will explain parallelizing the computations method of our lattice reduction algorithm. If the parallel processes have the same lattice basis and generate the same set of lattice vectors, these parallel computations would be useless. As stated in Sect. 4, we generate different lattice bases for each process. However, if parallel processes have completely different lattice bases, the parallel computation cannot be cooperative.

Fukase and Kashiwabara [20] proposed a parallel computation method such that the basis vectors in first consecutive indices are the same among the processes and the basis vectors in last consecutive indices are different among the

processes. Since the basis vectors at last consecutive indices are different, the same set of natural number representations generates a different set of lattice vectors. However, they did not describe an effective parallel computation for more than 100 processes. We propose an effective method for parallel computations, in which each process has same first consecutive basis vectors but different last consecutive basis vectors. In our method, the computational cost of synchronization is practically small. Our method consists of two parts: a part in which each process has different last consecutive basis vectors, and method part in which each process has common first consecutive basis vectors.

Lattice Basis Variants Generation. Here, we explain our new lattice reduction procedure with process-unique information in order to generate many lattice basis variants that have different basis vectors in last consecutive indices. Each process executes this procedure independently with process-unique information pui . The procedure is shown in Algorithm 2.

The discontinue rule at line 16 in Algorithm 2 is important. This rule depends on the process-unique information pui such that each process executes different lattice basis reduction operations for last consecutive indices of the lattice basis. Thanks to this procedure, a huge number of processes can generate many lattice basis variants even if they have the same lattice basis without communicating with each other.

Cooperation Between Many Processes. As mentioned above, one can make a large amount of processes whose lattice bases have different basis vectors in last consecutive indices. Since these processes have the same basis vectors in first consecutive indices from 1, they utilize the stock vectors they have in common. Fukase and Kashiwabara [20] proposed a method to keep this situation by using stock vectors, as explained in Sect. 3.2; however, it is difficult to maintain for a large number of processes solving higher dimensional problems without imposing a heavy synchronization penalty.

We propose a method to solve the above problem. In it, all the processes share additional global storage, and the basis vectors of the lattice basis possessed by each process are stored in it in the same manner as stock vectors but with a different strategy as explained next. We call the stored lattice vectors *link vectors*. When each process has a new lattice basis, its basis vectors are stored as link vectors. After each process loads link vectors from their storage, it uses them to reduce the lattice basis to the smallest one in the lexicographical order. In this method, the synchronization penalty (computational cost) consists of storing and loading link vectors from storage, but this penalty is practically small.

Note that we can adjust the parameter δ_{stock} in such a way that quite a few stock vectors are found in one process and one main loop evaluation, and we run around thousand processes in parallel, each of which stores stock vectors independently. However, the number of stored stock vectors at any given time is not that large, as we do not keep all stock vectors. This is because old stock

vectors are relatively longer than orthogonal basis vectors possessed by running processes, and it is therefore not necessary to store these, and we throw them away. This parameter adjustment is also explained in next subsection. The same strategy was used for the link vectors.

5.3 Parameter Choice

Our algorithm takes a basis \mathbf{B} and process-unique information pui with parameters ℓ_{fc} , ℓ_{lc} , δ , δ' , δ'' , δ_{stock} , ℓ_{link} , Θ , and two sets S and S' of NNRs used in line 5 in Algorithm 1 and line 4 in Algorithm 2 as input. \mathbf{B} is given by a user, and pui is given by a user or a facility, e.g., MPI libraries and a job scheduler, but choosing the remaining parameters is not trivial. We did not find an optimal choice of parameters because, to the best of our knowledge, basic theories allowing this have not been developed. In this section, we explain a method of parameter choice for our algorithm. Note that the concrete chosen values of the parameters when we solved a 150-dimensional problem instance appear in the next section.

More precisely, we observed intermediate values in the main loop of our algorithm (lines 2–14 in Algorithm 1) with input a basis (e.g., 150-dimensional problem instance in the SVP Challenge). The intermediate values which we observed are as follows: (1) The number of found stock vectors. (2) The sum of squared lengths of the orthogonal basis vectors. (3) The calculation times of the two lattice vector generations in the main loop called at lines 4 and 5 in Algorithm 1.

We want that (1) is quite small because we are only interested in relatively short lattice vectors, and adjusting this can be immediately done by decreasing δ_{stock} .

We adjust parameters ℓ_{fc} , ℓ_{lc} , ℓ_{link} , and Θ to appropriate values such that the processes running in parallel can keep a small value of (2) when reducing the basis. As mentioned in Sect. 5.1, it is quite hard to determine the value of Θ , so that we choose and adjust the parameters by observations regarding the performance and the intermediate values of the main loop of our algorithm.

Regarding (3), we adjust ℓ_{lc} , δ , δ' , δ'' , the set S of NNRs at line 5 in Algorithm 1, and the set S' of NNRs at line 4 in Algorithm 2. Our algorithm has two subroutines to generate lattice vectors by using S and S' (in line 5 in Algorithm 1 and line 4 in Algorithm 2), and they have different purposes. Since these lattice vector generations are the most costly parts of our algorithm, it is important to optimize the choices of S and S' ; however, these sets differ only in their size. Note that NNRs in S and S' are chosen by ascending order of expectation of squared lengths (shown in Theorem 2) of generated lattice vectors with several bases.

The purpose of the lattice vector generation with S (at line 5 in Algorithm 1) is finding relatively short lattice vectors, so the size of S should be large. The purpose of the other lattice vector generation with S' (at line 4 in Algorithm 2) is decreasing the sum of squared length of orthogonal basis vectors and generating many lattice bases variants as mentioned in Sect. 5.2 by repeatedly applying

lattice reduction (e.g., the LLL algorithm), so it is not necessary to choose the size of S' as large.

Actually, we adjust the parameters ℓ_{lc} , δ , δ' , δ'' , and sizes of sets S and S' in such a way that the calculation times of two lattice vector generations with S and S' are around several minutes. More concretely, we chose the parameters such that the calculation times are 5 min and 10 min, respectively, and the lattice vector generation with S' is repeatedly evaluated around 500 times per one Algorithm 2 evaluation on average when we found a solution of a 150-dimensional problem instance of the SVP Challenge.

6 Application to SVP Challenge

The shortest vector problem (SVP) is an important problem because of its relation to the security strength of lattice-based cryptosystems, and studying algorithms for solving it is an important research topic. The SVP Challenge was started in 2010 [45] as a way of advancing this research. The challenge provides a problem instance generator and accepts submissions of discovered short lattice vectors. As mentioned in Sect. 2.3, a submitted lattice vector of a lattice L is inducted into the hall-of-fame if its length is less than $1.05 \cdot \text{GH}(L)$ and less than the lengths of other registered lattice vectors if they exist for the same dimension.

6.1 Equipment

We applied our algorithm to the problems in the SVP Challenge. In this section, we explain our equipment and results.

First, let us describe our equipment. We used four clusters, called Oakleaf-FX (FX10), Chimera, ASGC, and Reedbush-U. The hardware specifications of these clusters are summarized in Tables 1, 2, 3, and 4. The FX10 cluster included a customized compiler, MPI library, and job scheduler. The Chimera, ASGC, and Reedbush-U clusters included compilers, MPI libraries, and job schedulers. In particular, we used GCC version 4.8.4, OpenMPI version 1.8.1, and Univa Grid Engine version 8.1.7 in the Chimera, Intel C++ Compiler version 14.0.2, Intel MPI Library version 4.1, and TORQUE version 4.2.8 in the ASGC, and Intel C++ Compiler version 16.0.3, Intel MPI Library version 5.1, and PBS Professional 13.1 in the Reedbush-U. Note that Hyper-Threading was enabled and Turbo Boost was disabled in all nodes of Chimera, while Hyper-Threading

Table 1. Specifications of cluster system Oakleaf-FX to solve the 134-dimension SVP Challenge; note that the units of “CPU frequency” and “Total RAM” are GHz and GB, respectively

CPU	CPU frequency	# of nodes	Total # of cores	Total RAM
SPARC64 IXfx	1.848	6	96	192

Table 2. Specifications of cluster system Chimera to solve SVP Challenge instances of 138, 140, 142, 144, 146, 148, and 150 dimensions; note that the units of “CPU frequency” and “Total RAM” are GHz and GB, respectively

CPU	CPU frequency	# of nodes	Total # of cores	Total RAM
Xeon E5540	2.53	80	640	6240
Xeon X5650	2.66	11	132	765
Xeon E5-2670	2.6	4	64	762
Xeon E5-2695 v3	2.3	2	56	256
Xeon E7-4870	2.4	1	80	4096
Grand total		98	972	12119

Table 3. Specifications of cluster system ASGC to solve 148 and 150-dimensions instances of SVP Challenge; note that the units of “CPU frequency” and “Total RAM” are GHz and GB, respectively

CPU	CPU frequency	# of nodes	Total # of cores	Total RAM
Xeon E5-2680 v2	2.8	5	100	640

Table 4. Specifications of cluster system Reedbush-U to solve the 150-dimensional instance of the SVP Challenge; note that the units of “CPU frequency” and “Total RAM” are GHz and GB, respectively

CPU	CPU frequency	# of nodes	Total # of cores	Total RAM
Xeon E5-2695 v4	2.1	24	864	5856

was disabled and Turbo Boost was disabled in all nodes of ASGC and Reedbush-U. Note also that these clusters were shared systems, so all the processes were controlled by the job schedulers. Now, let us briefly explain the pre-processing. Before executing processes for our algorithm, the lattice basis of the input was pre-processed. We used the BKZ algorithm implemented by `fpLLL` [50] to reduce components of the lattice basis to small integers.

For each process when we solve the problem instances, the process-unique information pui was given as the rank by the MPI libraries and job schedulers, namely, the pui value ranged from 0 to $m-1$, where m is the number of requested processes sent to the MPI libraries and job schedulers.

6.2 Experimental Results

We used our algorithm to solve SVP Challenge problem instances of 134, 138, 140, 142, 144, 146, 148, and 150 dimensions with seed 0. We used FX10 to solve the 134-dimensional instance, Chimera to solve the instances with 138–146 dimensions, and the two clusters Chimera and ASGC to solve the 148-dimensional instance.

To solve the 150-dimensional instance, we used the following numbers of cores and CPUs of the four clusters: (1) 28 cores of Xeon E5-2695 v3 and (2) 80 cores of Xeon E7-4870 in the parts of Chimera, (3) 100 cores of Xeon E5-2680 v2 in ASGC, (4) 192 cores of SPARC64 IXfx per job in FX10, and (5) 864 cores of Xeon E5-2695 v4 per job in Reedbush-U, for (1) 49 days, (2) 81 days, (3) 39 days, (4) 37 days, and (5) 188 days, respectively. We did not use these computing nodes simultaneously. Therefore, the total wall-clock time was about 394 days. Note that when we used the computing node (1), we requested 24 processes for job schedulers and the other cases (2), (3), (4), and (5), and the number of requested processes was the same as the number of cores.

When we found a solution of the 150-dimensional instance, tuned parameters of our algorithm were as follows: $\ell_{fc} = 50$, $\ell_{lc} = 50$, $\delta = 0.9999$, $\delta' = 0.985$, $\delta'' = 0.0002$, $\delta_{stock} = 1.08$, $\ell_{link} = 50$, and $\Theta = 0.93$. The method we used to choose these parameters is explained in Sect. 5.3.

We clarify the numbers of lattice vectors generated by the two subroutines in line 5 in Algorithm 1 with a set S of NNRs and in line 4 in Algorithm 2 with a set S' of NNRs, and the costs per generation when we solved the 150-dimensional problem and problems with a smaller dimensions less than 150. The sizes of S and S' are around 30 millions NNRs and 50 thousands NNRs, respectively. Note that lines 3–19 in Algorithm 2 are repeated until the termination condition in line 18 is satisfied, and the number of loops is 500 on average. The calculation times of the lattice vector generations with S and S' are around 5 min and 10 min, respectively. That is, the calculation time of these generations per lattice vector is around 12–20 μ s.

As mentioned in Sect. 5.2, it is not necessary to store all the stock vectors and link vectors, because old stock vectors and link vectors are relatively longer, so that we threw away old stock vectors and link vectors, and we cannot show the total size. The size of the remaining stock vectors and link vectors is around 60 gigabytes. We estimate that the total size of all the found stock vectors and link vectors is several hundred gigabytes for solving the 150-dimensional problem.

Table 5 lists the details of our solutions. The solution of the 150-dimensional problem is the current world record.

We show a comparison of solved dimension and single thread calculation time in seconds of our results and the results from the SVP Challenge in Fig. 1. In this figure, plotted boxes are our 8 results, and plotted circles are other 17 results for problems with dimensions greater than or equal to 100 and specified overall calculation times. It is clear that our algorithm outperforms others. This is experimental evidence that our proposal is effective at reducing higher-dimensional lattice bases and solving higher-dimensional SVPs.

Additionally, we show fittings of our and the SVP Challenge results in Fig. 1 obtained by `fit` function of GNU Plot version 5.2. As a result, we obtained the straight solid line for our results and the dotted line for the other SVP Challenge results. Note that these lines are expressed as $f(x) = c^{ax+b} + d$ with four variables a, b, c, d . Obtained values and asymptotic standard errors of a, b, c, d , and values of the root mean square (RMS) of residuals by using `fit` are as follows: $a =$

Table 5. Our experimental results for the SVP Challenge instances of dimensions 134, 138, 140, 142, 144, 146, 148, and 150, where “GH(L)” is the Gaussian heuristic of a given lattice L , “ v ” in the column header indicates a solution of the corresponding row, the “Max. # of requested processes” column shows the maximum number of requested processes sent to the job schedulers, and the “Days to solve” column lists the wall-clock times to obtain each solution

Dimension	$\lfloor \text{GH}(L) \rfloor$	$\lfloor \ v\ \rfloor$	$\frac{\ v\ }{\text{GH}(L)}$	Max. # of requested processes	Days to solve	Root Hermite factor
150	3090	3220	1.04192	864	394	1.00768
148	3070	3178	1.03512	1000	256	1.00769
146	3056	3195	1.04534	1000	64	1.00783
144	3025	3154	1.04284	700	50	1.00787
142	3003	3141	1.04609	700	50	1.00796
140	2991	3025	1.01139	500	50	1.00778
138	2972	3077	1.03516	500	140	1.00801
134	2927	2976	1.01695	92	72	1.00801

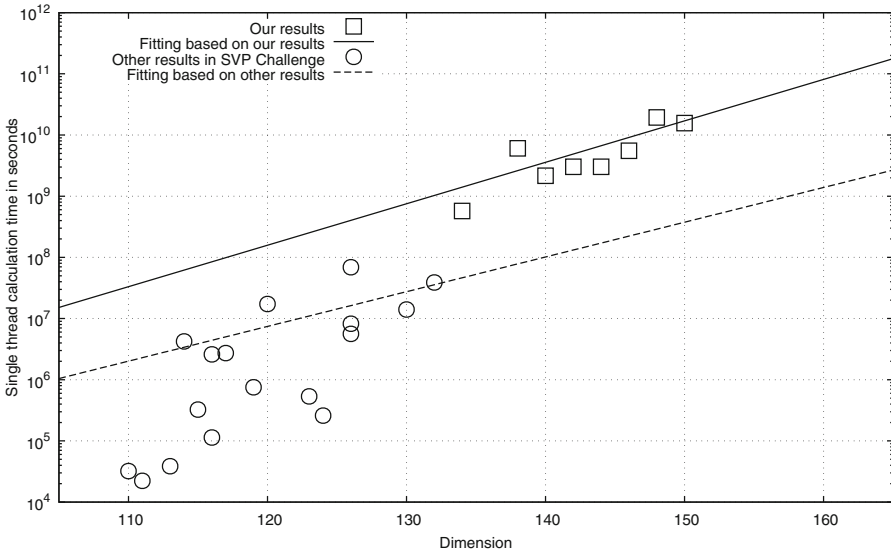


Fig. 1. Comparison of solved dimension and single thread calculation time in seconds of our results and results in the SVP Challenge, where plotted boxes are our 8 results, the straight solid line is fitted to our results, plotted circles are other 17 results of problems with dimensions greater than or equal to 100 and specified overall calculation times, and the dotted line is fitted to these results

1.00287 ± 933.3 , $b = 0.999973 \pm 7822$, $c = 1.16833 \pm 148.9$, and $d = 1.00093 \pm 2.948e + 10$, and the value of RMS is $4.85206e+09$ for the straight solid line, and $a = 0.957735 \pm 843.9$, $b = 0.988369 \pm 5095$, $c = 1.14625 \pm 123.6$, and $d = 1.0004 \pm 5.792e + 07$, and the value of RMS is $1.66635e+07$ for the dotted line.

7 Conclusion

We proposed an algorithm suitable for parallel computing environments to reduce the lattice basis and presented its results in the SVP Challenge.

Regarding the results in the SVP Challenge, we discovered solutions to problem instances of dimensions 134, 138, 140, 142, 144, 146, 148, and 150, a world record for the highest dimension. This is clear experimental evidence that our proposal is effective at reducing the lattice basis and solving the SVP by using a parallel computing environment.

Acknowledgements. A part of this work is supported by JST CREST grant number JPMJCR1688.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) [25], pp. 553–572. https://doi.org/10.1007/978-3-642-13190-5_28
2. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) [35], pp. 21–40. https://doi.org/10.1007/978-3-642-25385-0_2
3. Ajtai, M.: The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In: Vitter, J.S. (ed.) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, 23–26 May 1998, pp. 10–19. ACM (1998). <https://doi.org/10.1145/276698.276705>
4. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Leighton, F.T., Shor, P.W. (eds.) Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, 4–6 May 1997, pp. 284–293. ACM (1997). <https://doi.org/10.1145/258533.258604>
5. Aono, Y., Nguyen, P.Q.: Random sampling revisited: lattice enumeration with discrete pruning. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 65–102. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_3
6. Aono, Y., Wang, Y., Hayashi, T., Takagi, T.: Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 789–819. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_30
7. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, 8–10 January 2012, pp. 309–325. ACM (2012). <https://doi.org/10.1145/2090236.2090262>
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. TOCT **6**(3), 13:1–13:36 (2014). <http://doi.acm.org/10.1145/2633600>

10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 97–106. IEEE Computer Society (2011). <https://doi.org/10.1109/FOCS.2011.12>
11. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* **43**(2), 831–871 (2014). <http://dx.doi.org/10.1137/120868669>
12. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Naor, M. (ed.) Innovations in Theoretical Computer Science, ITCS 2014, Princeton, NJ, USA, 12–14 January 2014, pp. 1–12. ACM (2014). <https://doi.org/10.1145/2554797.2554799>
13. Buchmann, J., Ludwig, C.: Practical lattice basis sampling reduction. *Cryptology ePrint Archive, Report 2005/072* (2005). <https://eprint.iacr.org/2005/072>
14. Buchmann, J.A., Ludwig, C.: Practical lattice basis sampling reduction. In: Hess, F., Pauli, S., Pohst, M.E. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 222–237. Springer, Heidelberg (2006). https://doi.org/10.1007/11792086_17
15. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.), [35], pp. 1–20. https://doi.org/10.1007/978-3-642-25385-0_1
16. Dagdelen, Ö., Schneider, M.: Parallel enumeration of shortest lattice vectors. In: D’Ambra, P., Guarracino, M.R., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6272, pp. 211–222. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15291-7_21
17. Detrey, J., Hanrot, G., Pujol, X., Stehlé, D.: Accelerating lattice reduction with FPGAs. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 124–143. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_8
18. Dwork, C. (ed.): Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, 17–20 May 2008. ACM (2008)
19. Fischlin, M., Coron, J.-S. (eds.): EUROCRYPT 2016. LNCS, vol. 9665. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-49890-3>
20. Fukase, M., Kashiwabara, K.: An accelerated algorithm for solving SVP based on statistical analysis. *JIP* **23**(1), 67–80 (2015). <https://doi.org/10.2197/ipsjjip.23.67>
21. Gama, N., Nguyen, P.Q.: Finding short lattice vectors within mordell’s inequality. In: Dwork, C. (ed.) [18], pp. 207–216. <https://doi.org/10.1145/1374376.1374408>
22. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_3
23. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.), [25], pp. 257–278. https://doi.org/10.1007/978-3-642-13190-5_13
24. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.), [18], pp. 197–206. <https://doi.org/10.1145/1374376.1374407>
25. Gilbert, H. (ed.): EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-13190-5>
26. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052231>

27. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, CA, USA, 1–4 June 2013, pp. 545–554. ACM (2013). <https://doi.org/10.1145/2488608.2488677>
28. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. *J. ACM* **62**(6), 45 (2015). <http://doi.acm.org/10.1145/2824233>
29. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25
30. Hermans, J., Schneider, M., Buchmann, J.A., Vercauteren, F., Preneel, B.: Parallel shortest lattice vector enumeration on graphics cards. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 52–68. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12678-9_4
31. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0054868>
32. Hoffstein, J., Pipher, J., Silverman, J.H.: An Introduction to Mathematical Cryptography: Undergraduate Texts in Mathematics, 1st edn. Springer, Heidelberg (2008)
33. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Johnson, D.S., Fagin, R., Fredman, M.L., Harel, D., Karp, R.M., Lynch, N.A., Papadimitriou, C.H., Rivest, R.L., Ruzzo, W.L., Seiferas, J.I. (eds.) Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25–27 April 1983, Boston, Massachusetts, USA, pp. 193–206. ACM (1983). <https://doi.org/10.1145/800061.808749>
34. Kuo, P., Schneider, M., Dagdelen, Ö., Reichelt, J., Buchmann, J.A., Cheng, C., Yang, B.: Extreme enumeration on GPU and in clouds - How many dollars you need to break SVP challenges. In: Preneel, B., Takagi, T. (eds.) [43], pp. 176–191. https://doi.org/10.1007/978-3-642-23951-9_12
35. Lee, D.H., Wang, X. (eds.): ASIACRYPT 2011. LNCS, vol. 7073. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-25385-0>
36. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**(4), 515–534 (1982). <http://dx.doi.org/10.1007/BF01457454>
37. Ludwig, C.: Practical lattice basis sampling reduction. Ph.D. thesis, Technische Universität Darmstadt Universität (2005). <http://elib.tu-darmstadt.de/diss/000640>
38. Micciancio, D., Walter, M.: Fast lattice point enumeration with minimal overhead. In: Indyk, P. (ed.) Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, 4–6 January 2015, pp. 276–294. SIAM (2015). <https://doi.org/10.1137/1.9781611973730.21>
39. Micciancio, D., Walter, M.: Practical, predictable lattice basis reduction. In: Fischlin, M., Coron, J. (eds.), [19], pp. 820–849. https://doi.org/10.1007/978-3-662-49890-3_31
40. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm - Survey and Applications. Information Security and Cryptography. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-02295-1>
41. Pieterse, V., Black, P.E.: Single program multiple data. In: Dictionary of Algorithms and Data Structures, December 2004. <http://www.nist.gov/dads/HTML/singleprogrm.html>

42. Plantard, T., Schneider, M.: Creating a challenge for ideal lattices. *Cryptology ePrint Archive*, Report 2013/039 (2013). <https://eprint.iacr.org/2013/039>
43. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-23951-9>
44. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, Baltimore, MD, USA, 22–24 May 2005, pp. 84–93. ACM (2005). <https://doi.org/10.1145/1060590.1060603>
45. Schneider, M., Gama, N.: SVP Challenge. <http://www.latticechallenge.org/svp-challenge/>
46. Schneider, M., Göttert, N.: Random sampling for short lattice vectors on graphics cards. In: Preneel, B., Takagi, T. (eds.), [43], pp. 160–175. https://doi.org/10.1007/978-3-642-23951-9_11
47. Schnorr, C.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) *STACS 2003*. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36494-3_14
48. Schnorr, C.: Correction to lattice reduction by random sampling and birthday methods (2012). <http://www.math.uni-frankfurt.de/~dmst/research/papers.html>
49. Schnorr, C., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving. *Math. Program.* **66**, 181–199 (1994). <http://dx.doi.org/10.1007/BF01581144>
50. The FPLLL Development Team: FPLLL, a lattice reduction library (2016). <https://github.com/fplll/fplll>

Composable Security



Reusing Tamper-Proof Hardware in UC-Secure Protocols

Jeremias Mechler¹(✉), Jörn Müller-Quade¹, and Tobias Nilges²

¹ Karlsruhe Institute of Technology, Karlsruhe, Germany

jeremias.mechler@kit.edu

² Aarhus University, Aarhus, Denmark

Abstract. Universally composable protocols provide security even in highly complex environments like the Internet. Without setup assumptions, however, UC-secure realizations of cryptographic tasks are impossible. Tamper-proof hardware tokens, e.g. smart cards and USB tokens, can be used for this purpose. Apart from the fact that they are widely available, they are also cheap to manufacture and well understood.

Currently considered protocols, however, suffer from two major drawbacks that impede their practical realization:

- The functionality of the tokens is protocol-specific, i.e. each protocol requires a token functionality tailored to its need.
- Different protocols cannot reuse the same token even if they require the same functionality from the token, because this would render the protocols insecure in current models of tamper-proof hardware.

In this paper we address these problems. First and foremost, we propose formalizations of tamper-proof hardware as an *untrusted* and *global* setup assumption. Modeling the token as a global setup naturally allows to reuse the tokens for arbitrary protocols. Concerning a versatile token functionality we choose a simple *signature* functionality, i.e. the tokens can be instantiated with currently available signature cards. Based on this we present solutions for a large class of cryptographic tasks.

Keywords: Universal Composability · Tamper-proof hardware
Unique signatures · Global setup

1 Introduction

In 2001, Canetti [5] proposed the *Universal Composability (UC)* framework. Protocols proven secure in this framework have strong security guarantees for protocol composition, i.e. the parallel or interleaved execution of protocols. Subsequently, it was shown that it is not possible to construct protocols in this strict framework without additional assumptions [7]. Typical setup assumptions like a common reference string or a public key infrastructure assume a *trusted* setup.

T. Nilges—Supported by the European Union’s Horizon 2020 research and innovation programme under grant agreement #669255 (MPCPRO).

Katz [33] on the other hand put forward the idea that protocol parties create and exchange *untrusted* tamper-proof hardware tokens, i.e. the tokens may be programmed maliciously by the sending party.

Katz' proposal spawned a line of research that focuses mainly on the feasibility of UC-secure two-party computation. First, stateful tamper-proof hardware was considered [14, 17, 24, 26, 33, 38], then weaker models of tamper-proof hardware, where the hardware token cannot reliably keep a state, i.e. the receiver can reset the token [11, 12, 15, 18–20, 25–28, 34].

Common to all of the aforementioned results is the fact that each protocol requires a token functionality that is tailored to the protocol. From a practical point of view it seems unlikely that these tokens will ever be produced by hardware vendors, and software implementations on standard smart cards are far too inefficient. Another negative side effect of protocol-specific tokens is that users need to keep at least one token for each application, which is prohibitive in practice.

We would therefore like to be able to use widely available standard hardware for our protocols. Examples are signature cards, where the token functionality is a simple signature functionality. The signing key is securely stored inside the tamper-proof hardware, while the verification key can be requested from the card. These cards are not required to keep an internal state (the keys can be hardwired). As an alternative several works in the literature discuss bit-oblivious transfer (OT) tokens as a very simple and cheap functionality [2, 26, 31]. However, there are no standardized implementations of such tokens, while signature tokens are standardized and already deployed.

As it turns out, even if there were protocols that use a signature card as a setup assumption, it would not be possible to use the same token in a different protocol. This is due to the current definitions of tamper-proof hardware in the UC model. To the best of our knowledge, reusing tamper-proof hardware was only considered by Hofheinz et al. [29], who introduce the concept of *catalysts*. In their model, they show that the setup can be used for multiple protocols, unlike a normal UC setup, but they assume a *trusted* setup.

A recent line of research, e.g. [4, 8, 10], has focused on efficient protocols based on a *globally available setup*. This stronger notion of UC security, called *Generalized UC (GUC)*, was introduced by Canetti et al. [6] and captures the fact that protocols are often more efficient if they can use the same setup. Indeed, a globally available token in the sense of GUC would naturally allow different protocols to use the same token. We note that the work of Chandran et al. [11] and subsequent works following the approach of requiring only black-box access to the token during simulation (e.g. [12, 26]) might in principle be suitable for reuse, however none of these works consider this scenario and the problem of highly protocol-specific token functionalities is prevalent in all of these works.

1.1 Our Contribution

We apply the GUC methodology to resettable tamper-proof hardware and define the first global setup that is *untrusted*, in contrast to *trusted and incorruptible*

setups like a global random oracle [8], key registration with knowledge [6] or trusted processors [40].

We present two models for reusable tamper-proof hardware:

- The first model is inspired by the work of [29] and generalizes their approach from trusted signature cards to generic and untrusted resettable tokens. It is also designed to model real world restrictions regarding concurrent access to e.g. a smart card. A real world analogy is an ATM that seizes the card for the duration of the cash withdrawal. During that time, the card cannot be used to sign a document. We want to highlight that this only limits the access to the tokens for a short time, it is still possible to run several protocols requiring the same token in an interleaved manner.
- The second model is a GUC definition of a resettable tamper-proof hardware token following the approach of [8], which is meant to give a GUC definition of reusable tamper-proof hardware. In particular, this means that there are no restrictions at all regarding access to the token.

We also consider a peculiarity of real world signature cards that is typically ignored in idealized descriptions. Most signature cards outsource some of the hashing of the message, which is usually needed in order to generate a signature, to the client. This is done to make the signature generation more efficient. We formally capture this in a new definition of signatures where the signing process is partitioned into a preprocessing and the actual signing. As we will show, cards that do outsource the hashing—even if only in part—cannot be used in all scenarios. Nevertheless, we show that a wide range of cryptographic functionalities can be realized, even if the card enforces preprocessing.

- UC-secure commitments in both models, even with outsourced hashing by the signature card. This means that all currently available signature cards can in principle be used with our protocols.
- UC-secure non-interactive secure computation (NISC) in the GUC model. Here it is essential that the hashing is performed on the card, i.e. not all signature cards are suitable for these protocols. This result establishes the minimal interaction required for (one-sided) two-party computation.

We show that the number of tokens sent is optimal, and that stateful tokens do not yield any advantage in the setting of globally available or reusable tokens.

1.2 Our Techniques

Modelling reusable hardware tokens. In the definition of the “realistic” model, a protocol is allowed to send a `seize` command to the token functionality, which will block all requests by other protocols to the token until it is released again via `release`. We have to make sure that messages cannot be exchanged between different protocols, thus the receiving party (of the signature, i.e. the sender of the signature card) has to choose a nonce. This nonce

has to be included in the signature, thereby binding the message to a protocol instance. This obviously requires interaction, so non-interactive primitives cannot be realized in this model.

In order to explore the full feasibility of functionalities with reusable tokens and obtain more round-efficient protocols, we therefore propose a more idealized model following [8]. The simulator is given access to all “illegitimate” queries that are made to the token, so that all queries concerning a protocol (identified by a process ID PID), even from other protocols, can be observed. Essentially, this turns the token into a full-blown GUC functionality and removes the additional interaction from the protocols.

Commitments from signature cards. Concerning our protocols in the above described models, one of the main difficulties when trying to achieve UC security with hardware tokens is to make sure that the tokens cannot behave maliciously. In our case, this would mean that we have to verify that the signature was created correctly. Usually, e.g. in [20, 29], this is done via zero-knowledge proofs of knowledge, but the generic constructions that are available are highly inefficient. Instead, similar to Choi et al. [12], we use unique signatures. Unique signatures allow verification of the signature, but they also guarantee that the signature is subliminal-free, i.e. a malicious token cannot tunnel messages through the signatures.

Based on tokens with this unique signature functionality, we construct a straight-line extractable commitment. The main idea is to send the message to the token and obtain a signature on it. The simulator can observe this message and extract it. Due to the aforementioned partitioning of the signature algorithm on current smart cards, however, the simulator might only learn a hash value, which makes extraction impossible. We thus modify this approach and make it work in our setting. Basically, we keep the intermediate values sent to the token in the execution and use them as a seed for a PRG, which can in turn be used to mask the actual message. Since the simulator observes this seed, it can extract the message. However, the token can still abort depending on the input, so we also have to use randomness extraction on the seed, otherwise the sender of the token might learn some bits of the seed.

Using the straight-line-extractable commitment as a building block, we modify the UC commitment of [8] so that it works with signature cards.

Non-interactive witness-extractable arguments. A witness-extractable argument is basically a witness-indistinguishable argument of knowledge (WIAoK) with a straight-line extraction procedure. We construct such a non-interactive witness-extractable argument for later use in non-interactive secure computation (NISC). Our approach follows roughly the construction of Pass [39], albeit not in the random oracle model. [39] modify a traditional WIAoK by replacing the commitments with straight-line extractable ones. Further, they replace the application of a hash function to the transcript (i.e. the Fiat-Shamir heuristic) with queries to a random oracle. For our construction, we can basically use our previously constructed straight-line extractable commitments, but

we also replace the queries to the random oracle by calls to the signature token, i.e. we can use the EUF-CMA security of the signature to ensure the soundness of the proof.

As hinted at above, this protocol requires the ideal model, since using a nonce would already require interaction. Also, there is a subtle technical issue when one tries to use signatures with preprocessing instead of the random oracle. In the reduction to the EUF-CMA security (where the reduction uses the signing oracle to simulate the token), it is essential that the commitments contain an (encoded) valid signature *before* they are sent to the token. However, if we use preprocessing, the preprocessed value does not provide the reduction with the commitments, which could in turn be extracted to reveal the valid signature and break the EUF-CMA security. Instead, it only obtains a useless preprocessed value, and once the reduction obtains the complete commitments via the non-interactive proof from the adversary, a valid call to the signature card on these commitments means that the adversary has a valid way to obtain a signature and the reduction does not go through. If the protocol were interactive, this would not be an issue, because we could force the adversary to first send the commitments and then provide a signature in a next step. But since the protocol is non-interactive, this does not work and we cannot use signature cards with preprocessing for this protocol. We believe this to be an interesting insight, since it highlights one of the differences in feasibility between idealized and practically available hardware.

1.3 Related Work

In an independent and concurrent work, using an analogous approach based on [8], Hazay et al. [27] recently introduced a GUC-variant of tamper-proof hardware to deal with the problem of adversarial token transfers in the multi-party case. This problem is equivalent to the problem of allowing the parties to reuse the token in different protocols without compromising security. Apart from using completely different techniques, however, [27] are only interested in the general feasibility of round-efficient protocols. In contrast, we would like to minimize the number of tokens that are sent. Additionally, [27] only consider the somewhat idealized GUC token functionality, and do not investigate a more realistic approach (cf. Sect. 3). This is an important aspect, in particular since our results indicate that some of the protocols in the idealized model cannot be realized in our more natural token model that is compatible with existing signature cards. Thus, from a more practical point of view, even the feasibility of generic 2PC is not conclusively resolved from existing results.

Table 1 gives a concise overview of our result compared with previous solutions based on resettable hardware that make black-box use of the token program in the UC security proof. Other approaches as shown in e.g. [17, 18, 20] are more efficient, but require the token code and therefore cannot be reused.

Generally, physically uncloneable functions (PUFs) also provide a fixed functionality, which has (assumed) statistical security. One could thus imagine using PUFs to realize reusable tokens. However, in the context of transferable setups

Table 1. Comparison of our result with existing solutions based on resettable hardware that technically allow reusing the tokens. All results mentioned above allow UC-secure 2PC, either directly or via generic completeness results.

	# Tokens	Rounds	Assumption	Token func.
[11]	2 (bidir.)	$\Theta(\kappa)$	eTDP	Specific for com.
[26]	$\Theta(\kappa)$ (bidir.)	$\Theta(1)$	CRHF ^a	Specific for OT
[12]	2 (bidir.)	$\Theta(1)$	VRF ^b	Specific for OT
[27]	$\Theta(\kappa^2)$ (bidir.)	$\Theta(1)$	OWF	Specific for OT
Ours (Sect. 3)	2 (bidir.)	$\Theta(1)$	Unique Sign. ^c	Generic
Ours (Sect. 4)	2 (bidir.)	$\Theta(1)$	Unique Sign./DDH ^d	Generic

^a A protocol based on OWF is also shown, but the round complexity increases to $\Theta(\kappa/\log(\kappa))$. Additionally, it was shown by Hazay et al. [27] that there is a subtle error in the proof of the protocol.

^b Verifiable random functions (VRFs) are only known from specific number-theoretic assumptions [32, 35, 37]. They also present a protocol with similar properties based on a CRHF, but the number of OTs is bounded in this case.

^cUnique signatures are only known from specific number-theoretic assumptions and closely related to VRFs. These are required for our protocols.

^dDDH is necessary for the NISC protocol.

(i.e. setups that do not disclose whether they have been passed on), Boureau et al. [4] show that neither OT nor key exchange can be realized, and PUFs fall into the category of transferable setups. Tamper-proof hardware as defined in this paper on the other hand is not a transferable setup according to their definitions, so their impossibilities do not apply.

2 Preliminaries

2.1 UC Framework

We show our results in the generalized UC framework (GUC) of Canetti et al. [6]. Let us first briefly describe the basic UC framework [5], and then highlight the changes required for GUC. In UC, the security of a *real* protocol π is shown by comparing it to an *ideal* functionality \mathcal{F} . The ideal functionality is incorruptible and secure by definition. The protocol π is said to realize \mathcal{F} , if for any adversary \mathcal{A} in the real protocol, there exists a simulator \mathcal{S} in the ideal model that mimics the behavior of \mathcal{A} in such a way that any environment \mathcal{Z} , which is plugged either to the ideal or the real model, cannot distinguish both.

In UC, the environment \mathcal{Z} cannot run several protocols that share a state, e.g. via the same setup. In GUC, this restriction is removed. In particular, \mathcal{Z} can query the setup independently of the current protocol execution, i.e. the simulator will not observe this query.

We will realize a UC-secure commitment. The ideal functionality \mathcal{F}_{COM} is defined in Fig. 1.

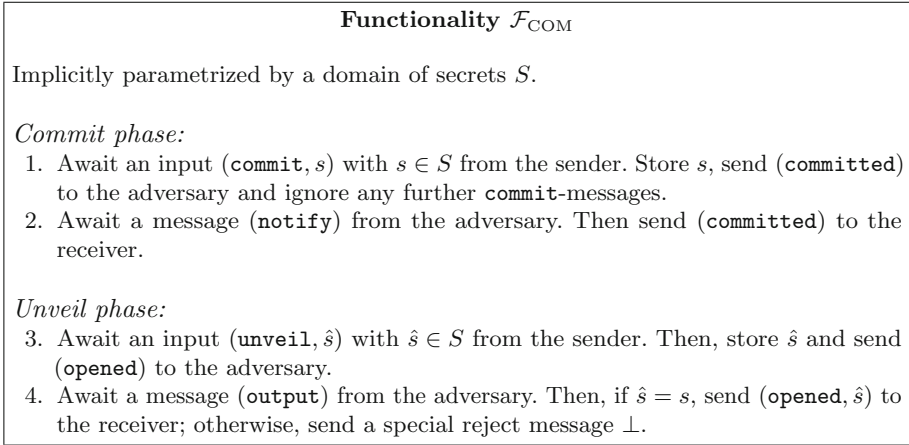


Fig. 1. Ideal functionality for commitments.

2.2 Commitments

We need several types of commitment schemes. A commitment is a (possibly interactive) protocol between two parties and consists of two phases. In the commit phase, the sender commits to a value and sends the commitment to the receiver. The receiver must not learn the underlying value before the unveil phase, where the sender sends the unveil information to the receiver. The receiver can check the correctness of the commitment. A commitment must thus provide two security properties: a hiding property that prevents the receiver from extracting the input of the sender out of the commitment value, and a binding property that ensures that the sender cannot unveil a value other than the one he committed to.

Definition 1. A commitment scheme COM between a sender S and a receiver R consists of two PPT algorithms **Commit** and **Open** with the following functionality.

- **Commit** takes as input a message s and computes a commitment c and unveil information d .
- **Open** takes as input a commitment c , unveil information d and a message s and outputs a bit $b \in \{0, 1\}$.

We require the commitment scheme to be correct, i.e. for all s :

$$\text{Open}(\text{Commit}(s), d, s) = 1$$

We assume the standard notions of statistical binding and computational hiding.

Further, we need extractable commitments. Extractability is a stronger form of the binding property which states that the sender is not only bound to one input, but that there also exists an (efficient) extraction algorithm that extracts this value. Our definition of extractable commitments is derived from Pass and Wee [41].

Definition 2. We say that $\text{COM} = (\text{Commit}, \text{Open})$ is extractable, if there exists an (expected) PPT algorithm Ext that, given black-box access to any malicious PPT algorithm \mathcal{A}_S , outputs a pair (\hat{s}, τ) such that

- (simulation) τ is identically distributed to the view of \mathcal{A}_S at the end of interacting with an honest receiver R in the commit phase,
- (extraction) the probability that τ is accepting and $\hat{s} = \perp$ is negligible, and
- (binding) if $\hat{s} \neq \perp$, then it is infeasible to open τ to any value other than \hat{s} .

Extractable commitments can be constructed from any commitment scheme via additional interaction, see e.g. [23, 41]. The definition of extractable commitments implicitly allows the extractor to rewind the adversarial sender to extract the input. In some scenarios, especially in the context of concurrently secure protocols, it is necessary that the extractor can extract the input without rewinding. This is obviously impossible in the plain model, as a malicious receiver could employ the same strategy to extract the sender’s input. Thus, some form of setup (e.g. tamper-proof hardware) is necessary to obtain straight-line extractable commitments.

Definition 3. We say that $\text{COM} = (\text{Commit}, \text{Open})$ is straight-line extractable if in addition to Definition 2, the extractor does not use rewinding.

Another tool that we need is a trapdoor commitment scheme, where the sender can equivocate a commitment if he knows a trapdoor. We adapt a definition from Canetti et al. [8].

Definition 4. A trapdoor commitment scheme TCOM between a sender S and a receiver R consists of five PPT algorithms KeyGen , TVer , Commit , Equiv and Open with the following functionality.

- KeyGen takes as input the security parameter and creates a key pair (pk, sk) , where sk serves as the trapdoor.
- TVer takes as input pk and sk and outputs 1 iff sk is a valid trapdoor for pk .
- Commit takes as input a message s and computes a commitment c and unveil information d .
- Equiv takes as input the trapdoor sk , message s' , commitment c , unveil information d and outputs an unveil information d' for s' .
- Open takes as input a commitment c , unveil information d and a message s and outputs a bit $b \in \{0, 1\}$.

The algorithm Equiv has to satisfy the following condition. For every PPT algorithm \mathcal{A}_R , the following distributions are computationally indistinguishable.

- (pk, c, d, s) , where $(pk, sk) \leftarrow \mathcal{A}_R(1^\kappa)$ such that $\text{TVer}(pk, sk) = 1$ and $(c, d) \leftarrow \text{Commit}(pk, s)$
- (pk, c', d', s) , where $(pk, sk) \leftarrow \mathcal{A}_R(1^\kappa)$ such that $\text{TVer}(pk, sk) = 1$, $(c', z) \leftarrow \text{Commit}(pk, \cdot)$ and $d' \leftarrow \text{Equiv}(sk, s, c', z)$

For example, the commitment scheme by Pedersen [42] satisfies the above definition.

2.3 Witness-Indistinguishability

We construct a witness-indistinguishable argument of knowledge in this paper.

Definition 5. A witness-indistinguishable argument of knowledge system for a language $\mathcal{L} \in \mathcal{NP}$ with witness relation $\mathcal{R}_{\mathcal{L}}$ consists of a pair of PPT algorithms (P, V) such that the following conditions hold.

- *Completeness:* For every $(x, w) \in \mathcal{R}_{\mathcal{L}}$,

$$\Pr[\langle P(w), V \rangle(x) = 1] = 1.$$

- *Soundness:* For every $x \notin \mathcal{L}$ and every malicious PPT prover P^* ,

$$\Pr[\langle P^*, V \rangle(x) = 1] \leq \text{negl}(|x|).$$

- *Witness-indistinguishability:* For every $w_1 \neq w_2$ such that $(x, w_1) \in \mathcal{R}_{\mathcal{L}}$, $(x, w_2) \in \mathcal{R}_{\mathcal{L}}$ and every PPT verifier V^* , the distributions $\{\langle P(w_1), V^* \rangle(x)\}$ and $\{\langle P(w_2), V^* \rangle(x)\}$ are computationally indistinguishable.
- *Proof of Knowledge:* There exists an (expected) PPT algorithm Ext such that for every $x \in \mathcal{L}$ and every PPT algorithm P^* , there exists a negligible function $\nu(\kappa)$ such that $\Pr[\text{Ext}(x, P^*) \in w_{\mathcal{L}}(x)] > \Pr[\langle P^*, V \rangle(x) = 1] - \nu(\kappa)$.

Witness-indistinguishable arguments/proofs of knowledge are also sometimes referred to as *witness-extractable*. Similar to the case of extractable commitments, one can also require the extractor to be straight-line, i.e. the extractor may not rewind the prover. Again, this requires an additional setup assumption and is not possible in the plain model.

Definition 6. We say that a witness-indistinguishable argument/proof system is straight-line witness-extractable if in addition to Definition 5, the extractor does not use rewinding.

2.4 Digital Signatures

A property of some digital signature schemes is the uniqueness of the signatures. Our definition is taken from Lysyanskaya [35]. Such schemes are known only from specific number theoretic assumptions.

Definition 7. A digital signature scheme SIG consists of three PPT algorithms KeyGen , Sign and Verify .

- $\text{KeyGen}(1^\kappa)$ takes as input the security parameter κ and generates a key pair consisting of a verification key vk and a signature key sgk .
- $\text{Sign}(\text{sgk}, m)$ takes as input a signature key sgk and a message m , and outputs a signature σ on m .
- $\text{Verify}(\text{vk}, m, \sigma)$ takes as input a verification key vk , a message m and a presumed signature σ on this message. It outputs 1 if the signature is correct and 0 otherwise.

We require correctness, i.e. for all m and $(\text{vk}, \text{sgk}) \leftarrow \text{KeyGen}(1^\kappa)$:

$$\text{Verify}(\text{vk}, m, \text{Sign}(\text{sgk}, m)) = 1.$$

A signature scheme is called *unique* if the following property holds: There exists no tuple $(\text{vk}, m, \sigma_1, \sigma_2)$ such that $\text{SIG.Verify}(\text{vk}, m, \sigma_1) = 1$ and $\text{SIG.Verify}(\text{vk}, m, \sigma_2) = 1$ with $\sigma_1 \neq \sigma_2$.

We point out that in the above definition, vk , σ_1 , and σ_2 need not be created honestly by the respective algorithms, but may be arbitrary strings.

3 Real Signature Tokens

It is our objective to instantiate the token functionality with a signature scheme. In order to allow currently available signature tokens to be used with our protocol, our formalization of a generalized signature scheme must take the peculiarities of real tokens into account.

One of the most important aspects regarding signature tokens is the fact that most tokens split the actual signing process into two parts: the first step is a (deterministic) preprocessing that usually computes a digest of the message. To improve efficiency, some tokens require this step to be done on the host system, at least in part. In a second step, this digest is signed on the token using the encapsulated signing key. In our case, this means that the *adversary contributes to computing the signature*. This has severe implications regarding the extraction in UC-secure protocols, because it is usually assumed that the simulator can extract the input from observing the query to the token.

To illustrate the problem, imagine a signature token that executes textbook RSA, and requires the host to compute the hash. A malicious host can *blind* his real input due to the homomorphic properties of RSA. Let (e, N) be the verification key and d the signature key for the RSA function. The adversary chooses a message m and computes the hash value $h(m)$ under the hash function h . Instead of sending $h(m)$ directly to the signature token, he chooses a random r , computes $h(m)' = h(m) \cdot r^e \bmod N$ and sends $h(m)'$ to the token. The signature token computes $\sigma' = (h(m) \cdot r^e)^d = h(m)^d \cdot r \bmod N$ and sends it to the adversary, who can multiply σ' by r^{-1} and obtain a valid signature σ on m . Obviously, demanding EUF-CMA for the signature scheme is not enough, because the signature is valid and the simulator is not able to extract m .

The protocols of [29] will be rendered insecure if the tokens perform *any kind* of preprocessing outside of the token, so the protocols cannot be realized

with most of the currently available signature tokens (even if they are trusted). We aim to find an exact definition of the requirements, so that tokens which outsource part of the preprocessing can still be used in protocols. The following definition of a signature scheme with preprocessing thus covers a large class of currently available signature tokens and corresponding standards.

Definition 8. (*Digital signatures with preprocessing*). A signature scheme SIG with preprocessing consists of five PPT algorithms KeyGen, PreSg, Sign, Vfy and Verify.

- KeyGen(1^κ) takes as input the security parameter κ and generates a key pair consisting of a verification key vk and a signature key sgk .
- PreSg(vk, m) takes as input the verification key vk , the message m and outputs a deterministically preprocessed message p with $|p| = n$.
- Sign(sgk, p) takes as input a signing key sgk and a preprocessed message p of fixed length n . It outputs a signature σ on the preprocessed message p .
- Vfy(vk, p, σ) takes as input a verification key vk , a preprocessed message p and a presumed signature σ on this message. It outputs 1 if the signature is correct and 0 otherwise.
- Verify(vk, m, σ) takes as input a verification key vk , a message m and a presumed signature σ on this message. It computes $p \leftarrow \text{PreSg}(\text{vk}, m)$ and then checks if $\text{Vfy}(\text{vk}, p, \sigma) = 1$. It outputs 1 if the check is passed and 0 otherwise.

We assume that the scheme is correct, i.e. it must hold for all messages m that

$$\forall \kappa \in \mathbb{N} \forall (\text{vk}, \text{sgk}) \leftarrow \text{KeyGen}(1^\kappa) : \text{Verify}(\text{vk}, m, \text{Sign}(\text{sgk}, \text{PreSg}(\text{vk}, m))) = 1.$$

Additionally, we require uniqueness according to Definition 7.

Existential unforgeability can be defined analogously to the definition for normal signature schemes. However, the EUF-CMA property has to hold for both KeyGen, Sign and Vfy and KeyGen, Sign and Verify. The PreSg algorithm is typically realized as a collision-resistant hash function.

All protocols in the following sections can be instantiated with currently available signature tokens that adhere the definition above. Tokens that completely outsource the computation of the message digest to the host do not satisfy this definition (because KeyGen, Sign and Vfy are not EUF-CMA secure).

The full version of this paper [36] contains an analysis for generic pre-processings, in the following we assume for simplicity that PreSg is the identity function.

3.1 Model

Our definition of reusable resettable tamper-proof hardware is defined analogously to normal resettable tamper-proof hardware tokens as in [20, 26], but we add a mechanism that allows a protocol party to seize the hardware token. This approach is inspired by the work of Hofheinz et al. [29], with the difference that

we consider untrusted tokens instead of a trusted signature token. While the token is seized, no other (sub-)protocol can use it. An adversarial sender can still store a malicious functionality in the wrapper, and an adversarial receiver is allowed to reset the program. The formal description of the wrapper $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ is given in Fig. 2.

We assume that the token receiver can verify that it obtained the correct token, e.g. by requesting some token identifier from the sender.

For completeness, we add the definition of a catalyst introduced by Hofheinz et al. [29].

Definition 9. *Let Π be a protocol realising the functionalities \mathcal{F} and \mathcal{C} in the \mathcal{C} -hybrid model. We say that \mathcal{C} is used as a catalyst if Π realises \mathcal{C} by simply relaying all requests and the respective answers directly to the ideal functionality \mathcal{C} .*

Functionality $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$

Implicitly parametrized by a security parameter κ .

Creation:

1. Await an input (**create**, M, t) from the token issuer, where M is a deterministic Turing machine and $t \in \mathbb{N}$. Store (M, t) and send (**created**) to the adversary. Ignore all further **create**-messages.
2. Await a message (**delivery**) from the adversary. Then, send (**ready**) to the token receiver.

Execution:

3. Await an input (**run**, w, sid) from the receiver. If no **create**-message has been sent, return a special symbol \perp . Otherwise, if $seized = sid$, run M on w from its most recent state. When M halts without generating output or t steps have passed, send \perp to the receiver; otherwise store the current state of M and send the output of M to the receiver.
4. Await an input (**seize**, sid) from the receiver. If $seized = \perp$, set $seized = sid$.
5. Await an input (**release**) from the receiver. Set $seized = \perp$.

Reset (adversarial receiver only):

6. Upon receiving a message (**reset**) from a corrupted token receiver, reset M to its initial state.

Fig. 2. The wrapper functionality by which we model reusable resettable tamper-proof hardware. The runtime bound t is merely needed to prevent malicious token senders from providing a perpetually running program code M ; it will be omitted throughout the rest of the paper.

In other words, the environment (and therefore other protocols) have access to the catalyst \mathcal{C} while it is used in the protocol Π . In particular, this implies that the catalyst \mathcal{C} cannot be simulated for a protocol. All in all, this notion is very similar to Definition 10.

3.2 UC-Secure Commitments

Straight-Line Extractable Commitment

We need a straight-line extractable commitment scheme in the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model to achieve two-party computation. We enhance a protocol due to Hofheinz et al. [29] which assumes trusted signature tokens as a setup such that it remains secure even with maliciously created signature tokens. Towards this goal, we adapt the idea of Choi et al. [12] to use unique signatures to our scenario. This is necessary, because verifying the functionality of an untrusted token is difficult. A unique signature scheme allows this verification very efficiently (compared to other measures such as typically inefficient ZK proofs). Additionally, it prevents the token from channeling information to the receiver of the signatures via subliminal channels.

Our protocol proceeds as follows. As a global setup, we assume that the commitment receiver has created a token containing a digital signature functionality, i.e. basically serving as a signature oracle. In a first step, the commitment receiver sends a nonce N to the sender such that the sender cannot use messages from other protocols involving the hardware token. The sender then draws a random value x . It ignores the precomputation step and sets the result p_x of this step to be x concatenated with the nonce N . The value p_x is sent to the token, which returns a signature. From the value p_x the sender derives the randomness for a one-time pad otp and randomness r for a commitment. Using r the sender commits to x , which will allow the extractor to verify if he correctly extracted the commitment. The sender also commits to the signature, x and N in a separate extractable commitment. To commit to the actual input s , the sender uses otp . Care has to be taken because a maliciously programmed signature card might leak some information about p_x to the receiver. Thus, the sender applies a 2-universal hash function before using it and sends all commitments and the blinded message to the receiver. To unveil, the sender has to send its inputs and random coins to the receiver, who can then validate the correctness of the commitments. A formal description of the protocol is shown in Fig. 3. We abuse the notation in that we define $(c, d) \leftarrow \text{COM.Commit}(x)$ to denote that the commitment c was created with randomness d .

Theorem 1. *The protocol $\Pi_{\text{COM}}^{\text{se}}$ in Fig. 3 is a straight-line extractable commitment scheme as per Definition 3 in the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model, given that unique signatures exist, using $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ as a catalyst.*

Very briefly, extractability follows from the fact that the extractor can see all messages that were sent to the token, including the seed for the PRG that allows to extract the commitments c_s and c_x . Therefore, the extractor can just search through all messages that were sent until it finds the input that matches the commitment values. Hiding follows from the hiding property of the commitments and the pseudorandomness of the PRG. The randomness extraction with the 2-universal hash function prevents the token from leaking any information that might allow a receiver to learn some parts of the randomness of the commitments.

We split the proof into two lemmata, showing the computational hiding property of $\Pi_{\text{COM}}^{\text{se}}$ in Lemma 1 and the straight-line extraction in Lemma 2.

Lemma 1. *The protocol $\Pi_{\text{COM}}^{\text{se}}$ in Fig. 3 is computationally hiding, given that COM is an extractable computationally hiding commitment scheme, f is a linear 2-universal hash function, PRG is a pseudorandom generator and SIG is an EUF-CMA-secure unique signature scheme.*

Proof. Let us consider a modified commit phase of the protocol $\Pi_{\text{COM}}^{\text{se}}$: instead of committing to the values s, x, N, σ_x , the sender S inputs random values in the

Protocol $\Pi_{\text{COM}}^{\text{se}}$

Let \mathcal{T} be an instance of $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ and PRG be a pseudorandom generator. Further let COM be a computationally hiding and extractable commitment scheme. Let SIG be a unique signature scheme according to Definition 8.

Global setup phase:

- Receiver: Compute $(\text{vk}, \text{sgk}) \leftarrow \text{SIG.KeyGen}(1^\kappa)$. Program a stateless token \mathbb{T} with the following functionality.
 - Upon receiving a message (vk) , return vk .
 - Upon receiving a message (sign, m) , compute $\sigma_m \leftarrow \text{SIG.Sign}(\text{sgk}, m)$ and output σ_m .
 Send $(\text{create}, \mathbb{T})$ to \mathcal{T} .
- Sender: Query \mathcal{T} with (vk) to obtain the verification key vk and check if it is a valid verification key for SIG.

Commit phase:

1. Receiver: Choose a nonce $N \leftarrow \{0, 1\}^\kappa$ uniformly at random and send it to the sender.
2. Sender: Let s be the sender’s input.
 - Draw $x \leftarrow \{0, 1\}^{3\kappa}$ uniformly at random and choose a linear 2-universal hash function f from the family of linear 2-universal hash functions $\{f_h : \{0, 1\}^{4\kappa} \rightarrow \{0, 1\}^\kappa\}_{h \leftarrow \mathcal{H}}$.
 - Send (seize) to \mathcal{T} . Set $p_x = x||N$ and send (sign, p_x) to \mathcal{T} to obtain σ_x . Abort if $\text{SIG.Vfy}(\text{vk}, p_x, \sigma_x) \neq 1$.
 - Derive $(\text{otp}, r) \leftarrow \text{PRG}(f(p_x))$ with $|\text{otp}| = |s|$ and compute $c_s = s \oplus \text{otp}$, $(c_x, r) \leftarrow \text{COM.Commit}(p_x)$ and $(c_\sigma, d_\sigma) \leftarrow \text{COM.Commit}(\sigma_x, x, N)$.
 - Send (c_s, c_x, c_σ, f) to the receiver. Release \mathcal{T} by sending (release) .

Unveil phase:

3. Sender: Send $(s, x, \sigma_x, d_\sigma)$ to the receiver.
4. Receiver: Set $p_x = x||N$ and compute $(\text{otp}, r) \leftarrow \text{PRG}(f(p_x))$. Check if $\text{SIG.Vfy}(\text{vk}, p_x, \sigma_x) = 1$, $\text{COM.Open}(c_x, r, x) = 1$, $\text{COM.Open}(c_\sigma, d_\sigma, (\sigma_x, x, N)) = 1$ and $c_s = s \oplus \text{otp}$. If not, abort; otherwise accept.

Fig. 3. Computationally secure straight-line extractable commitment scheme in the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model.

commitments and replaces the generated pseudorandom string by a completely random string. Thus no information about the actual input remains. In the following, we will show that from the receiver's point of view, the real protocol and the modified protocol as described above are computationally indistinguishable. This implies that the commit phase of the protocol $\Pi_{\text{COM}}^{\text{se}}$ is computationally hiding. Consider the following series of hybrid experiments.

Experiment 0: The real protocol $\Pi_{\text{COM}}^{\text{se}}$.

Experiment 1: Identical to Experiment 0, except that instead of computing $(\text{otp}, r) \leftarrow \text{PRG}(f(p_x))$, draw a uniformly at random and compute $(\text{otp}, r) \leftarrow \text{PRG}(a)$.

Experiment 2: Identical to Experiment 1, except that instead of using $\text{PRG}(a)$ to obtain otp and r , S draws otp and r uniformly at random.

Experiment 3: Identical to Experiment 2, except that instead of using COM to commit to (σ_x, x, N) , S commits to a random string of the same length.

Experiment 4: Identical to Experiment 3, except that instead of using COM to commit to p_x with randomness r , S commits to a random string of the same length. This is the ideal protocol.

Experiments 0 and 1 are statistically close, given that f is a linear 2-universal hash function and SIG is unique. A malicious receiver \mathcal{A}_R provides a maliciously programmed token T^* which might help distinguish the two experiments. In particular, the token might hold a state and it could try to communicate with \mathcal{A}_R via two communication channels:

1. T^* can try to hide messages in the signatures.
2. T^* can abort depending on the input of S .

The first case is prevented by using a unique signature scheme. The sender S asks T^* for a verification key vk^* and can verify that this key has the correct form for the assumed signature scheme. Then the uniqueness property of the signature scheme states that each message has a unique signature. Furthermore, there exist no other verification keys such that a message has two different signatures. It was shown in [3] that unique signatures imply subliminal-free signatures. Summarized, given an adversary \mathcal{A}_R that can hide messages in the signatures, we can use this adversary to construct another adversary that can break the uniqueness property of the signature scheme.

The second case is a bit more involved. The main idea is to show that applying a 2-universal hash function to p_x generates a uniformly distributed value, even if R has some information about p_x . Since x is drawn uniformly at random from $\{0, 1\}^{3\kappa}$, T^* can only abort depending on a logarithmic part of the input. Otherwise, the probability for the event that T^* aborts becomes negligible in κ (because the leakage function is fixed once the token is sent). Let X be the random variable describing inputs into the signature token and let Y describe the random variable representing the leakage. In the protocol, we apply $f \in \{f_h : \{0, 1\}^{4\kappa} \rightarrow \{0, 1\}^\kappa\}_{h \leftarrow \mathcal{H}}$ to X , which has at least min-entropy 3κ , ignoring the nonce N . Y has at most 2 possible outcomes, abort or proceed. Thus, [16] gives a lower bound for the average min-entropy of X given Y , namely

$$\tilde{H}_\infty(X|Y) \geq H_\infty(X) - H_\infty(Y) = 3\kappa - 1.$$

Note that f is chosen *after* \mathbf{R}^* sent the token. This means that we can apply the Generalized Leftover Hash Lemma (cf. [16]):

$$\Delta((f_{\mathcal{H}}(X), H, Y); (U_k, H, Y)) \leq \frac{1}{2} \sqrt{2^{\tilde{H}_\infty(X|Y)} 2^\kappa} \leq \frac{1}{2} \sqrt{2^{-(3\kappa-1)+\kappa}} \leq 2^{-\kappa}$$

We conclude that from \mathcal{A}_R 's view, $f(x)$ is distributed uniformly over $\{0, 1\}^\kappa$ and thus Experiment 0 and Experiment 1 are statistically indistinguishable. We will only sketch the rest of the proof.

Computational indistinguishability of Experiments 1 and 2 follows directly from the pseudorandomness of PRG, i.e. given a receiver \mathbf{R}^* that distinguishes both experiments, we can use this receiver to construct an adversary that distinguishes random from pseudorandom values. Experiment 2 and Experiment 3 are computationally indistinguishable given that COM is computationally hiding. From a distinguishing receiver \mathbf{R}^* we can directly construct an adversary that breaks the hiding property of the commitment scheme. And by the exact same argumentation, Experiments 3 and 4 are computationally indistinguishable. \square

We now show the straight-line extractability of $\Pi_{\text{COM}}^{\text{se}}$.

Lemma 2. *The protocol $\Pi_{\text{COM}}^{\text{se}}$ in Fig. 3 is straight-line extractable, given that COM is an extractable computationally hiding commitment scheme and SIG is an EUF-CMA-secure unique signature scheme.*

Proof. Consider the extraction algorithm in Fig. 4. It searches the inputs of \mathcal{A}_S into the hybrid functionality $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ for the combination of input and randomness for the commitment that is to be extracted.

Extractor Ext_{SEC}

Upon input $c^* = ((c_s^*, c_x^*, c_\sigma^*, f^*), Q)$, where Q is the set of all queries that \mathcal{A}_S sent to $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$, start the following algorithm.

1. For all $\alpha \in Q$, compute $(\text{otp}, \hat{r}) \leftarrow \text{PRG}(f^*(\alpha))$ and test if $\text{COM.Open}(c_x^*, \hat{r}, \alpha) = 1$. Otherwise, abort.
2. Let (otp, \hat{r}) be the values obtained in the previous step. Output $\hat{s} = c_s^* \oplus \text{otp}$.

Fig. 4. The extraction algorithm for the straight-line extractable commitment protocol $\Pi_{\text{COM}}^{\text{se}}$.

Let Q denote the set of inputs that \mathcal{A}_S sent to $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$. Extraction will fail only in the event that a value x^* is unveiled that has never been sent to \mathcal{T} , i.e. $p_x^* \notin Q$. We have to show that Ext_{SEC} extracts c_s^* with overwhelming probability, i.e. if the receiver accepts the commitment, an abort in Step 1 happens only with negligible probability.

Assume for the sake of contradiction that \mathcal{A}_5 causes this event with non-negligible probability $\varepsilon(\kappa)$. We will use \mathcal{A}_5 to construct an adversary \mathcal{B} that breaks the EUF-CMA security of the signature scheme SIG with non-negligible probability. Let vk be the verification key that \mathcal{B} receives from the EUF-CMA experiment. \mathcal{B} simulates $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ for \mathcal{A}_5 by returning vk upon receiving a query (vk) ; further let Q be the set of queries that \mathcal{A}_5 sends to $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$. For each query (sign, m) , \mathcal{B} forwards the message to the signature oracle of the EUF-CMA game and returns the resulting signature σ to \mathcal{A}_5 .

\mathcal{B} now simulates the interaction between \mathcal{A}_5 and R up to the point when \mathcal{A}_5 sends the message c_σ^* . The next messages between \mathcal{A}_5 and R represent the interaction between an honest receiver and a malicious commitment sender \mathcal{A}'_5 for the extractable commitment scheme COM. Thus, \mathcal{B} constructs a malicious \mathcal{A}'_5 from the state of \mathcal{A}_5 , which interacts with an external commitment receiver.

Due to the extractability of COM, there exists an extractor Ext that on input $(c_\sigma^*, \mathcal{A}'_5)$ outputs a message $(\hat{\sigma}_x, \hat{x}, \hat{N})$ except with negligible probability $\nu(\kappa)$. \mathcal{B} runs Ext , sets $\hat{p}_x = \hat{x} \parallel \hat{N}$ and outputs $(\hat{\sigma}_x, \hat{p}_x)$ to the EUF-CMA experiment and terminates.

From \mathcal{A}_5 's view, the above simulation is distributed identically to the real protocol conditioned on the event that the unveil of the commitment c_σ succeeds. By assumption, \mathcal{A}_5 succeeds in committing to a signature with non-negligible probability $\varepsilon(\kappa)$ in this case. It follows that the extractor Ext of COM will output a message $(\hat{\sigma}_x, \hat{x}, \hat{N})$ with non-negligible probability $\varepsilon(\kappa) - \nu(\kappa)$. Thus \mathcal{B} will output a valid signature $\hat{\sigma}_x$ for a value \hat{p}_x with non-negligible probability. However, it did not query the signature oracle on this value, which implies breaking the EUF-CMA security of the signature scheme SIG.

Thus, the extractor Ext_{SEC} will correctly output the value s with overwhelming probability. \square

Obtaining UC-Secure Commitments

In order to achieve computationally secure two-party computation, we want to transform the straight-line extractable commitment from Sect. 3.2 into a UC-secure commitment. A UC-secure commitment can be used to create a UC-secure CRS via a coin-toss (e.g. [20]). General feasibility results, e.g. [9], then imply two-party computation from this CRS.

One possibility to obtain a UC-secure commitment from our straight-line extractable commitment is to use the compiler of Damgård and Scafuro [15], which transforms any straight-line extractable commitment into a UC-secure commitment. The compiler provides an information-theoretic transformation, but this comes at the cost of requiring $O(\kappa)$ straight-line extractable commitments to commit to one bit only. If we use a signature token, this translates to many calls to the signature token and makes the protocol rather inefficient.

Instead, we adapt the UC commitment protocol of [8] to our model. The key insight in their protocol is that trapdoor extraction is sufficient to realize a UC-secure commitment. They propose to use a trapdoor commitment in conjunction with straight-line extractable commitments via a global random oracle to realize a UC-secure commitment. If we wanted to replace their commitments

with our construction, we would encounter a subtle problem that we want to discuss here. In their compiler, the commitment sender first commits to his input via the trapdoor commitment scheme. Then, he queries the random oracle with his input (which is more or less equivalent to a straight-line extractable commitment) and the unveil information for the trapdoor commitment. In the security proof against a corrupted sender, the simulator has to extract the trapdoor commitment. Thus, in their case, the simulator just searches all queries to the random oracle for the correct unveil information. In our very strict model, if we replace the oracle call with our straight-line extractable commitments, this approach fails. At first sight, it seems possible to just use the extractor for the straight-line extractable commitment to learn the value. However, it is crucial for the proof of security against a corrupted receiver that the commitment value is never published. Without this value, however, the extraction procedure will not work. Further, while we can still see all queries that are made to the hardware token, the simulator does not (necessarily) learn the complete input, but rather a precomputed value for the signature. Therefore, a little more work is necessary in order to realize a UC-secure commitment in our model.

In essence, we can use the techniques of the straight-line extractable commitment from the previous section, although we have to enhance it at several points. First, we need to query the signature token twice, for both x and r , instead of deriving r from x via a PRG. This is necessary because all protocol steps have to be invertible in order to equivocate the commitment, and finding a preimage for a PRG is not efficiently possible. Second, we have to replace the extractable commitments by *extractable trapdoor* commitments¹.

The protocol proceeds as follows: First, the receiver chooses a trapdoor for the trapdoor commitment TCOM_{ext} and commits to it via a straight-line extractable commitment. This ensures that the simulator against a corrupted receiver can extract the trapdoor and then equivocate the commitments of TCOM_{ext} . The sender then commits with TCOM_{ext} to his input (in a similar fashion as in the straight-line extractable commitment) and uses the token to sign the unveil information. Against a corrupted sender, the simulator can thus extract the unveil information and thereby extract the commitment. The commitment is sent to the receiver, which concludes the commit phase. To unveil, the sender first commits to the unveil information of TCOM_{ext} such that he cannot change his commitment when the receiver unveils the trapdoor in the next step. From there, the commitments are checked for validity and if everything checks out, the commitment is accepted. The formal description of our protocol is given in Fig. 5.

Theorem 2. *The protocol Π_{COM} in Fig. 5 computationally UC-realizes \mathcal{F}_{COM} (cf. Sect. 2.1) in the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model, using $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ as a catalyst, given that TCOM_{ext} is an extractable trapdoor commitment, SECOM is a straight-line extractable commitment and SIG is an EUF-CMA-secure unique signature scheme.*

¹ Note that any commitment scheme can be made extractable (with rewinding) via an interactive protocol, e.g. [23, 41].

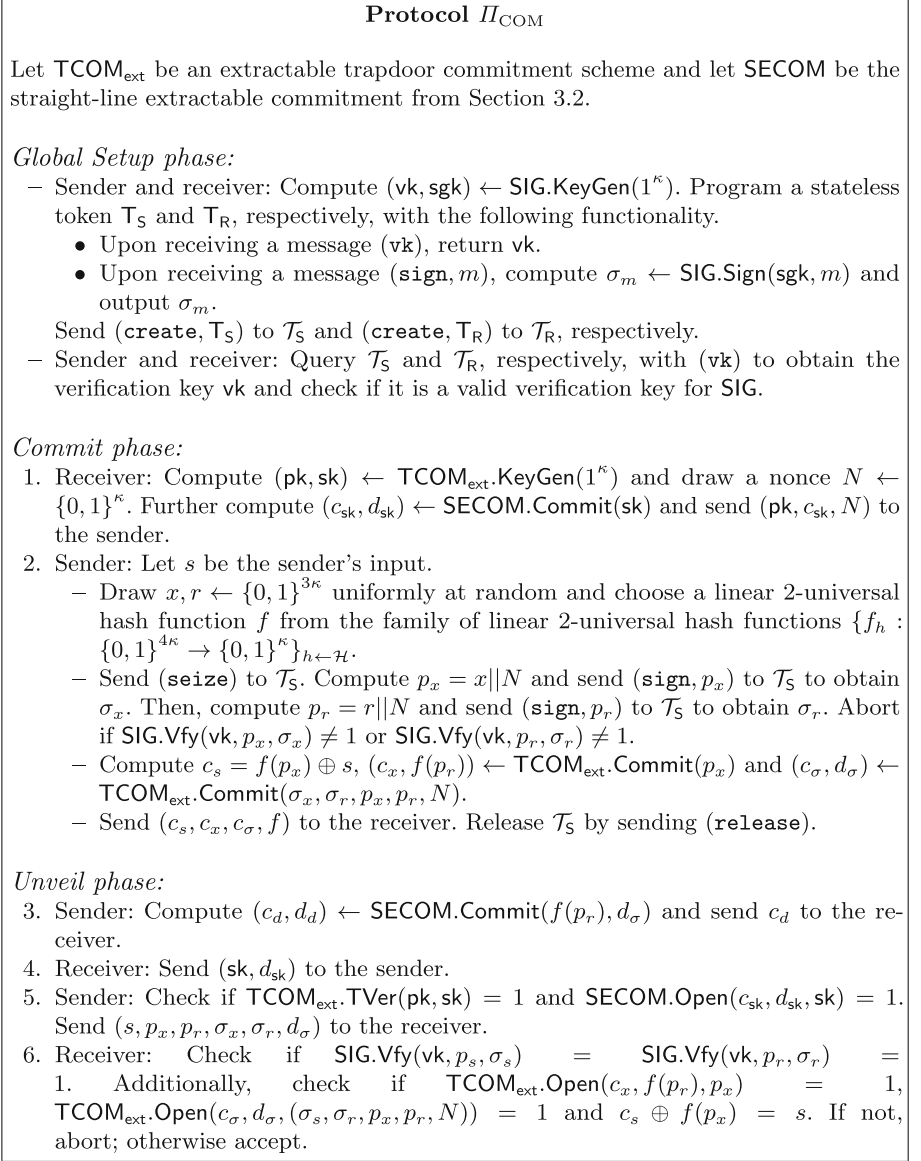


Fig. 5. Computationally UC-secure protocol realizing \mathcal{F}_{COM} in the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model.

Proof. Corrupted sender. Consider the simulator in Fig. 6. It is basically a modified version of the extraction algorithm for the straight-line extractable commitment. Against a corrupted sender, we only have to extract the input of the sender and input it into the ideal functionality.

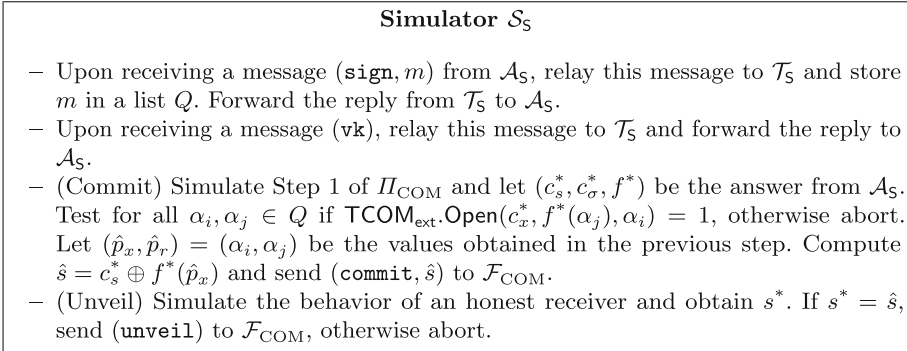


Fig. 6. Simulator against a corrupted sender in the protocol Π_{COM}

The only possibility for an environment \mathcal{Z} to distinguish $\text{Real}_{\mathcal{A}_5}^{\Pi_{\text{COM}}}$ and $\text{Ideal}_{\mathcal{S}_5}^{\mathcal{F}_{\text{COM}}}$ is the case of an abort by the simulator. However, we can adapt Lemma 2 to this scenario.

It follows that the extraction is successful with overwhelming probability and the simulation is thus indistinguishable from a real protocol run.

Corrupted receiver. The case of a corrupted receiver is more complicated. The simulator proceeds as follows. In the commit phase, he just commits to the all zero string and sends the rest of the messages according to the protocol. To equivocate the commitment, the simulator first extracts the trapdoor $\hat{\text{sk}}$ from the commitment that the receiver sent in the commit phase. He computes the image t under the 2-universal hash function f that equivocates c_s to the value \hat{s} obtained from the ideal functionality. Then, he samples a preimage \hat{p}_x of t , and uses the trapdoor $\hat{\text{sk}}$ to equivocate the commitment c_x to \hat{p}_x . Let \hat{p}_r be the new unveil information. The simulator sends both \hat{p}_x and \hat{p}_r to the token \mathcal{T}_R to obtain σ_x and σ_r . Now, the second commitment c_σ has to be equivocated to the new signatures and inputs. From there, the simulator just executes a normal protocol run with the newly generated values.

Let \mathcal{A}_R be the dummy adversary. The formal description of the simulator is given in Fig. 7.

Experiment 0: This is the real model.

Experiment 1: Identical to Experiment 0, except that \mathcal{S}_1 aborts if the extraction of $\hat{\text{sk}}$ from c_{sk}^* fails, although $\text{SECOM}.\text{Open}(c_{\text{sk}}^*, d_{\text{sk}}^*, \text{sk}^*) = 1$.

Experiment 2: Identical to Experiment 1, except that \mathcal{S}_2 uses a uniformly random value t_x instead of applying f to p_x , and computes a preimage \hat{p}_x of t_x under the linear 2-universal hash function f .

Experiment 3: Identical to Experiment 2, except that \mathcal{S}_3 computes $(c_\sigma, d_\sigma) \leftarrow \text{TCOM}_{\text{ext}}.\text{Commit}(0)$ in the commit phase. In the unveil phase, he sends $(\text{sign}, \hat{p}_x), (\text{sign}, \hat{p}_r)$ to \mathcal{T}_R . As an unveil information, he computes $\hat{d}_\sigma \leftarrow \text{TCOM}_{\text{ext}}.\text{Equiv}(\hat{\text{sk}}, (\hat{\sigma}_x, \hat{\sigma}_r, \hat{p}_x, \hat{p}_r, N), c_\sigma, d_\sigma)$.

Experiment 4: Identical to Experiment 3, except that \mathcal{S}_4 computes $(c_x, d_x) \leftarrow \text{TCOM}_{\text{ext}}.\text{Commit}(0)$ in the commit phase and then computes the unveil information $\hat{p}_r \leftarrow \text{TCOM}_{\text{ext}}.\text{Equiv}(\hat{\text{sk}}, \hat{p}_x, c_x, d_x)$. This is the ideal model.

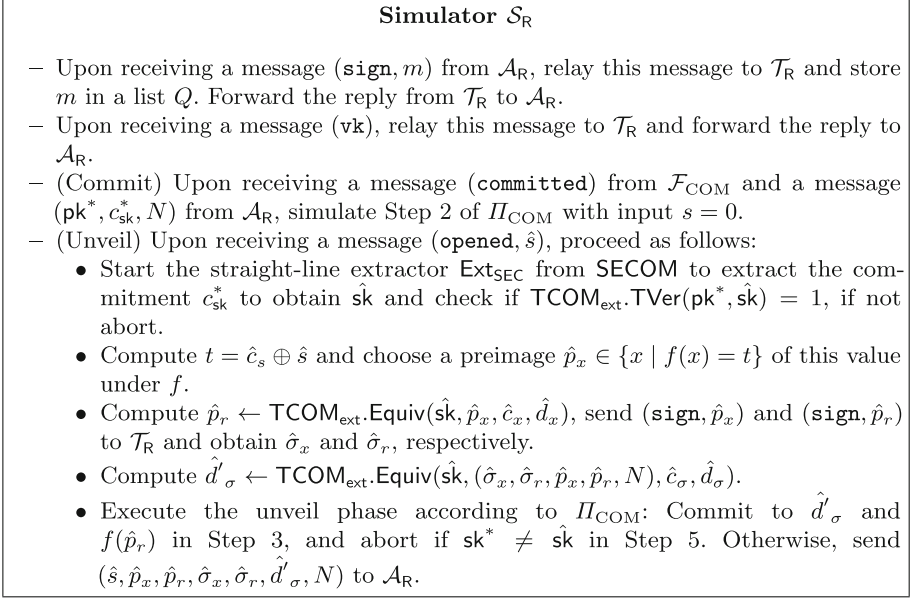


Fig. 7. Simulator against a corrupted receiver in the protocol Π_{COM}

Experiment 0 and Experiment 1 are computationally indistinguishable given that SECOM is a straight-line extractable commitment. A distinguishing environment can directly be transformed into an adversary that breaks the straight-line extraction property. Experiments 1 and 2 are statistically indistinguishable, given that f is a 2-universal hash function (the same argumentation as in Lemma 1 applies). Additionally, it is obvious that a preimage is efficiently sampleable due to the linearity of f . Experiment 2 and Experiment 3 are computationally indistinguishable, given that TCOM_{ext} is a trapdoor commitment scheme. A distinguishing environment \mathcal{Z} can straightforwardly be used to break the equivocation property of the commitment scheme. The same argumentation holds for Experiment 3 and Experiment 4. \square

4 Ideal Signature Tokens

The model considered in the previous section allows a broad class of signature algorithms that can be placed on the token. This comes with the drawback that some UC functionalities cannot be realized. In particular, non-interactive protocols are directly ruled out by the model. In this section, we want to explore what is theoretically feasible with reusable hardware tokens, at the cost of limiting the

types of signature tokens that are suitable for our scenario. Therefore, we require that the complete message that is to be signed is given to the signature token. Nevertheless, there are currently available signature cards that can be used for the protocols that are presented in this section.

4.1 Model

In contrast to $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$, we now adapt the simulation trapdoor of Canetti et al. [8] from a global random oracle to the scenario of reusable tamper-proof hardware. To overcome the problem that the simulator cannot read queries to the setup functionality outside of the current protocol, the authors require parties that query the setup to include the current session id SID of the protocol. If a malicious party queries the setup in another protocol, using the SID of the first protocol, the setup will store this query in a list and give the simulator access to this list (via the ideal functionality with which the simulator communicates). This mechanism ensures that the simulator only learns illegitimate queries, since honest parties will always use the correct SID.

We thus enhance the standard resettable wrapper functionality $\mathcal{F}_{\text{wrap}}^{\text{resettable}}$ by the query list, and parse inputs as a concatenation of actual input and the session id (cf. Fig. 8).

Compared to our previous reusable token specification $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$, it is no longer necessary to use a nonce to bind the messages to one specific protocol instance. Thus, the inherent interaction of the $\mathcal{F}_{\text{wrap}}^{\text{ru-strict}}$ -hybrid model is removed in the $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ -hybrid model. This will allow a much broader class of functionalities to be realized. For our purposes, however, we have to assume that the token learns the complete input, in contrast to the strict model. This is similar to the model assumed in [29], but in contrast to their work, we focus on untrusted tokens.

Let us briefly state why we believe that this model is still useful. On the one hand, there are signature tokens that support that the user inputs the complete message without any preprocessing. On the other hand, the messages that we input are typically rather short (linear in the security parameter), implying that the efficiency of the token is not reduced by much. Even to the contrary, this allows us to construct more round- and communication-efficient protocols, such that the overall efficiency increases.

Our security notion is as follows.

Definition 10. *Let \mathcal{F} be an ideal functionality and let Π be a protocol. We say that Π UC-realizes \mathcal{F} in the global tamper-proof hardware model if for any real PPT adversary \mathcal{A} , there exists an ideal PPT adversary \mathcal{S} such that for every PPT environment \mathcal{Z} , it holds that*

$$\text{Ideal}_{\mathcal{F}, \mathcal{S}}^{\mathcal{F}_{\text{wrap}}^{\text{ru}}}(\mathcal{Z}) \approx \text{Real}_{\Pi, \mathcal{A}}^{\mathcal{F}_{\text{wrap}}^{\text{ru}}}(\mathcal{Z})$$

Compared to the standard UC security, the setup is now available both in the real and the ideal settings.

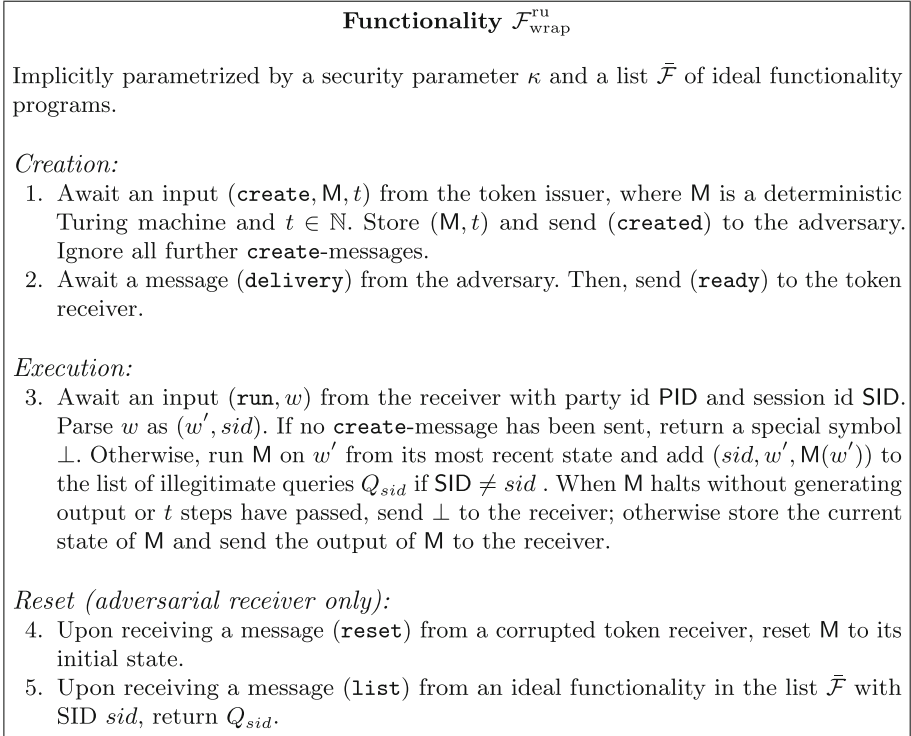


Fig. 8. The wrapper functionality by which we model reusable resettable tamper-proof hardware. The runtime bound t is merely needed to prevent malicious token senders from providing a perpetually running program code M ; it will be omitted throughout the rest of the chapter.

4.2 UC-Secure Non-Interactive Two-Party Computation

In this section, we show how to realize UC-secure non-interactive computation and the required tools. In the full version [36] we show a small modification to the straight-line extractable commitment from Sect. 3.2 such that it is non-interactive by simply removing the nonce. This is used for the construction in the next section.

Non-Interactive Straight-Line Witness-Extractable Arguments

Our protocol is based on the construction of Pass [39], who presented a protocol for a non-interactive straight-line witness-extractable proof (NIWIAoK) in the random oracle model. Let $\Pi = (\alpha, \beta, \gamma)$ be a Σ -protocol, i.e. a three message zero-knowledge proof system. We also assume that Π has special soundness, i.e. from answers γ_1, γ_2 to two distinct challenges β_1, β_2 , it is possible to reconstruct the witness that the prover used.

The main idea of his construction is as follows. Instead of performing a Σ -protocol interactively, a Fiat-Shamir transformation [22] is used to make

the protocol non-interactive. The prover computes the first message α of the Σ -protocol, selects two possible challenges β_1 and β_2 , computes the resulting answers γ_1 and γ_2 based on the witness w according to the Σ -protocol for both challenges and computes commitments c_i to the challenge/response pairs. Instead of having the verifier choose one challenge, in [22], a hash function is applied to the commitment to determine which challenge is to be used. The prover then sends (α, c) and the unveil information of the c_i to the verifier. The verifier only has to check if the unveil is correct under the hash function and if the resulting Σ -protocol transcript $(\alpha, \beta_i, \gamma_i)$ is correct. The resulting protocol only has soundness $\frac{1}{2}$ and thus has to be executed several times in parallel. [39] replaces the hash function by a random oracle and thus obtains a proof system. Further, if the commitments to (β_i, γ_i) are straight-line extractable, the resulting argument system will be witness-extractable, i.e. an argument of knowledge.

The straight-line extractable commitment $\Pi_{\text{COM}}^{\text{se}}$ from Sect. 3.2 requires interaction, so we cannot directly plug this into the protocol without losing the non-interactive nature of the argument system. But note that the first message of $\Pi_{\text{COM}}^{\text{se}}$ is simply sending a nonce, which is no longer necessary in the $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ -hybrid model. Thus, by omitting this message, $\Pi_{\text{COM}}^{\text{se}}$ becomes a valid non-interactive straight-line extractable commitment.

A formal description of the protocol complete NIWIAoK is given in Fig. 9.

Theorem 3. *The protocol Π_{NIWI} in Fig. 9 is a straight-line witness-extractable argument as per Definition 6 in the $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ -hybrid model, given that NICOM is a straight-line extractable commitment scheme and SIG is an EUF-CMA-secure unique signature scheme.*

Proof. Let Π be a public-coin special-sound honest-verifier zero-knowledge (SHVZK) protocol.

Completeness: Completeness of Π_{NIWI} follows directly from the completeness of the Σ -protocol Π .

Witness-Indistinguishability: Cramer et al. [13, 39] show that a SHVZK protocol directly implies a public-coin witness-indistinguishable protocol. Since witness-indistinguishable protocols are closed under parallel composition as shown by Feige and Shamir [21], Π_{NIWI} is witness-indistinguishable.

Extractability: Let Ext_{NIC} be the straight-line extractor of NICOM. We will construct a straight-line extractor for Π_{NIWI} (cf. Fig. 10).

It remains to show that if the verifier accepts, Ext_{NIWI} outputs a correct witness with overwhelming probability. First, note that Ext_{NIC} extracts the inputs of c^* with overwhelming probability, and by the special soundness of Π , we know that if both challenges in the commitment are extracted, Ext_{NIWI} will obtain a witness. Thus, the only possibility for Ext_{NIWI} to fail with the extraction is if a malicious PPT prover \mathcal{A}_{P} manages to convince the verifier with a witness w^* such that $(x, w^*) \notin \mathcal{R}_{\mathcal{L}}$.

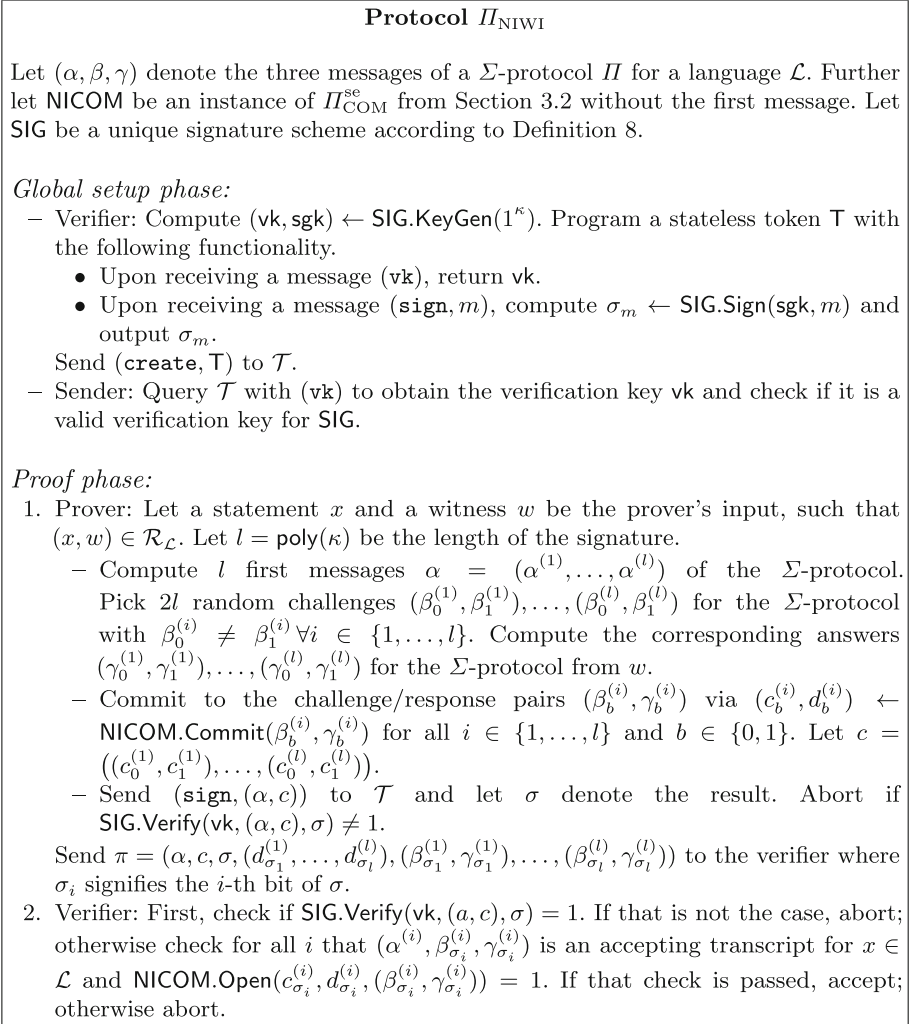


Fig. 9. Computationally secure non-interactive straight-line witness-extractable argument in the $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ -hybrid model.

Each of the l instances of Π has soundness $\frac{1}{2}$, since a malicious \mathcal{A}_{P} can only answer at most one challenge correctly, and otherwise a witness is obtained. Thus, \mathcal{A}_{P} has to make sure that in all l instances, the correctly answered challenge is selected. Assume for the sake of contradiction that \mathcal{A}_{P} manages to convince the verifier with some non-negligible probability $\varepsilon(\kappa)$ of a witness w^* such that $(x, w^*) \notin \mathcal{R}_{\mathcal{L}}$. We will construct an adversary \mathcal{B} from \mathcal{A}_{P} that breaks the EUF-CMA property of **SIG** with probability $\varepsilon(\kappa)$.

Let \mathcal{B} be the adversary for the EUF-CMA game. Let vk be the verification key that \mathcal{B} receives from the EUF-CMA game. \mathcal{B} simulates $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ to \mathcal{A}_{P} by

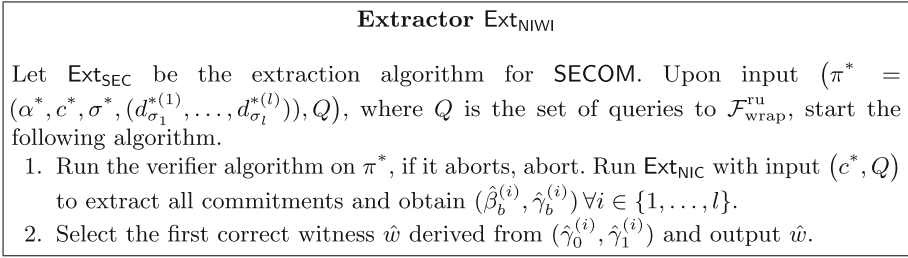


Fig. 10. The extraction algorithm for the non-interactive straight-line witness-extractable argument Π_{NIWI} .

returning vk upon receiving a query (vk) ; further let Q be the set of queries that \mathcal{A}_{P} sends to $\mathcal{F}_{\text{wrap}}^{\text{ru}}$. For each query (sign, m) , \mathcal{B} forwards the message to the signature oracle of the EUF-CMA game and returns the resulting signature σ to \mathcal{A}_{P} .

If \mathcal{B} receives a signature query of the form (sign, m^*) with $m^* = (\alpha^*, c^*)$, start the extractor Ext_{NIC} with input (c^*, Q) to extract the commitments c^* using Q . Create a signature σ^* by selecting σ_i^* as the index of the correctly evaluating challenge. The verifier will only accept if that is the case. If $\text{SIG.Verify}(\text{vk}, (\alpha^*, c^*), \sigma^*) = 1$, send (m^*, σ^*) to the EUF-CMA game, otherwise abort. We thus have that \mathcal{A}_{P} wins the EUF-CMA game with probability $\varepsilon(\kappa)$, which contradicts the EUF-CMA security of SIG. □

UC-secure NISC

Non-interactive secure computation (NISC) [30] is typically a two-party protocol. The main idea is to execute a constant round two-party computation, but reduce the number of rounds to two. In the OT-hybrid model, garbled circuits realize a non-interactive evaluation of any functionality (if the sender requires no output): the sender garbles the circuit and sends it to the receiver, who learns some labels via the OTs to evaluate the garbled circuit. It remains to realize the OT protocol with minimal interaction, and such a protocol was provided by Peikert et al. [43], taking 2 rounds of interaction given a CRS. Combining the two building blocks, a NISC protocol proceeds as follows: the receiver first encodes his input using the first message of the OT protocol and sends it to the sender. The sender in turn garbles the circuit and his inputs depending on the received message and sends the resulting garbled values to the receiver. Now the receiver can obtain some of the labels and evaluate the garbled circuit.

In order to obtain NISC, [8] build such a one-sided simulatable OT using a NIWIAoK as constructed in the previous section. The construction is black-box, i.e. we can directly replace their NIWIAoK with ours and obtain the same result. The actual NISC protocol of [8] is based on the protocol of Ashfar et al. [1]. Our modifications to the protocol are only marginal. In order for the simulation against the sender to work, the simulator must extract the seeds that the sender used to create the circuits. That is the main technical difference

between [8] and our solution. [8] have the sender send a random value to their global random oracle and use the answer as the seed. Thus, the simulator learns the seed and can extract the sender’s inputs. In our case, we let the sender choose a random value and have him send it to $\mathcal{F}_{\text{wrap}}^{\text{ru}}$ to obtain a signature, which allows us to extract this value. Due to possible leakage the sender has to apply a 2-universal hash function to the random value and use the result as the seed, but the technique (and the proof) is essentially the same as for our straight-line extractable commitment scheme. We provide a detailed protocol description in the full version [36].

5 Limitations

It is known that there exist limitations regarding the feasibility of UC-secure protocols based on resettable tamper-proof hardware, both with computational and with statistical security. Concerning statistical security, Goyal et al. [25] show that non-interactive commitments and OT cannot be realized from resettable tamper-proof hardware tokens, even with standalone security. In the computational setting, Döttling et al. [20] and Choi et al. [12] show that if (any number of) tokens are sent only in one direction, i.e. are not exchanged by both parties, it is impossible to realize UC-secure protocols without using non-black-box techniques. Intuitively, this follows from the fact that the simulator does not have any additional leverage over a malicious receiver of such a token. Thus, a successful simulator strategy could be applied by a malicious receiver as well. The above mentioned results apply to our scenario as well.

Jumping ahead, the impossibilities stated next hold for both specifications of reusable tamper-proof hardware that we present in the following. In particular, GUC and GUC-like frameworks usually impose the restriction that the simulator only has black-box access to the reusable setup. Thus, compared to the standard definition of resettable tamper-proof hardware, the model of resettable reusable tamper-proof hardware has some limitations concerning non-interactive two-party computation. The degree of non-interactivity that can be achieved with resettable hardware, i.e. just sending tokens (and possibly an additional message) to the receiver, is impossible to obtain in the model of resettable reusable hardware.

Corollary 1. *There exists no protocol Π_{PF} using any number of reusable and resettable hardware tokens $\mathcal{T}_1, \dots, \mathcal{T}_n$ issued from the sender to the receiver that computationally UC-realizes the ideal point function \mathcal{F}_{PF} .*

Proof (Sketch). This follows directly from the observation that the simulator for protocols based on reusable hardware is only allowed to have black-box access to the token, i.e. the simulator does not have access to the code of the token(s). Applying [12, 20] yields the claim.

The best we can hope for is a protocol for non-interactive two-party computation where the parties exchange two messages (including hardware tokens)

to obtain a (somewhat) non-interactive protocol. Maybe even more interesting, even stateful reusable hardware tokens will not yield any advantage compared to resettable tokens, if the tokens are only sent in one direction.

Corollary 2. *There exists no protocol Π_{OT} using any number of reusable and stateful hardware tokens $\mathcal{T}_1, \dots, \mathcal{T}_n$ issued from the sender to the receiver that statistically UC-realizes \mathcal{F}_{OT} .*

Proof (Sketch). First note, as above, that the simulator of a protocol against a token sender will not get the token code because he only has black-box access to the token. Thus the simulator cannot use rewinding during the simulation, which is the one advantage that he has over the adversary. The simulator falls back to observing the input/output behavior of the token, exactly as in the case of standard resettable hardware. Due to the impossibility of statistically secure OT based on resettable hardware [25], the claim follows.

Acknowledgement. The authors would like to thank Dirk Achenbach and Björn Kaidel for helpful discussions on an earlier version of this paper, and the anonymous reviewers for their comments. We further thank Maciej Obrenski for discussions on the entropy estimation of the signatures.

References

1. 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
2. Agrawal, S., Ananth, P., Goyal, V., Prabhakaran, M., Rosen, A.: Lower bounds in the hardware token model. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 663–687. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_28
3. Bohli, J.-M., González Vasco, M.I., Steinwandt, R.: A subliminal-free variant of ECDSA. In: Camenisch, J.L., Collberg, C.S., Johnson, N.F., Sallee, P. (eds.) IH 2006. LNCS, vol. 4437, pp. 375–387. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74124-4_25
4. Boureanu, I., Ohkubo, M., Vaudenay, S.: The limits of composable crypto with transferable setup devices. In: Bao, F., Miller, S., Zhou, J., Ahn, G.J. (eds.) ASIACCS 15, pp. 381–392. ACM Press, April 2015
5. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
6. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_4
7. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_2

8. Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 14, pp. 597–608. ACM Press, November 2014
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press, May 2002
10. Canetti, R., Shahaf, D., Vald, M.: Universally composable authentication and key-exchange with global PKI. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 265–296. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_11
11. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_31
12. Choi, S.G., Katz, J., Schröder, D., Yerukhimovich, A., Zhou, H.-S.: (Efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 638–662. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_27
13. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
14. Damgård, I., Nielsen, J.B., Wichs, D.: Universally composable multiparty computation with partially isolated parties. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 315–331. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_19
15. Damgård, I., Scafuro, A.: Unconditionally secure and universally composable commitments from physical assumptions. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 100–119. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_6
16. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **38**(1), 97–139 (2008)
17. Döttling, N., Kraschewski, D., Müller-Quade, J.: Unconditional and composable security using a single stateful tamper-proof hardware token. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 164–181. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_11
18. Döttling, N., Kraschewski, D., Müller-Quade, J., Nilges, T.: From stateful hardware to resettable hardware using symmetric assumptions. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 23–42. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26059-4_2
19. Döttling, N., Kraschewski, D., Müller-Quade, J., Nilges, T.: General statistically secure computation with bounded-resettable hardware tokens. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 319–344. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_14
20. Döttling, N., Mie, T., Müller-Quade, J., Nilges, T.: Implementing resettable UC-functionalities with untrusted tamper-proof hardware-tokens. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 642–661. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_36
21. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC, pp. 416–426. ACM Press, May 1990

22. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
23. Goldreich, O.: The Foundations of Cryptography: Basic Techniques, vol. 1. Cambridge University Press, Cambridge (2001)
24. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_3
25. Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 173–190. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_10
26. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_19
27. Hazay, C., Polychroniadou, A., Venkitasubramanian, M.: Composable security in the tamper-proof hardware model under minimal complexity. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9985, pp. 367–399. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53641-4_15
28. Hazay, C., Polychroniadou, A., Venkitasubramanian, M.: Constant round adaptively secure protocols in the tamper-proof hardware model. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 428–460. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_15
29. Hofheinz, D., Müller-Quade, J., Unruh, D.: Universally composable zero-knowledge arguments and commitments from signature cards. In: Proceedings of the 5th Central European Conference on Cryptology MoraviaCrypt 2005 (2005)
30. 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
31. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
32. Jager, T.: Verifiable random functions from weaker assumptions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 121–143. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_5
33. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_7
34. Kolesnikov, V.: Truly efficient string oblivious transfer using resettable tamper-proof tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 327–342. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_20
35. Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 597–612. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_38
36. Mechler, J., Müller-Quade, J., Nilges, T.: Universally composable (non-interactive) two-party computation from untrusted reusable hardware tokens. Cryptology ePrint Archive, Report 2016/615 (2016). <http://eprint.iacr.org/2016/615>
37. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th FOCS, pp. 120–130. IEEE Computer Society Press, October 1999

38. Moran, T., Segev, G.: David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_30
39. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_19
40. Pass, R., Shi, E., Tramèr, F.: Formal abstractions for attested execution secure processors. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 260–289. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_10
41. Pass, R., Wee, H.: Black-box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_24
42. 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
43. 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



On Composable Security for Digital Signatures

Christian Badertscher¹(✉) , Ueli Maurer¹, and Björn Tackmann²

¹ Department of Computer Science, ETH Zurich, 8092 Zürich, Switzerland
{badi,maurer}@inf.ethz.ch

² IBM Research – Zurich, 8803 Rüschlikon, Switzerland
bta@zurich.ibm.com

Abstract. A digital signature scheme (DSS), which consists of a key-generation, a signing, and a verification algorithm, is an invaluable tool in cryptography. The first and still most widely used security definition for a DSS, existential unforgeability under chosen-message attack, was introduced by Goldwasser, Micali, and Rivest in 1988.

As DSSs serve as a building block in numerous complex cryptographic protocols, a security definition that specifies the guarantees of a DSS under composition is needed. Canetti (FOCS 2001, CSFW 2004) as well as Backes, Pfitzmann, and Waidner (CCS 2003) have described ideal functionalities for signatures in their respective composable-security frameworks. While several variants of these functionalities exist, they all share that the verification key and signature values appear explicitly.

In this paper, we describe digital signature schemes from a different, more abstract perspective. Instead of modeling all aspects of a DSS in a monolithic ideal functionality, our approach characterizes a DSS as a construction of a repository for authentically reading values written by a certain party from certain assumed repositories, e.g., for transmitting verification key and signature values. This approach resolves several technical complications of previous simulation-based approaches, captures the security of signature schemes in an abstract way, and allows for modular proofs.

We show that our definition is equivalent to existential unforgeability. We then model two example applications: (1) the certification of values via a signature from a specific entity, which with public keys as values is the core functionality of public-key infrastructures, and (2) the authentication of a session between a client and a server with the help of a digitally signed assertion from an identity provider. Single-sign-on mechanisms such as SAML rely on the soundness of the latter approach.

1 Introduction

A digital signature scheme (DSS) allows a signer to authenticate a message such that everyone can verify the authenticity. The signer initially generates an asym-

B. Tackmann—Work partially done while author was at Department of Computer Science and Engineering, UC San Diego.

metric key pair consisting of a *signing key* and a *verification key*. The signing key, which is kept secret by the signer, allows to generate signatures for messages. The verification key is made public and allows to verify that a message was indeed signed using the corresponding signing key. DSSs are a crucial component in many of today’s widely-used cryptographic protocols. They underlie the public-key infrastructure (PKI) that is used to provide authentication in most Internet protocols, and they are used to authenticate e-mails as well as to provide non-repudiation for electronic documents. They are also used as a building block in numerous cryptographic protocols.

1.1 Formalizing Message Authentication

The core idea of our approach is that digitally signing a message can be understood as the signer’s declaration that the message belongs to a certain context, which is described by the verification key. This context may be the signer’s commitment to be legally liable for the content of the message (e.g., a contract), or simply that the message is meant to originate from the signer. Abstractly, this can be understood as writing the message to a certain type of *repository* that allows other parties to verify for given messages whether they have been written to the repository, i.e., assigned to the context.

The real-world/ideal-world paradigm. Many security definitions, and in particular most composable security frameworks [6, 22, 25], are based on the real-world/ideal-world paradigm. The real world models the use of a protocol, whereas the ideal world formalizes the security guarantees that the protocol is supposed to achieve. The structure of the real-world model is depicted for a simple setting in Fig. 1 on the left, where R describes the *assumed resources* [20, 22] or *hybrid functionalities* [6] used by the protocol π . The “open lines” on the left and right indicate the interfaces that the honest parties use to access the protocol π , whereas the line on the bottom indicates a potential attacker’s access to R .

In the ideal world, as depicted in Fig. 1 on the right, the box S formalizes the intended security guarantees and is referred to as *constructed resource* [22]



Fig. 1. Left: Execution of protocol π in the real-world model. **Right:** Ideal-world model described by S with simulator σ . In both figures, the dotted lines are “free” interfaces explained below.

or *ideal functionality* [6]. The access of the honest parties to S is via direct interfaces, whereas a potential attacker accesses S via the so-called *simulator* σ . To define security, one considers random experiments in which a *distinguisher* (sometimes called *environment*) is connected to all open interfaces of either of the two settings in Fig. 1. The intuition behind the simulator is that, if the two settings are indistinguishable, then any attack on π (and working with assumed resource R) can be translated via σ into an attack on S , but S is secure by definition. Therefore, using the protocol π with the assumed resource R provides at least the same security guarantees as using the constructed resource S .

Signature schemes as constructions. We formalize the security of a DSS in the real-world/ideal-world paradigm and based on different types of repositories to which messages can be written and from which messages can be read, by different parties with potentially different access permissions. As described above, the goal of using the signature scheme in the described way can be seen as constructing an *authenticated* repository, where only the signer can write messages and all verifiers can check the validity. This repository takes the role of S in Fig. 1.

Using a signature scheme for this purpose requires an authenticated repository that can hold one message. This repository is used to transmit the signature verification key. We also assume one repository that can hold multiple messages, but this repository can be *insecure*, meaning that write access to the repository is not exclusive to the signer. This repository is used to transmit the signature strings. We also make the storage of the signing key explicit as a *secure* repository where both write and read access is exclusive to the signer. These three assumed repositories correspond to R in Fig. 1.

A signature scheme then uses the described repositories in the obvious way: the signer begins by generating a key pair, writes the signing key to the secure repository and the verification key to the authenticated one. Upon a request to sign a message m , the signer retrieves the signing key from the secure repository, computes the signature, and stores it in the insecure repository. For checking the validity of a message m , a verifier reads the verification key from the authenticated repository and the signature from the insecure one, and runs the signature verification algorithm. Our security statement is, then, that this use of the signature scheme constructs the desired authenticated repository for multiple messages from the three described assumed repositories.

The advantage of such a composable security statement is that applications and higher-level protocols can be designed and analyzed using the abstraction of such a repository; in particular, no reduction proof is required since the composition theorem immediately guarantees the soundness of this approach. More technically, if a protocol π constructs S from R and protocol π' constructs T from S , then composing the two protocols leads to a construction of T from R .

Abstract communication semantics. The purpose of a repository is to model the fact that a certain message written by one party can be made accessible to a

different party in an abstract manner. Indeed, a DSS is a generic security mechanism and can be used by various applications; the definition of a DSS should abstract from the particular way in which the verification key and the signature are delivered to the verifier. For instance, a communication network used for transmission may guarantee messages to be delivered within a certain time, or an attacker may be able to eavesdrop on messages. Using a DSS—intuitively—preserves such properties of the communication network. The repositories used in this work are general enough to model various different such concrete types of transferring the values.

This generality is, more technically, achieved through a *free* interface that appears in both the real-world and the ideal-world model and that is indicated by the dotted lines in Fig. 1. In the random experiment, this interface is accessed directly by the distinguisher. The free interface is reminiscent of the environment access to the global setup functionality in the GUC model [10], but in our model each resource/functionality can have such a free interface.¹

A free interface allows the distinguisher to interact with both resources R and S directly. This results in a stronger and more general condition compared to considering the capabilities at that interface as part of the attacker’s interface and, therefore, in the ideal-world model providing them to the simulator. More intuitively, the free interface can be seen as a way for the distinguisher to enforce that certain aspects in the real and the ideal world are the same. We will use the free interface to let the distinguisher control the transmission semantics; this leaves our statements general and independent of any concrete such semantics.

In more detail, the write and read interfaces of the repository are defined to write to or read from buffers associated to the interface. The repository also has free interfaces that control the transfer of messages from write buffers to read buffers. In other words, capabilities such as writing messages to a buffer in the repository or reading messages from one are separated from the mechanisms for making messages written to the repository visible at a specific reader interface. Control over the operations governing the visibility is granted to the environment—this makes the security statements independent of specific network models. In particular, the statements imply those in which these capabilities are granted to an attacker controlling the network.

Interfaces and partitioning of capabilities. The interfaces of a resource group capabilities. Often, each interface can be seen as corresponding to one particular party in a given application scenario, which can then attach a protocol machine (or *converter* [22]) to this interface, as in Fig. 1. Yet, for a general security definition such as that of a DSS, we do not want to fix the number of possible verifiers in advance, or even prohibit that the signing key may be transmitted securely between and used by different parties. As one can always merge several interfaces and provide them to the same party, it is beneficial to

¹ The direct communication between the environment and the functionality requires a modification of the control function in UC, but does not affect the composition theorem. In most formal frameworks [17, 22, 25], no modification is necessary.

target a fine-grained partitioning of capabilities into interfaces, and therefore a fine-grained partitioning of the protocol into individual protocol machines.

For our repositories, this means that if each interface gives access to one basic operation (such as writing or reading one value), one can always subsume a certain subset of these capabilities into one interface and assign it to a single party. We achieve the most fine-grained partitioning by modeling each invocation of an algorithm of the signature scheme as a single protocol machine, and capture passing values between the machines explicitly via repositories.

Specifications. For generality or conciseness of description, it is often desirable to not fully specify a resource or functionality. For instance, a complete description of the construction would entail the behavior of the signature scheme in the case where a signature shall be verified before the verification key is delivered to the verifier. The approach generally used in the literature on UC in such cases is to delegate such details to the adversary, to model the worst possible behavior. In this work, we follow a more direct approach, and explicitly leave the behavior undefined in such cases.

Our formalization follows the concept of *specifications* by Maurer and Renner [23], which are sets of resources that, for example, fulfill a certain property. As such they are suitable to express an incomplete description of a resource, namely by considering the set of all resources that adhere to such a (partially defined) description. Maurer and Renner describe concrete types of specifications such as all resources that can be distinguished from a specific one by at most a certain advantage, or all resources that are obtained from a specific one by applying certain transformations.

We use specifications in this work to describe the behavior of a resource in environments that use the resource in a restricted way, in the sense that the inputs given to the resource satisfy certain conditions, such as that the verification key must have been delivered before messages can be verified. This alleviates the requirement of specifying the behavior of the resource for input patterns that do not occur in applications, and simplifies the description. Needless to say, this also means that for each application one has to show that the use of the resource indeed adheres to the specified conditions.

The repositories in this work. In summary, we consider specifications of repositories as described above. Repositories provide multiple interfaces, each of which allows exactly one write or read operation. A repository that allows for k write operations has k writer interfaces, and for n read operations it has n reader interfaces, and each operation can be understood as writing to or reading from one specific buffer. A write interface may allow the writer to input an arbitrary value from the message space, or, in a weaker form, it may allow the writer to only copy values from buffers at some read interfaces. A read interface may either allow to retrieve the contents of the corresponding buffer, or to input a value and check for equality with the one in the buffer.

The resource additionally provides free interfaces for transferring the contents of write buffers to read buffers. As discussed above, the access to these interfaces

for managing the visibility of messages is given to the distinguisher, not the attacker, to abstract from specific communication semantics.

All repositories in this work can be viewed as specific instances of the one described above, where different types of capabilities are provided at different parties' interfaces. For instance, a repository in which an attacker has only read-interfaces, but cannot write chosen messages, can be considered as *authenticated*, since all messages must originate from the intended writers. A repository where the attacker can also write can be considered as *insecure*, since messages obtained by honest readers could originate either from honest writers or the attacker.

1.2 Background and Previous Work

The concept of a DSS was first envisioned by Diffie and Hellman and referred to as *one-way authentication* [14]. Early instantiations of this concept were given by Rivest et al. [26] and by Lamport [18]. The provable-security treatment of DSS was initiated by Goldwasser et al. [15], who also introduced the first and still widely-used security definition called *existential unforgeability under chosen-message attack*. In this definition, a hypothetical attacker that has access to honestly computed signatures on messages of his own choice aims at creating a signature for some new message. A scheme is deemed secure if no efficient attacker can provide such a forgery with non-negligible probability.

Canetti [7] and independently Pfitzmann and Waidner [25] developed security frameworks that allow for security-preserving composition of cryptographic schemes. In these frameworks, the security of a cryptographic scheme, such as a DSS, is modeled by idealizing the algorithms and their security properties, and a concrete scheme is then proved to satisfy the idealization under certain computational assumptions. Higher-level schemes and protocols that make use of a DSS can be analyzed using the idealized version of the scheme. One main advantage of composable frameworks is that they guarantee the soundness of this approach; a higher-level protocol proven secure with respect to an idealized signature scheme will retain its security even if the idealized scheme is replaced by any concrete scheme that is proven secure. In contrast to standard reductionist proofs, this method does *not* require to prove an explicit reduction from breaking the signature scheme to breaking the higher-level protocol; this follows generically from the composition theorem. Still, even in protocol analyses within composable frameworks, existential unforgeability remains widely used, despite the existence of composable models within these formal frameworks.

The first composable notion for digital signatures has been proposed by Canetti [6, 8] via an ideal signing functionality \mathcal{F}_{SIG} . The functionality idealizes the process of binding a message m to a public key vk via an ideal signature string s . In a nutshell, when the honest sender signs a message, he receives an idealized signature string. This signature string allows any party to verify that the message has indeed been signed by the signer. \mathcal{F}_{SIG} enforces consistency and unforgeability in an ideal manner: if the honest signer has never signed a message m , no signature string leads to successful verification. Likewise, verification with a legitimately generated signature string for a message m always succeeds.

Special care has to be taken in case the signer is dishonest, in which case the above guarantees for unforgeability are generally lost. The formalization given by Backes et al. [2] in their framework follows a by and large similar approach.

Several versions of the signature functionality have been suggested in previous work [1, 6, 8, 9, 11, 12]. All these versions, however, require interaction with the ideal-model adversary for operations that correspond to local computations in any real-world scheme, such as the initial creation of the key pair or the generation of a signature. Camenisch *et al.* [5] point out that this unnatural weakness, allowing the adversary to delay operations in the idealized security guarantee, has often gone unnoticed and even lead to flaws in proofs of higher-level schemes based on signatures. As a further example, consider a signer S that has never signed a message m . If an honest party P verifies m with respect to some signature string s , the verification should fail. Yet, the adversary gets activated during any local verification request and can corrupt the signer just before providing the response. The adversary thus has complete freedom on whether to let P accept or reject the signature string s on message m . This behavior is arguably counter-intuitive and it is a property that signature schemes do not possess. The solution of Camenisch *et al.* [5] requires to modify the universal composability framework by introducing the concept of *responsive* environments and adversaries that are mandated to answer specific requests immediately to model local tasks. While Camenisch *et al.* do re-prove the composition theorem for their modified framework, such a modification of the framework has the downside of further increasing its complexity and, at least in principle, making security analyses in the original and modified frameworks incompatible.

Besides the technical difficulties in defining the signature functionality \mathcal{F}_{SIG} consistently, it is less abstract than what one would expect, since the signature string and the verification key are an explicit part of the interface. Indeed, Canetti [8, p. 5] writes:

The present formalization of \mathcal{F}_{SIG} and $\mathcal{F}_{\text{CERT}}$ is attractive in that it allows a very modular approach where each instance of the ideal functionality handles only a single instance of a signature scheme (i.e., a single pair of signature and verification keys). This has several advantages as described in this work. However, the interface of the signature scheme is somewhat less abstract than we may have wanted. Specifically, the interface contains an idealized “signature string” that is passed around among parties [...].

Indeed, Canetti [8, p. 7] starts by describing a “first attempt” functionality \mathcal{F}_1 that is a “depository of signed messages,” where the signer can input a message and the verifiers can check. This functionality can be seen as a simplified version of the authenticated repository we described above. He then argues, however, that including the technical details in the functionality’s interface is inevitable, see [8, p. 7]:

The lack of explicit signature strings also causes some other modeling problems. For instance, modeling natural operations such as sending an “encrypted signature” that is usable only by the holders of the decryption key cannot be done in a modular way [...]. We conclude that in order to capture our intuitive notion of

signature schemes, an ideal signature functionality should make the “signature string” part of its interface. [...]

We want to argue here that, despite the similarity, the arguments given in [8] do not apply to our definition. The first argument is that the formulation binds the messages to the signer’s identity instead of the verification key, which requires prior communication to transmit the verification key. While this argument is correct, and our definition makes the repository for transmitting the verification key explicit, we stress that the repositories abstract from concrete types of communication and merely specify that the correct verification key generated by the signer is accessible, in some way, to the verifier. The means of *how* it became accessible do not have to be specified.

The second argument is that (beyond requiring the communication of the signature string, which is analogous to the verification key), protocols that communicate a signature over a different connection than specified, such as an encrypted one, is a modeling challenge. One such protocol is SAML [16], where a signed assertion on the identity of a party is sent through a TLS connection. Despite the fact that this assertion is indeed encrypted, and SAML would therefore appear to be in the class of protocols referred to by Canetti, we show that our model, which does not explicitly expose the signature string, indeed allows to analyze the security of protocols like SAML. The reason is again that our model abstracts from the concrete communication semantics and in particular also allows to model the case where a signature is transferred securely.

There are protocols that make explicit use of the verification key or signature as a bit string and for which our model in its current form does not support a modular analysis. One example is the transformation from CPA-secure public-key encryption (PKE) to non-malleable PKE by Choi *et al.* [13], where each ciphertext is protected via an instance of a one-time signature scheme, and the bits of the verification key are used to select a certain subset of instances of the CPA-secure PKE. For the security reduction to succeed, however, it is necessary that the verification key be not only a bit string, but that it also be different for each instance, with high probability. While this property is clearly satisfied by every secure DSS, and therefore also each DSS that realizes \mathcal{F}_{SIG} , it is not captured in the functionality alone, where the adversary can freely choose the verification key. Hence, a composable analysis of the Choi *et al.* scheme in the \mathcal{F}_{SIG} -hybrid model is inherently impossible. In summary, this shows that the property of outputting *some* string as the verification key is not sufficient at least for the application of [13]. Another example are protocols that require parties to provide proofs (e.g., of knowledge) over inputs and outputs of the DSS algorithms. Yet, also here, the same issues appear with the formalization \mathcal{F}_{SIG} that is independent of any concrete scheme. In summary, it remains open whether there is a natural scheme that can be modularly proved based on \mathcal{F}_{SIG} , but not using the more abstract definition we put forth in this paper.

Finally, our work can be seen as orthogonal to the work of Canetti et al. [12], which extends the model of Canetti [6, 8] to the case where verification keys are available globally. While our model does not restrict the use of the constructed

resource, the central aspect of our work is the different paradigm underlying the specification of the functionalities.

1.3 Contributions

The first main contribution of our work is the formal model sketched in Sect. 1.1 above, which we formally specify in Sect. 3. We additionally prove several statements about DSSs using this model; in particular, we exemplify the use of the construction by two applications.

Capturing the security of a DSS. We define, in Sect. 4.1, the security of a DSS as constructing an authenticated repository, shown on the right-hand side of Fig. 2, from an insecure repository, called “insecure Rep” on the left-hand side of Fig. 2, an “authenticated Rep” to which one message can be written, and a “secure Rep” that allows to write a single message, but to which the adversary has neither read- nor write-interfaces. As shown in Fig. 2, using the signature scheme, which consists of the converters labeled *setup*, *write*, and *check*, requires the two single-message repositories for distributing the signing and verification keys. In more detail, in *write* each message is signed and the signature input into the insecure repository. Checking whether a given message m has been written to the repository is done by verifying the received signature for m within *check*.

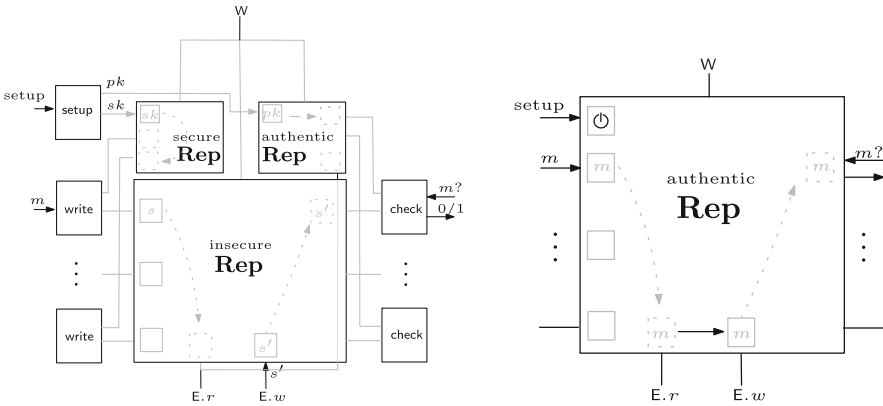


Fig. 2. Illustration of the main construction that characterizes a digital signature scheme. The assumed resources with the protocol (left) and the constructed resource (right). The adversarial interfaces are denoted by $E.w$ (write) and $E.r$ (read) and the free interface is denoted by W . The protocol is applied at the honest users’ interfaces of the assumed resources.

We then prove that this construction statement is equivalent to the existential unforgeability of secure digital signature schemes in the sense of [15]:

Theorem (informal). *A DSS constructs an authenticated multi-message repository from an insecure multi-message repository, an authenticated single-message repository and a secure single-message repository if and only if it is existentially unforgeable.*

Following the discussion in [8], we have to argue that our abstract formalization of a signature scheme indeed models the intuitively expected properties of such a scheme. In particular, in Sect. 5, we show that the formalization directly models the *transferability* property of signature schemes in the sense that a receiver of a signature can forward it to another party, who can also verify it. We then proceed to discuss two concrete applications.

Message registration resource. We show that the security of a DSS in our model immediately implies that it can be used to construct a (authenticated) message registration resource. This resource allows multiple parties to input messages, which are then authenticated by one party referred to as the *issuer*. Letting the messages be public keys corresponds to the use of signatures in a public-key infrastructure.

Assertions and SAML. Finally, we show how our constructive definition can be used to prove the soundness of an important step in single-sign-on (SSO) mechanisms, which is to authenticate a session between a client and a server (often denoted service provider in this context) with the help of a digitally signed assertion from an identity provider.

2 Preliminaries

2.1 Discrete Systems and Notation

We model all components as discrete reactive systems and describe them in pseudo-code using the following conventions: We write $x \leftarrow y$ for assigning the value y to the variable x . For a distribution \mathcal{X} over some set, $x \leftarrow \mathcal{X}$ denotes sampling x according to \mathcal{X} . For a finite set X , $x \leftarrow X$ denotes assigning to x a uniformly random value in X . For a table T of key-value pairs, with values in a set \mathcal{V} and keys in a set \mathcal{S} , we denote by the assignment $T[s] \leftarrow v$ the binding of a key $s \in \mathcal{S}$ to a value $v \in \mathcal{V}$. This assignment overwrites any prior binding of s to some value. Analogously, $v \leftarrow T[s]$ denotes the look-up of the value that is currently bound to key s . If no value is bound to s , this look-up is defined to return \perp . The *empty table* is defined as the table where any look-up returns \perp .

More formally, discrete reactive systems are modeled by random systems [19]. An important similarity measure on those is given by the distinguishing advantage. More formally, the advantage of a distinguisher \mathbf{D} in distinguishing two discrete systems, say \mathbf{R} and \mathbf{S} , is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) = \Pr[\mathbf{DR} = 1] - \Pr[\mathbf{DS} = 1],$$

where $\Pr[\mathbf{DR} = 1]$ denotes the probability that \mathbf{D} outputs 1 when connected to the system \mathbf{R} . More concretely, \mathbf{DR} is a random experiment, where the distinguisher repeatedly provides an input to one of the interfaces and observes the output generated in reaction to that input before it decides on its output bit.

A further important concept for discrete systems is a *monotone binary output (MBO)* [21] or *bad event* [4]. This concept is used to define a similarity between

two systems, the *game equivalence* [19] or *equivalence until bad* [4], which means that two systems behave equivalently until the MBO is set (i.e., as long as the bad event does not occur), but may deviate arbitrarily thereafter. A widely-used result is the so-called *Fundamental Lemma of Game Playing* [4, 19], which states that the distinguishing advantage between two such systems is bounded by the probability of provoking the MBO (i.e., bad event).

We stress that while especially the notion of bad event carries the connotation that such an event is supposed to occur only with small probability, this need not be the case. In particular, we will define specifications by means of the equivalence of two systems until an MBO is set, irrespective of how likely or unlikely this event is for a particular adversary. Such a specification is still interesting if, for each particular setting of interest, this probability turns out to be small.

2.2 Definition of Security

We use a term algebra to concisely write security statements. The *resources*, such as repositories, are written in bold-face font and provide *interfaces*, which are labeled by identifiers from a set \mathcal{I} , which can be accessed by parties. Protocol machines used by parties are also referred to as *converters* and are attached to some interface of a resource. This composition, which for a converter π , interface I , and resource \mathbf{R} is denoted by $\pi^I \mathbf{R}$, again yields a resource. For a vector of converters $\pi = (\pi_{I_1}, \dots, \pi_{I_n})$ with $I_i \in \mathcal{I}$, and a subset of interfaces $\mathcal{P} \subseteq \{I_1, \dots, I_n\}$, $\pi_{\mathcal{P}} \mathbf{R}$ denotes the resource where π_I is connected to interface I of \mathbf{R} for every $I \in \mathcal{P}$. For \mathcal{I} -resources $\mathbf{R}_1, \dots, \mathbf{R}_m$ the *parallel composition* $[\mathbf{R}_1, \dots, \mathbf{R}_m]$ is again an \mathcal{I} -resource that provides at each interface access to the corresponding interfaces of all subsystems.

In this paper, we make statements about resources with interface sets of the form $\mathcal{I} = \mathcal{P} \cup \{\mathbf{E}, \mathbf{W}\}$ where \mathcal{P} is the set of (honest) interfaces. A *protocol* is a vector $\pi = (\pi_{I_1}, \dots, \pi_{I_{|\mathcal{P}|}})$ that specifies one converter for each interface $I \in \mathcal{P}$. Intuitively, \mathcal{P} can be thought of as the interfaces that honestly apply the specified protocol π . On the other hand, interface \mathbf{E} corresponds to the interface with potentially dishonest behavior and no protocol is applied at this interface. Intuitively, this interface models the attacker’s capabilities to interfere with the honest protocol execution. Interface \mathbf{W} is the free interface that models the influence of the environment on the resource. A constructive security definition then specifies the goal of a protocol in terms of *assumed* and *constructed* resources. We state the definition of a construction of [22].

Definition 1. *Let \mathbf{R} and \mathbf{S} be resources with interface set \mathcal{I} . Let ε be a function that maps distinguishers to a value in $[-1, 1]$ and let the interface label set be $\mathcal{I} = \mathcal{P} \cup \{\mathbf{E}, \mathbf{W}\}$ with $\mathcal{P} \cap \{\mathbf{E}, \mathbf{W}\} = \emptyset$. A protocol, i.e., a vector of converters $\pi = (\pi_{I_1}, \dots, \pi_{I_{|\mathcal{P}|}})$, constructs \mathbf{S} from \mathbf{R} within ε and with respect to the simulator sim , if*

$$\forall \mathbf{D} : \Delta^{\mathbf{D}}(\pi_{\mathcal{P}} \mathbf{R}, \text{sim}^{\mathbf{E}} \mathbf{S}) \leq \varepsilon(\mathbf{D}). \tag{1}$$

This condition ensures that whatever an attacker can do with the assumed resource, she could do as well with the constructed resource by using the simulator sim . Turned around, if the constructed resource is secure by definition, there is no successful attack on the protocol.

The notion of construction is composable, which intuitively means that the constructed resource can be replaced in any context by the assumed resource with the protocol attached without affecting the security. This is proven in [22, 23].

Specifications and relaxed specifications. As discussed in the introduction, we consider *specifications* [23] of reactive discrete systems, meaning systems that are not fully specified. The specifications can be understood in the sense of game equivalence: we define an event on the inputs (and outputs) of the discrete system, and the specification states that a system must show a certain specified behavior until the condition is fulfilled, but may deviate arbitrarily afterward.

The security statements according to Definition 1 can then be understood as follows. A protocol constructs from a specification \mathcal{S} another specification \mathcal{T} if for each system \mathbf{S} that satisfies \mathcal{S} there exists a system \mathbf{T} that satisfies \mathcal{T} such that the protocol constructs \mathbf{T} from \mathbf{S} [23].

While game equivalence in general is defined based on an arbitrary MBO of the system, the MBOs considered in this paper will be of a specific and simple form: they only depend on the order in which specific inputs are given to the systems. This formalizes the guarantee that the resource behaves according to the specification if the inputs have been given in that order. A stronger condition therefore corresponds to a weaker specification, and it is easy to see that if a protocol constructs \mathcal{T} from \mathcal{S} , and the same additional condition is specified to obtain weakened specifications \mathcal{S}^- from \mathcal{S} and \mathcal{T}^- from \mathcal{T} , then the same protocol also constructs \mathcal{T}^- from \mathcal{S}^- . (This assumes that \mathcal{S}^- and \mathcal{T}^- are weakened in the same way. The statement can equivalently be seen as requiring the distinguishing advantage to be small only for a subset of distinguishers).

As the specifications in this work, as described above, can be seen as partially defined discrete systems, we use the same notation, i.e., boldface fonts. In particular, we can understand Eq. (1) as extending to such partially defined discrete systems, by changing the system to respond with a constant output to the distinguisher once the MBO has been provoked. Due to the specific property of the MBO, a distinguisher cannot gain advantage by provoking the MBO.

2.3 Digital Signature Schemes

We recall the standard definition of a DSS from the literature.

Definition 2. A digital signature scheme $\Sigma = (K, S, V)$ for a message space \mathcal{M} and signature space Ω consists of a (probabilistic) key generation algorithm K that returns a key pair (sk, vk) , a (possibly probabilistic) signing algorithm S , that given a message $m \in \mathcal{M}$ and the signing key sk returns a signature $s \leftarrow S_{sk}(m)$, and a (possibly probabilistic, but usually deterministic) verification algorithm V , that given a message $m \in \mathcal{M}$, a candidate signature $s' \in \Omega$, and the verification

key vk returns a bit $V_{vk}(m, s')$. The bit 1 is interpreted as a successful verification and 0 as a failed verification. It is required that $V_{vk}(m, S_{sk}(m)) = 1$ for all m and all (vk, sk) in the support of K . We generally assume $\mathcal{M} = \Omega = \{0, 1\}^*$.

The standard security definition for DSS is existential unforgeability under chosen message attack [15], as described in the introduction. Since we target concrete security, we directly define the advantage of an adversary.

Definition 3 (EU-CMA). For a digital signature scheme $\Sigma = (K, S, V)$, the EU-CMA advantage of an adversary \mathbf{A} is defined using the security game $\mathbf{G}_\Sigma^{\text{EU-CMA}}$ in Fig. 3, in more detail,

$$\Gamma^{\mathbf{A}}(\mathbf{G}_\Sigma^{\text{EU-CMA}}) := \Pr^{\mathbf{A}\mathbf{G}_\Sigma^{\text{EU-CMA}}}[\text{WON} = 1].$$

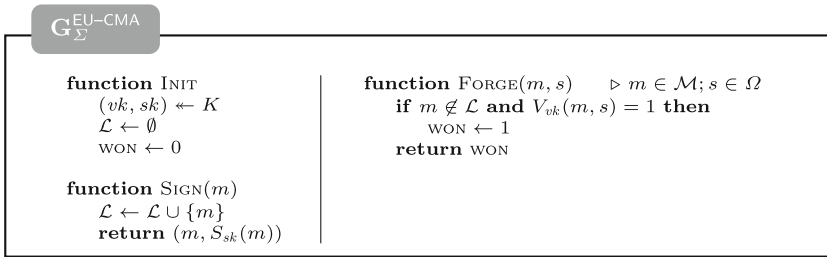


Fig. 3. The security game EU-CMA.

Signature schemes may or may not allow to recover the message from the signature. Each signature scheme can easily be turned into one with message recovery by viewing (m, s) as the signature instead of s .

Definition 4. A digital signature scheme with message recovery $\Sigma_{\text{rec}} = (K, S, R)$ is a digital signature scheme where the verification algorithm V is replaced by a recovery algorithm R , that takes a candidate signature s' and outputs a value $R_{vk}(s') \in \mathcal{M} \cup \{\perp\}$, where \perp is used to indicate that the signature s' is invalid. The correctness condition demands that $R_{vk}(S_{sk}(m)) = m$ for all m and all (vk, sk) in the support of K . The security notion is as in Definition 3, except for the winning condition: a successful adversary provides a signature s' such that $m' := R_{vk}(s') \neq \perp$ and m' was not a query to the signing oracle.

3 Message Repositories

We formalize the message repositories described in the introduction, and show how they can be instantiated to model specific communication networks.

3.1 Description of Message Repositories

We consider general message repositories that export a certain capability, such as reading or writing a single message, at each of its interfaces. There are four types of ways in which one can access the repository to read or write its content: each interface $A \in \mathcal{W}$ allows to insert one message into the repository. Interface $B \in \mathcal{R}$ allows to read a message that has been written to the repository and made visible for B . Each interface $C \in \mathcal{C}$ allows to write values into the repository by specifying from which (reader) interfaces the values should be copied; no new values can be inserted at interface C . For each copy-interface, there is a set of associated read-interfaces from which they can copy. Each interface $V \in \mathcal{V}$ allows to verify whether a certain value m is visible at the interface; this can be seen as a restricted type of read access. Finally, the free interface W allows to manage the visibility of messages. On a call $\text{TRANSFER}(A, B)$, the message written at A becomes visible at reader interface B . We often call the receiving interfaces *the receivers*. A precise specification of the repository appears in Fig. 4. As indicated by the keyword **Assume**, the behavior of the repository may be undefined if this assumption is not fulfilled, this is according to the discussion of specifications in Sects. 1 and 2. In contrast, “ $\triangleright m \in \mathcal{M}$ ” is to be understood as a reminder or comment for the reader; the input m given to the system is necessarily in the alphabet \mathcal{M} by definition of the system. (More technically, while the condition in **Assume** may be violated by an input, which may provoke an MBO, $m \in \mathcal{M}$ will always be satisfied).

Note that one can easily generalize this basic specification to other types of read- or write-interfaces, for example to model output of partial information about a message, such as the length, but which we do not consider here and consider it as part of future work. Following the motivation of Sect. 1, for generality, we consider each described operation as associated with a separate interface.²

Definition 5. For finite and pairwise disjoint sets $\mathcal{W}, \mathcal{R}, \mathcal{C}, \mathcal{V}$, and a family $\{\mathcal{R}_C\}_{C \in \mathcal{C}}$ of sets $\mathcal{R}_C \subset \mathcal{R}$ for all $C \in \mathcal{C}$, we define the repository $\mathbf{Rep}_{\mathcal{R}, \mathcal{V}, \{\mathcal{R}_C\}_{C \in \mathcal{C}}}^{\mathcal{C}, \mathcal{W}}$ as in Fig. 4. For later reference, we define for $n, m, \ell, k \in \mathbb{N}$, the standard sets $\mathcal{W} = \{A_i\}_{i \in [n]}$, $\mathcal{R} = \{B_i\}_{i \in [\ell]}$, $\mathcal{C} = \{C_i\}_{i \in [m]}$ and $\mathcal{V} = \{V_i\}_{i \in [k]}$. If nothing else is specified, these standard interface names are used. We define the shorthand notation $\mathbf{Rep}_{\ell, k}^{m, n} := \mathbf{Rep}_{\mathcal{R}, \mathcal{V}, \{\mathcal{R}_C\}_{C \in \mathcal{C}}}^{\mathcal{C}, \mathcal{W}}$ for these standard sets and $\mathcal{R}_C = \mathcal{R}$ for all $C \in \mathcal{C}$. For $C = \emptyset$ we use the simplified notation $\mathbf{Rep}_{\mathcal{R}, \mathcal{V}}^{\mathcal{W}}$.

Different security guarantees can be expressed using this repository by considering different allocations of read-, write-, or transfer-interfaces to different parties as discussed in the introduction. For instance, an attacker could have access to both read- and write-interfaces, to model traditional insecure communication. If the attacker only has access to read-interfaces (but not to write-interfaces beyond potentially copy-interfaces to forward received messages), the

² Recall that it is always possible to merge several existing interfaces into one interface to model that a party or the attacker, in a certain application scenario, has the capability to write and read many messages.

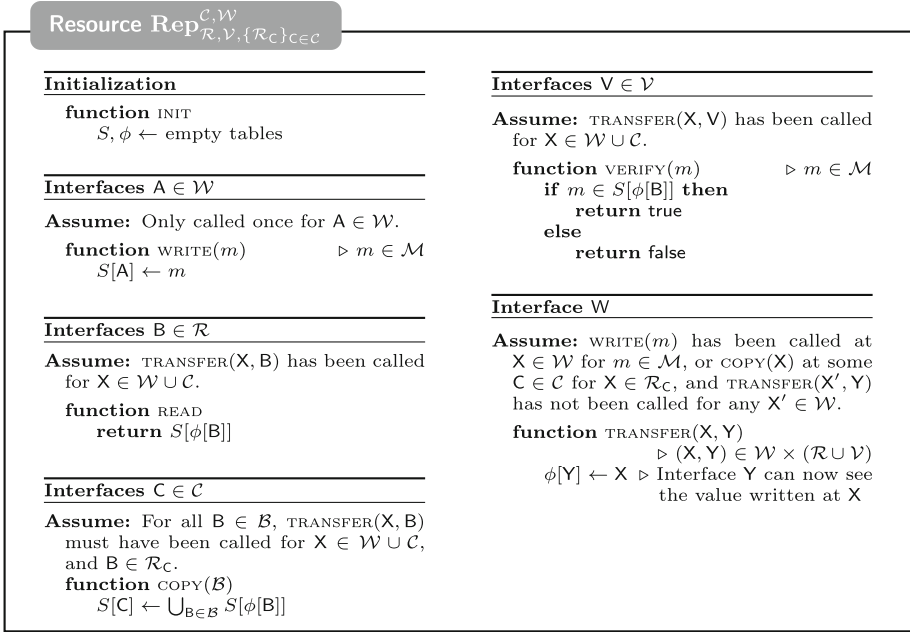


Fig. 4. Specification of a repository resource. For ease of notation, we treat values $m \in \mathcal{M}$ and singular sets $\{m\}$ for $m \in \mathcal{M}$ interchangeably.

repository corresponds to authenticated message transmission from a honest write-interface.

3.2 Modeling Security Guarantees by Access to the Repository

For security statements we need to associate each (non-free) interface to either an honest party or a possible attacker. As additional notation, we define the adversarial interfaces sets $\mathcal{E}_r := \{E_{1.r}, \dots, E_{k.r}\}$ (for some $k > 0$), $\mathcal{E}_w := \{E_{1.w}, \dots, E_{k.w}\}$, and $\mathcal{E}_c := \{E_{1.c}, \dots, E_{k.c}\}$ where the size k of this set is typically defined by the context. We can then specify repositories with different security guarantees.

- Insecure repositories allow adversarial write and read access. They can be described by $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$, which means that all interfaces are either read- or write-interfaces.
- An authenticated repository disallows adversarial write-operations of arbitrary messages. Only (the honest) interface \mathcal{W} can input content into the repository. This situation is described by the resource $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset, \{\mathcal{E}_r\}_{c \in \mathcal{C}}}^{\mathcal{E}_c, \mathcal{W}}$, which indicates that the attacker may still be able to copy values from interfaces \mathcal{E}_r at each interface \mathcal{E}_c .
- A repository without adversarial read-access, but with write access, models perfect confidentiality, and is described by $\mathbf{Rep}_{\mathcal{R}, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$.

While the (natural) variants described above will be the only ones used in this work, the formalism allows to flexibly define various further combinations of honest-user and adversarial capabilities.

3.3 Example: Modeling Networks Through Repositories

For considering concrete applications, such as a specific type of network transfer, the repository can be instantiated appropriately. In this section, we briefly describe in which sense statements about repositories imply statements about a network in which senders can *send* a message to a set of desired recipients, but which is under complete control of an attacker. We describe such a network in more detail in the full version of this work [3]. In a nutshell, such a network can be described as a repository where for each write-interface of the honest senders, the attacker interface has a read-interface, and for each read-interface of the honest receivers, the attacker interface has a write-interface. Additionally, the attacker interface has the capabilities of the free interface that allow to transfer the values between the write- and the read-interfaces. This enables the attacker to eavesdrop on all values from the writer and to determine all values sent to the receiver; the traditional worst-case assumption.

4 A Constructive Perspective on Digital Signatures

4.1 The Basic Definitions

Our security definition for DSSs is based on the repositories introduced in Sect. 3. Intuitively, the honest parties execute a protocol to construct from an insecure repository, in which the attacker has full write access, one repository that allows the writer to authenticate a single message (this will be used for the verification key), and one repository that allows to store a single message securely (this will be used for the signing key), an authenticated repository that can be used for multiple messages. We generally use the notation introduced in Sect. 3. We first introduce the specifications that capture authenticated repositories since they are of primary interest in this section. The first type considers repositories where the role of the receiver interfaces is to verify values in the repository:

Definition 6. *Let $\mathcal{W}, \mathcal{R}, \mathcal{E}_w, \mathcal{E}_r$ denote the standard interface names. A specification $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$, in the sense of a partially defined discrete system, is an authenticated repository for verification if the following conditions are fulfilled. (1) It has at least the interfaces $\mathcal{I} = \mathcal{W} \cup \mathcal{R} \cup \mathcal{E}_w \cup \mathcal{E}_r$, where all inputs at $I \notin \mathcal{I}$ are ignored (i.e., the resource has the default behavior of directly returning back to the caller). (2) For all inputs at some interface $I \in \mathcal{I}$, the behavior is identical to the one specified in $\mathbf{Rep}_{\mathcal{E}_r, \mathcal{R}, \{\mathcal{E}_r\}}^{\mathcal{E}_w, \mathcal{W}, \mathcal{C} \subseteq \mathcal{E}_w}$ for I , wherever the behavior of $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ is defined. More formally, this means that for a given sequence of inputs, the conditional distribution of $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$, where the outputs for inputs at interfaces not in \mathcal{I} are marginalized, is the same as the conditional distribution of $\mathbf{Rep}_{\mathcal{E}_r, \mathcal{R}, \{\mathcal{E}_r\}}^{\mathcal{E}_w, \mathcal{W}, \mathcal{C} \subseteq \mathcal{E}_w}$ without those inputs.*

The second definition is analogous and considers repositories where the role of the receiver interfaces is to authentically receive values:

Definition 7. Let $\mathcal{W}, \mathcal{R}, \mathcal{E}_w, \mathcal{E}_r$ denote the standard interface names. The specification $\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$, in the sense of a partially defined discrete system, is an authenticated repository for receiving if it has at least the interfaces $\mathcal{I} = \mathcal{W} \cup \mathcal{R} \cup \mathcal{E}_w \cup \mathcal{E}_r$, all inputs at $I \notin \mathcal{I}$ are ignored, and for all inputs at some interface $I \in \mathcal{I}$ the behavior is identical to the one specified in $\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}, \emptyset, \{\mathcal{E}_r\}_{c \in \mathcal{E}_w}}^{\mathcal{E}_w, \mathcal{W}}$ for I , wherever the behavior of $\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ is defined. We omit \mathcal{E}_w in the notation if it is equal to \emptyset .

In the following, whenever referring to the sets $\mathcal{W}, \mathcal{R}, \mathcal{E}_w$, and \mathcal{E}_r , we implicitly refer to the standard names introduced in the previous section.

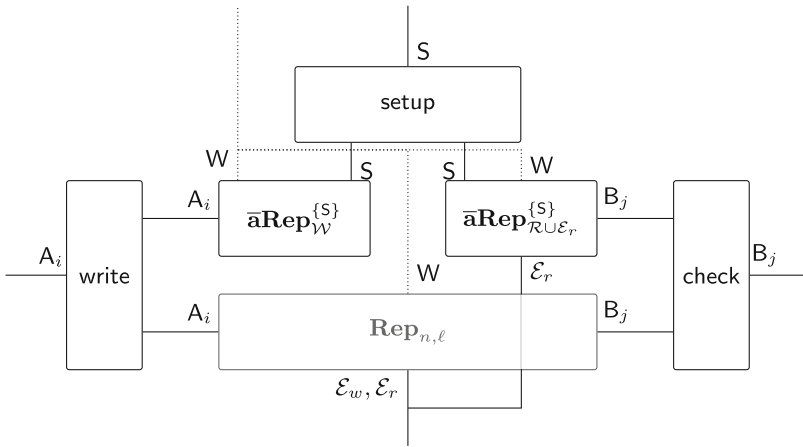


Fig. 5. The real-world setting of the signature construction.

Assumed resources. As outlined in Sect. 1, to construct an authenticated repository, we require (beyond an insecure repository to transmit the signatures) an additional resource that allows to distribute one value authentically to all verifiers and one value securely to all signers (Fig. 5). This assumed communication is described by the specification $\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{W}}^{\{S\}}$, which specifies resources with one writer interface S and no active adversarial interface. Information can only be transferred from S to the interfaces of \mathcal{W} . To model the authenticated (but not confidential) transmission of a value, we assume another resource as specified by $\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_c, S}$ where information can only be transferred from S to the interfaces in \mathcal{R} , but is not limited to those as also adversarial interfaces may read this value or copy it via the interfaces in \mathcal{E}_c . We define the assumed system as consisting of the two above-described resources and an insecure repository $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$, i.e., as

$$\mathbf{R}_{n,\ell} := \left[\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{W}}^{\mathcal{S}}, \bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_c, \mathcal{S}}, \mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w} \right]. \quad (2)$$

For clarity, whenever we explicitly refer to the assumed mechanism to distribute the keys, we use the shorthand notation

$$\mathbf{Dist} := \left[\bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{W}}^{\mathcal{S}}, \bar{\mathbf{a}}\mathbf{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_c, \mathcal{S}} \right].$$

Protocol converters. We assign one converter to each of the three roles: a converter `write` for the (honest) writer interfaces, a converter `check` for the (honest) reader interfaces and a setup-converter `setup` at interface `C`. We define the vector of converters $\mathbf{DSS} := (\text{setup}, \text{write}, \dots, \text{write}, \text{check}, \dots, \text{check})$ with n copies of `write`, ℓ copies of converter `check` and one converter `setup`. The set of honest interfaces in this section is defined as $\mathcal{P} := \{\mathcal{S}\} \cup \mathcal{W} \cup \mathcal{R}$.

Goal of construction: an authenticated repository. Intuitively, the use of a DSS should allow us to construct from a repository $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r}^{\mathcal{W} \cup \mathcal{E}_w}$ that allows both the honest users and the attacker to write multiple messages, and a repository that exclusively allows one honest user to write the verification key authentically, a repository in which the attacker has no write access. The reason is that writing a message that will be accepted by honest readers requires to present a valid signature relative to the verification key, thus the attacker would be required to forge signatures. This intuition does, however, not quite hold.

Indeed, when using the insecure repository, the attacker can still copy valid signatures generated by the honest writer to which he has read access via any of his write interfaces. Since honest readers may later gain read access to those copied signatures, the attacker can indeed control *which* of the messages originating from the honest writer will be visible at those interfaces. The repository that is actually constructed is a specification $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ as in Definition 6. The goal of a digital signature scheme can thus be understood as amplifying the capabilities of authenticated repositories as defined using the specifications above.

To give a more concrete intuition, a particular constructed resource still has an interface `S` and accepts queries `TRANSFER(S, Ai)` and `TRANSFER(S, Bj)`, in addition to those provided by $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. Providing input at these interfaces, as indicated by the dead ends drawn in Fig. 6, has no effect, but may influence whether further outputs of the system are still defined (because, e.g., inputs to the system may have been provided in an order such that the behavior of the DSS is not defined).

In the remainder of the section, we prove an equivalence between the validity of the described construction and the definition of existential unforgeability. As the protocol converters described above do not exactly match the algorithms in the traditional definition of a DSS, we also explain how to convert between the two representations of a signature scheme.

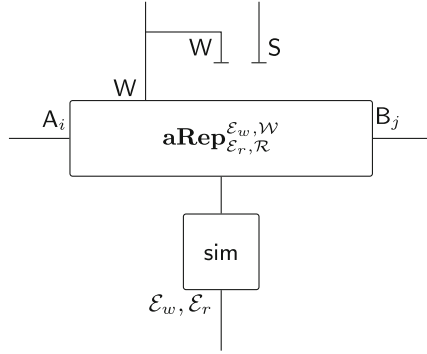


Fig. 6. The ideal-world setting of the signature construction. Inputs at the interfaces whose corresponding lines stop before the box (interfaces W and S in this example) have no effect on the behavior, therefore they are ignored in the specification.

4.2 Unforgeability of Signatures Implies Validity of Construction

The constructed specification $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ has further (inactive) interfaces beyond those in $\mathcal{I} = \mathcal{W} \cup \mathcal{R} \cup \mathcal{E}_w \cup \mathcal{E}_r$, and behaves equivalently to $\mathbf{Rep}_{\mathcal{E}_r, \mathcal{R}, \{\mathcal{E}_r\}_{C \in \mathcal{E}_w}}^{\mathcal{E}_w, \mathcal{W}}$, as long as the assumed order of inputs is respected. The following theorem states that any existentially unforgeable digital signature scheme can be used to construct such an authentic repository from the assumed resources (see also Fig. 2 for a depiction of this statement).

Constructing a specification $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ according to Definition 6 can be a vacuous statement: the specification can be undefined for all possible orders of inputs. The statement we prove in this section, therefore, explicitly specifies for which orders $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ is defined. In particular, the specification is defined for all orders of inputs for which the underlying specifications $\bar{\mathbf{aRep}}_{\mathcal{W}}^S$ and $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_c, S}$ are defined, plus the following natural conditions of a DSS: the keys are generated first and are distributed before anything is signed or verified at a writer or reader interface. As long as these conditions are satisfied, the specification defines the output of the resource.

We now state the formal theorem whose proof appears in the full version [3].

Theorem 1. *Let $n, \ell \in \mathbb{N}$. For any given digital signature scheme $\Sigma = (K, S, V)$, let the converters write, check, and setup be defined as in Fig. 8. Then, for the simulator \mathbf{sim} defined in Fig. 7, there is an (efficient) reduction \mathbf{C} described in the proof, that transforms any distinguisher \mathbf{D} for systems $\text{DSS}_{\mathcal{P}}\mathbf{R}_{n, \ell}$ and $\mathbf{sim}^E \mathbf{aRep}_{n, \ell}$, with $\mathbf{aRep}_{n, \ell} = \mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ as described above, into an adversary $\mathbf{A} := \mathbf{DC}$ against the game $\mathbf{G}_{\Sigma}^{\text{EU-CMA}}$ such that*

$$\Delta^{\mathbf{D}}(\text{DSS}_{\mathcal{P}}\mathbf{R}_{n, \ell}, \mathbf{sim}^E \mathbf{aRep}_{n, \ell}) \leq \Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\text{EU-CMA}}),$$

and where $\mathbf{aRep}_{n, \ell}$ is defined as long as the assumed specification is defined and the following conditions hold:

- Command **SETUP** is issued at the \mathcal{S} -interface before any other command;
- Command **TRANSFER**($\mathcal{S}, \mathcal{A}_i$) is issued at the \mathcal{W} -interface corresponding to the first setup repository before **WRITE** is issued at the \mathcal{A}_i -interface;
- Command **TRANSFER**($\mathcal{S}, \mathcal{B}_i$) is issued at the \mathcal{W} -interface corresponding to the second setup repository before **READ** is issued at the \mathcal{B}_i -interface.
- There are no **TRANSFER**(\mathcal{X}, \mathcal{Y}) queries with $\mathcal{X} \in \mathcal{E}_w$ and $\mathcal{Y} \in \mathcal{E}_r$, that is, we exclude communication from the adversarial writer to adversarial reader-interfaces.

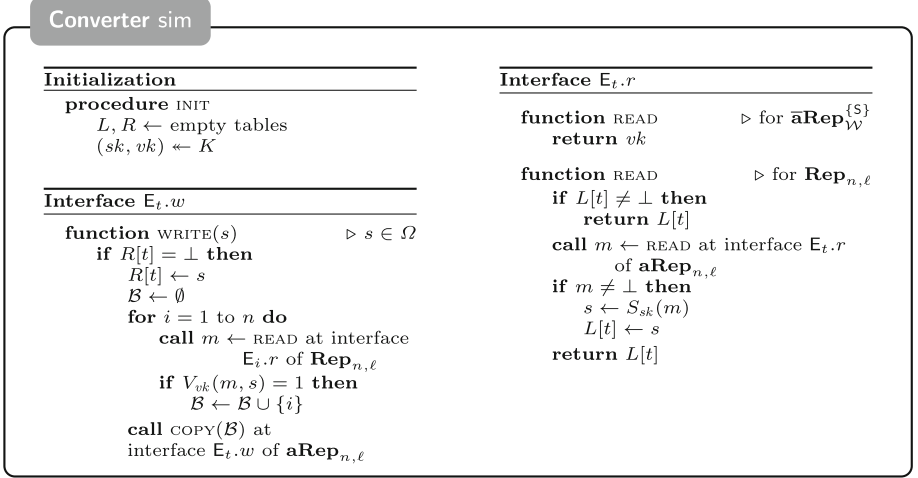


Fig. 7. Simulator for the proof of Theorem 1.

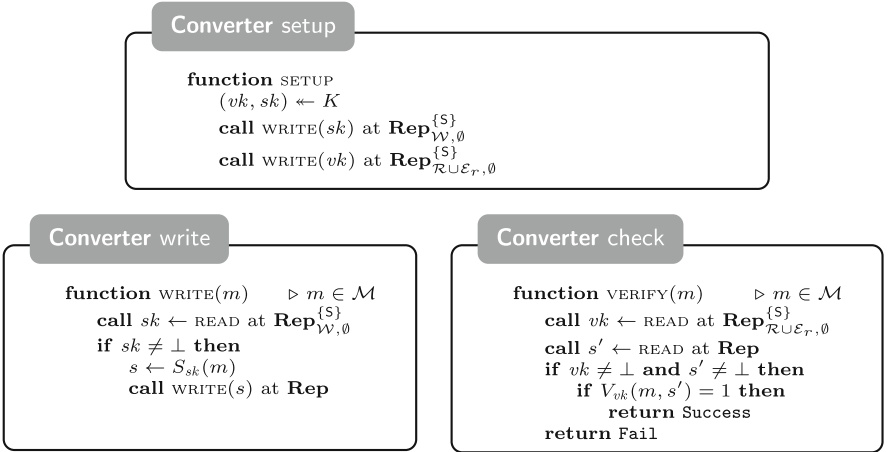


Fig. 8. The three protocol converters derived from a signature scheme $\Sigma = (K, S, V)$.

4.3 Chaining Multiple Construction Steps

The construction proved in Theorem 1 assumes (amongst others) an authenticated repository $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{S}}$ and constructs an authenticated repository $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. A natural question is in which sense multiple such construction steps can be chained, corresponding to signing the verification key of one instance with a different instance of the scheme. For this to work out, we have to “upgrade” the resource $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ to a resource $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ as needed by Theorem 1, where we can then use any interface $X \in \mathcal{W}$ as the interface \mathcal{S} to transmit the secret key. Of course, we additionally require resources $\bar{\mathbf{aRep}}_{\mathcal{W}'}^{\{X\}}$ for distributing the secret keys and $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W}' \cup \mathcal{E}'_w}$ for transmitting the signatures.

The chaining is then achieved by the protocol that consists of converters send and receive, sends the messages over an (additional) insecure repository $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$ and authenticates them via $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. Protocol converter send simply inputs the same message to both resources, whereas receive verifies the messages obtained through the insecure repository at the authenticated repository. This protocol perfectly constructs an authenticated repository with delivery from the two assumed resources.

Theorem 2. *Let $n, \ell \in \mathbb{N}$, and consider a protocol SND with converters send for all interfaces in \mathcal{W} and converters receive for all interfaces in \mathcal{R} , defined as described above. Then, for the simulator sim described below,*

$$\text{SND}_{\mathcal{P}} \left[\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}, \mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}} \right] \equiv \text{sim}^E \bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}},$$

wherever both resources are defined. The constructed resource $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ accepts TRANSFER commands at sub-interfaces corresponding to both assumed resources, and requires, for a given message to be transferred, both those commands to be issued.

The simulator sim responds to READ queries at the \mathcal{E}_r -interfaces corresponding to $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$ or $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ by obtaining the transmitted messages from $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. Once COPY has been called at an \mathcal{E}_w -interface at $\mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$ and the corresponding message has been input at the same \mathcal{E}_w -interface of $\mathbf{Rep}_{\mathcal{R} \cup \mathcal{E}_r, \emptyset}^{\emptyset, \mathcal{W} \cup \mathcal{E}_w}$, sim issues the same COPY command at $\bar{\mathbf{aRep}}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. Together with Theorem 1, this means that sending a message along with a signature constructs an authenticated repository from which the authenticated messages can be read. Several such constructions can then be chained in the expected way.

4.4 Validity of Construction Implies Unforgeability of Signatures

In this section, we show that any converters achieving the construction of \mathbf{aRep} from \mathbf{Rep} and \mathbf{Dist} contain a digital signature scheme that is existentially unforgeable under chosen-message attacks. More precisely, we state the following theorem proven in the full version [3].

Theorem 3. *Let $n, \ell \in \mathbb{N}$. Consider arbitrary converters **setup**, **write**, and **check** and define the protocol as $\text{DSS} := (\text{setup}, \text{write}, \dots, \text{write}, \text{check}, \dots, \text{check})$ (for the honest interfaces) with n copies of **write**, ℓ copies of converter **check** and one converter **setup**. We derive a digital signature scheme $\Sigma = (K, S, V)$ below in Fig. 9 with the following property: given any adversary against the signature scheme that asks at most n queries to **SIGN** and ℓ queries to **FORGE**, we construct (efficient) distinguishers \mathbf{D}_i , $i = 1 \dots 5$, such that for the systems $\text{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\text{sim}^{\mathbf{E}}\mathbf{aRep}_{n,\ell}$, with $\mathbf{aRep}_{n,\ell} = \mathbf{aRep}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$, for all simulators sim ,*

$$\Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\text{EU-CMA}}) \leq \sum_{i=1}^5 \Delta^{\mathbf{D}_i}(\text{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}, \text{sim}^{\mathbf{E}}\mathbf{aRep}_{n,\ell}),$$

and where $\mathbf{aRep}_{n,\ell}$ is defined as long as the assumed specification is defined and under the same additional conditions as in Theorem 1.

As a corollary, one can specifically deduce that if there exists a simulator sim such that systems $\text{DSS}_{\mathcal{P}}\mathbf{R}_{n,\ell}$ and $\text{sim}^{\mathbf{E}}\mathbf{aRep}_{n,\ell}$ are indistinguishable, then the constructed signature scheme Σ is existentially unforgeable under chosen message attacks.

Obtaining the signature scheme from the converters. The key generation, signing, and verification functions are derived from the converters **setup**, **write**, and **check** that construct \mathbf{aRep} from $[\mathbf{Dist}, \mathbf{Rep}]$ as follows: The key generation K consists of evaluating the function setup.SETUP , the two values written to the resource \mathbf{Dist} are considered as the corresponding key pair. The secret key is the value that is written to the first sub-system of \mathbf{Dist} . The signing algorithm $S_{sk}(m)$ consists of evaluating the function $\text{write.WRITE}(m)$. The signature for message m is defined as the value that is written to the repository. Any request to obtain a value from resource \mathbf{Dist} is answered by providing the signing key sk . The verification algorithm $V_{vk}(m, s)$ consists of evaluating the function $\text{check.VERIFY}(m)$ and the candidate signature s is provided as the actual value in the repository and the verification key vk is given as the value in \mathbf{Dist} . The formal description of the algorithms appear in Fig. 9.

4.5 Digital Signatures with Message Recovery

So far we have focused on repositories that offer the capability to check whether a given value has been written to the buffer and denoted them by \mathbf{aRep} . Now, we consider repositories that offer the capability to retrieve the value that has been transferred to an interface. In other words, the goal of this section is to show how to construct the specification $\bar{\mathbf{aRep}}_{\mathcal{E}_r, \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$. While the construction of \mathbf{aRep} from \mathbf{Rep} and \mathbf{Dist} is achieved by traditional signature schemes, the construction of $\bar{\mathbf{aRep}}$ from the same assumed resources is achieved by signature schemes with message recovery. Intuitively, converter **check** is replaced by a converter **read** whose task is to recover and output the message (and not simply check the authenticity of a given message). It is easy to see that any signature scheme

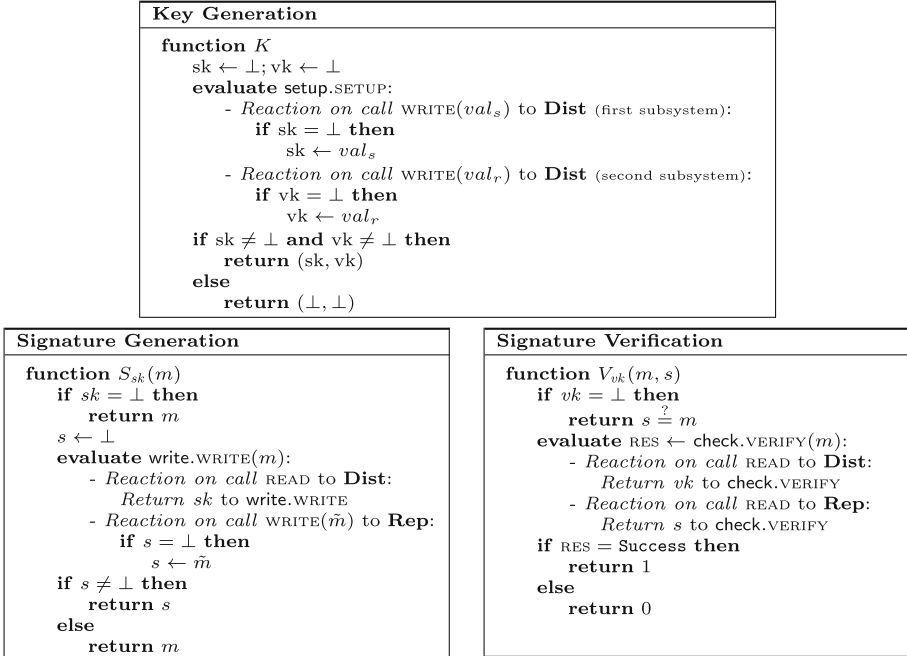


Fig. 9. Signature scheme (K, S, V) extracted from converters `setup`, `write`, and `check`.

$\Sigma_{\text{rec}} = (K, \Sigma, R)$ can be used to derive converters that achieve the construction (similar to the previous section). For the other direction, we have:

Theorem 4. *Let $n, \ell \in \mathbb{N}$. Consider arbitrary converters `setup`, `write`, and `read` and define the protocol as $\text{DSS} := (\text{setup}, \text{write}, \dots, \text{write}, \text{read}, \dots, \text{read})$ (for the honest interfaces) with n copies of `write`, ℓ copies of converter `read` and one converter `setup`. One can derive a digital signature scheme $\Sigma_{\text{rec}} = (K, S, R)$ with message recovery with the following property: given any adversary against the signature scheme that asks at most n queries to `Sign` and ℓ queries to `forge`, we derive (efficient) distinguishers \mathbf{D}_i , $i = 1 \dots 5$, such that for the systems $\text{DSS}_{\mathcal{P}\mathbf{R}_{n,\ell}}$ and $\text{sim}^E \bar{\mathbf{a}}\text{Rep}_{n,\ell}$, with $\bar{\mathbf{a}}\text{Rep}_{n,\ell} = \bar{\mathbf{a}}\text{Rep}_{\mathcal{E}_r \cup \mathcal{R}}^{\mathcal{E}_w, \mathcal{W}}$, for all simulators sim ,*

$$\Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma_{\text{rec}}}^{\text{EU-CMA}}) \leq \sum_{i=1}^5 \Delta^{\mathbf{D}_i}(\text{DSS}_{\mathcal{P}\mathbf{R}_{n,\ell}}, \text{sim}^E \bar{\mathbf{a}}\text{Rep}_{n,\ell}),$$

and where $\bar{\mathbf{a}}\text{Rep}_{n,\ell}$ is defined as long as the assumed specification is defined and under the same additional conditions as in Theorem 1.

Proof. We omit the proof and simply mention that it follows the same line of argumentation as the proof of Theorem 3. Algorithms K and S are derived in the same way as in Sect. 4.4 and the recovery algorithm $R_{vk}(s)$ is derived from

converter read by evaluating the function READ (and appropriately providing s and vk) and to return whatever this function returns. \square

5 On the Transferability of Verification Rights

Universal verification is arguably an important property of signatures. Anybody possessing the public key and a valid signature string s for some message m can verify the signature. This implies furthermore that signatures are naturally transferable, which is essential for their key role in public-key infrastructures or signing electronic documents. In this section, we demonstrate that our definition directly implies transferability by constructing a message repository in which information can be forwarded among readers. The high-level idea is to apply a converter to the free interface that instead copies the desired message from the sender buffer, where it was input originally, to the targeted reader buffer.

The role of the free interface. Recall that the role of the free interface in the repository resources is to transfer the contents from certain write-buffers to certain read-buffers. The transferability of signatures then simply means that values can also be transferred from read-buffers to other read-buffers; this can easily be achieved by translating the transfer-requests appropriately.

The core idea, then, is to observe that the new repository and the old repository only differ by attaching a converter at interface W . We assign a new name to this resource and define $\mathbf{aTRep} = \text{relay}^W \mathbf{aRep}$ (and analogously $\bar{\mathbf{aTRep}} := \text{relay}^W \bar{\mathbf{aRep}}$) with a converter relay that always remembers the existing assignments of reader to writer interfaces and on a transfer-query for two reader interfaces, it simply connects the corresponding writer-interface. The resource \mathbf{aTRep} is additionally formally described in Fig. 10.

The converter. Converter relay distinguishes two types of inputs: transfer commands from a writer to a reader $\text{TRANSFER}(X, Y)$ are forwarded to the connected repository. Transfer commands between two readers, $\text{TRANSFER}(R_1, R_2)$ are translated to transfer commands $\text{TRANSFER}(X, R_2)$, where X denotes the writer interface where the value readable at R_1 was first input.

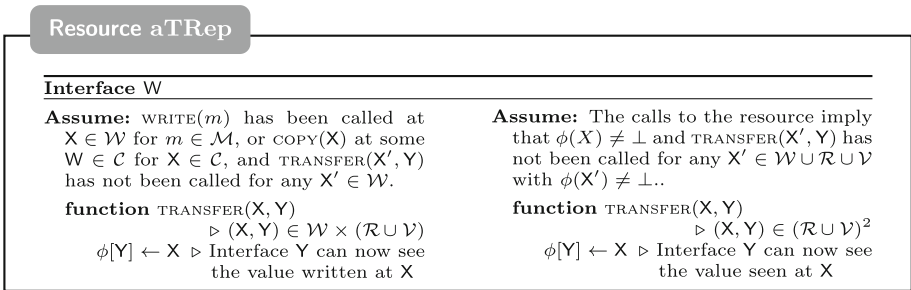


Fig. 10. Specification of a repository resource with transferable rights. Only the modifications with respect to Fig. 4 are shown; the other functions are as described there.

A simple black-box construction. Any protocol that constructs \mathbf{aRep} from \mathbf{Rep} (and \mathbf{Dist}) also constructs $\text{relay}^W \mathbf{aRep}$ from $\text{relay}^W \mathbf{Rep}$ (and \mathbf{Dist}), where the assumed resource $\text{relay}^W \mathbf{Rep}$ is an insecure repository that also allows information transfer between two receivers, i.e., sending a signature from one receiver to another. This is easy to see: assume there was a distinguisher \mathbf{D} for systems $\text{sim}^E \text{relay}^W \mathbf{aRep}$ and $[\text{relay}^W \mathbf{Rep}, \mathbf{Dist}]$, and we are going to construct a distinguisher \mathbf{D}' for the underlying two resources without the converter relay attached. (Note that sim is the same simulator as in Theorem 1). Distinguisher \mathbf{D}' simply behaves as \mathbf{D} but additionally emulates relay for queries at the free interface.

6 Application 1: Implementing a Registration Service

The goal of this section is to construct a resource that allows several parties to send messages authentically to a population of receivers, via one *issuer* that authenticates the messages. This happens in public-key infrastructures, where the issuer, which is also denoted by *certification authority* in that context and can authenticate messages, acts as a relay. This is the setup of a (simple) public-key infrastructure and its use in Internet protocols, where the senders correspond to the *submitters* of public keys to the CA (registration), and the receivers are the consumers those public keys to authenticate messages. For the remainder of the section, we will therefore refer to the senders as submitters and the receivers as consumers (although the resource can of course also be used in other protocols).

The registration resource \mathbf{Reg} . We denote the set of interfaces for the submitters by $\mathcal{S} := \{S_1, \dots, S_\ell\}$, the consumers by $\mathcal{C} := \{C_1, \dots, C_\ell\}$, and the interfaces for the issuer by I . The adversarial interface is denoted by E . The registration resource \mathbf{Reg} offers the capability to input a value x at any submitter interface. Once this value has been transferred to the issuer, he can acknowledge the value by calling `ISSUE` at its interface. Once this happened, the value x , together with the information which submitter has input the value, can be made available at any consumer interface and, in addition, it can be transferred between any two consumer interfaces (or submitter interfaces). The formal description of the behavior of \mathbf{Reg} appears in Fig. 11.

Assumed resources and the protocol. We assume a network resource \mathbf{Net} which allows any party interface to send (by calling `SEND`) and receive messages (by calling `RECEIVE`), and allows the attacker to read all messages and send any message (note that the honest parties in a network have no means to verify who sent the message). In addition, we assume authentic communication as a setup. More formally, let $\mathbf{Ch}_{I\leftarrow}$ be a system that has interface set $\{I\} \cup \mathcal{S} \cup \{E_1, \dots, E_\ell\}$. Each interface except the issuer offers the capability to send one message, i.e., to call `SEND(m)`, which can be fetched at the issuer interface (they are authentic in the sense that the message cannot be modified and the resource indicates to the receiver who is the sender of the message). The issuer interface I can be thought of as being divided into 2ℓ sub-interfaces, and each sub-interface offers the capability to obtain the message from the corresponding sender (and hence

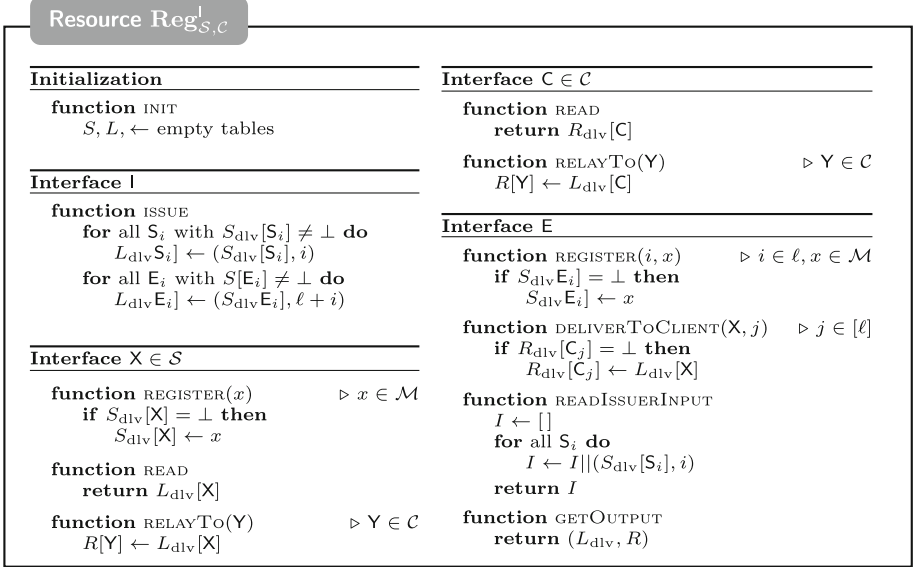


Fig. 11. The registration service resource.

identifies the sender reliably). Also, let the system $\mathbf{Ch}_{\mathbb{I} \rightarrow}$ be defined similarly, but which allows the issuer to send two messages in an authenticated manner to *each* submitter and one to each consumer.³

We can now describe the protocol that implements a registration service based on the above setup. The issuer's converter, upon **ISSUE**, takes all values x received on the incoming authenticated channel and acknowledges them by signing value (x, λ) , where λ is a unique identifier that the issuer assigns to its sub-interface from which x was received.⁴ The issuer sends the signed value back via the outgoing authentic channel. The protocol for the submitters, upon **REGISTER**(x) simply send x to the issuer over the authentic channel. Finally, the consumer converter reads inputs from the insecure network. When reading a new input, they verify the received value-signature pair and output the associated value only if the signature can be verified.

Theorem 5. *Let \mathcal{S}, \mathcal{C} be the above sets and \mathbb{I} an interface name (different from all remaining interfaces). The protocol described above (and formally specified as pseudo-code in the full version) constructs resource **Reg** from the described set of authenticated channels. More specifically, for converters **issue** and **reg**, there is a simulator $\tilde{\text{sim}}$ (formally specified as pseudo-code in the full version of this work), such that for all distinguishers **D** there is an attacker against the signature scheme (with essentially the same efficiency), i.e.,*

³ This setup reflects that we need to distribute the issuer's verification key to each participant and in addition one signature to each submitter.

⁴ In an application, this identifier could be the name of a server or a company. For concreteness we assume the identifier of the i th sub-interface to be the number i .

$$\Delta^D(\text{issue}^l \text{reg}^{S_1} \dots \text{reg}^{S_\ell} \text{rel}^{C_1} \dots \text{rel}^{C_\ell} [\text{Ch}_{\rightarrow}, \text{Ch}_{\leftarrow}, \text{Net}], \tilde{\text{sim}}^E \text{Reg}_{S,C}^l) \leq \Gamma^A(\mathbf{G}_\Sigma^{\text{EU-CMA}}).$$

Proof. Due to the abstract nature of repositories, we can easily represent the real world by a wrapped repository, and the ideal world as a wrapped authenticated repository and conclude the statement by invoking the results from the previous section. The proof is given in the full version [3]. \square

7 Application 2: Authenticating Sessions Using Assertions

Unilaterally secure channels. Establishing secure sessions in the internet is a crucial task. The most widely known solution to establish secure session is TLS, that, in a first handshake phase, establishes a shared key between client and server. Subsequently, this key is used to authenticate and encrypt the communication. In TLS, the server is usually authenticated, whereas the client is not. This results in an only unilateral authenticity guarantee [24]: while the client is guaranteed that its messages are received by the intended server, the server does not know whether he is communicating with a legitimate client or with an attacker. This guarantee for unilaterally secure channels is captured by the resource $\text{NET}_{\text{uni}}^n$, and the guarantee provided by the mutually authenticated secure channel is captured by the resource $\text{NET}_{\text{mut}}^{n,\text{ldP}}$ as described in Figs. 12 and 13.

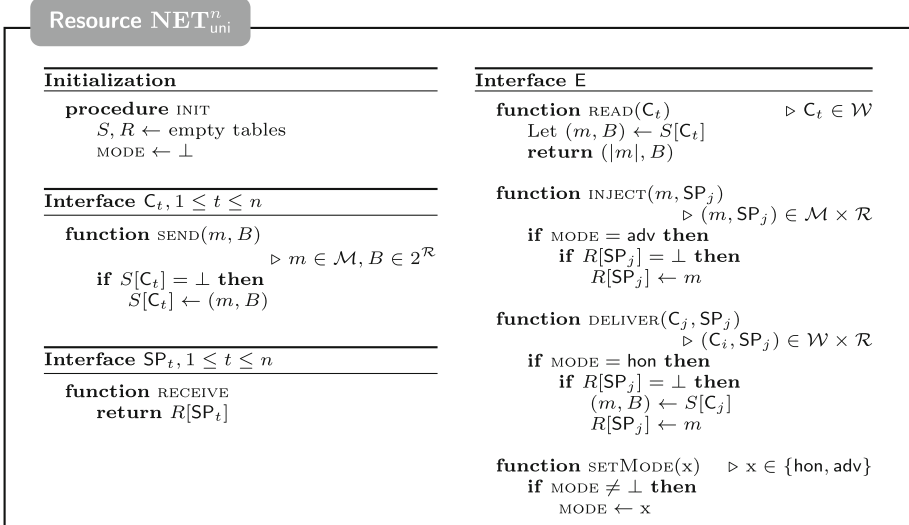


Fig. 12. The unilaterally secure network resource: the adversarial interface E can choose whether the network runs in *secure* ($\text{MODE} = \text{hon}$) or *adversarial* mode ($\text{MODE} = \text{adv}$).

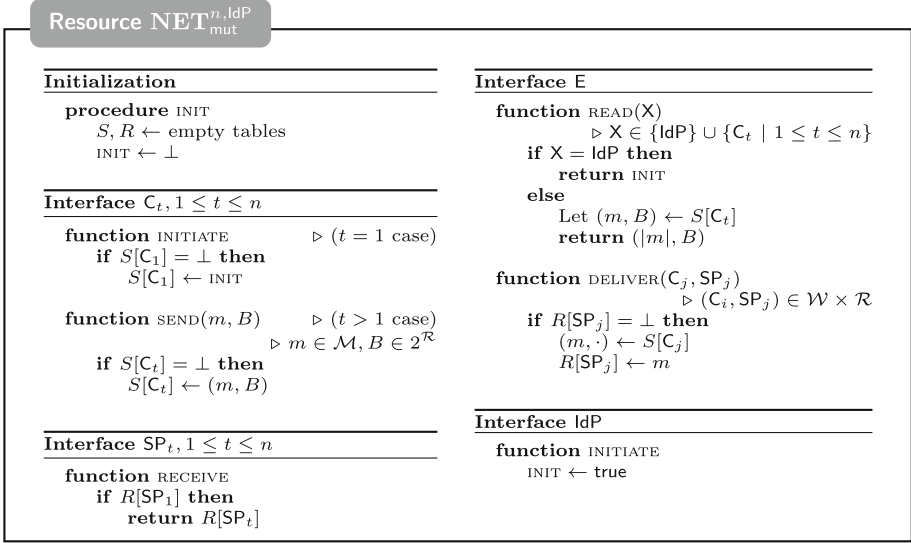


Fig. 13. In a mutually secure network between a client and a service provider, the adversary can only deliver messages between the client and the service provider.

Modeling session authentication of SSO schemes. In a typical single-sign-on use case, the clients have a unilaterally authenticated session with the service provider. Aside of that they have establish a mutually authenticated and secure session with the identity provider. In practice, such a session is authenticated using a secure channel protocol involving an authentication based on passwords, hardware tokens, or one-time codes. In short, there is a secure channel between the identity provider and the client denoted by $\text{SEC}_{\text{IdP}, C_1}$. Aside of this assumed channel, we again need a mechanism to distribute the verification key of the identity provider using an authenticated channel between the identity provider and the service provider. which we denote by $\text{Ch}_{\text{IdP} \rightarrow \text{SP}}$. The client realizes a mutually authenticated secure channel $\text{NET}_{\text{mut}}^{n, \text{IdP}}$ by relaying a signed message (and its signature string), i.e., *the assertion*, from the identity provider to the service provider (this is usually denoted as IdP-initiated scenario in SSO terms) and have the signature verified by the service provider. In more detail, the protocol converter `assert` for the identity provider distributes its verification key and signs a specific token and sends it to the client via the assumed secure channel. The client converter `fwd` forwards this token to the service provider. Finally, the converter of the service provider, denoted `filter`, only starts outputting messages once the token is received and verified as the first message from $\text{NET}_{\text{uni}}^n$. Note that we treat all interfaces SP_i as sub-interfaces of one service provider interface SP. We establish the following theorem:

Theorem 6. *The protocol described above (and formally specified as pseudo-code in the full version), consisting of the service provider protocol filter, the*

identity provider protocol `assert`, and the client protocol `fwd` , constructs the mutually secure network $\mathbf{NET}_{\text{mut}}^{n, \text{IdP}}$, from the assumed, unilaterally secure, setting $[\mathbf{Ch}_{\text{IdP} \rightarrow \text{SP}}, \mathbf{SEC}_{\text{IdP}, \text{C}_1}, \mathbf{NET}_{\text{uni}}^n]$. More specifically, there is a simulator `sim` (formally specified as pseudo-code in the full version of this work) such that for any distinguisher \mathbf{D} , there is an attacker \mathbf{A} against the underlying signature scheme (with essentially the same efficiency), i.e.,

$$\Delta^{\mathbf{D}}(\text{fwd}^{\text{C}_1} \text{assert}^{\text{IdP}} \text{filter}^{\text{SP}} [\mathbf{Ch}_{\text{IdP} \rightarrow \text{SP}}, \mathbf{SEC}_{\text{IdP}, \text{C}_1}, \mathbf{NET}_{\text{uni}}^n], \tilde{\text{sim}} \mathbf{NET}_{\text{mut}}^{n, \text{IdP}}) \leq \Gamma^{\mathbf{A}}(\mathbf{G}_{\Sigma}^{\text{EU-CMA}}).$$

Proof. The proof is given in the full version [3] and follows a similar idea to the one of the previous section. \square

The approach of sending assertions to upgrade a unilaterally authenticated channel to full authentication is used, for instance, in the widely used SAML protocol [16]. This section can be seen as a proof of an abstract version of SAML.

References

1. Backes, M., Hofheinz, D.: How to break and repair a universally composable signature functionality. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 61–72. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30144-8_6
2. Backes, M., Pfitzmann, B., Waidner, M.: A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015, January 2003. <http://eprint.iacr.org/2003/015>
3. Badertscher, C., Maurer, U., Tackmann, B.: On composable security for digital signatures. Cryptology ePrint Archive, Report 2018/015 (2018). <https://eprint.iacr.org/2018/015>. (full version of this paper)
4. 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
5. Camenisch, J., Enderlein, R.R., Krenn, S., Küsters, R., Rausch, D.: Universal composition with responsive environments. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 807–840. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_27
6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, version of October 2001. <http://eprint.iacr.org/2000/067>
7. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings of the 42nd Symposium on Foundations of Computer Science, pp. 136–145. IEEE (2001)
8. Canetti, R.: Universally composable signature, certification and authentication. In: Proceedings of CSFW 2004 (2004)
9. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, version of December 2005. <http://eprint.iacr.org/2000/067>

10. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_4
11. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_16
12. Canetti, R., Shahaf, D., Vald, M.: Universally composable authentication and key-exchange with global PKI. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 265–296. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_11
13. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_24
14. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)
15. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)
16. Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., Maler, E.: Profiles for the Security Assertions Markup Language (SAML). OASIS Standard, March 2015
17. Küsters, R., Tuengerthal, M.: Joint state theorems for public-key encryption and digital signature functionalities with local computation. In: Proceedings of 21st IEEE Computer Security Foundations Symposium, CSF 2008 (2008)
18. Lamport, L.: Constructing digital signatures from a one-way function. Technical report CSL1998, SRI International, Menlo Park, California, October 1979
19. Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_8
20. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27375-9_3
21. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_8
22. Maurer, U., Renner, R.: Abstract cryptography. In: ICS, pp. 1–21 (2011)
23. Maurer, U., Renner, R.: From indifferentiability to constructive cryptography (and back). In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9985, pp. 3–24. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53641-4_1
24. Maurer, U., Tackmann, B., Coretti, S.: Key exchange with unilateral authentication: composable security definition and modular protocol design. *Cryptology ePrint Archive*, Report 2013/555, September 2013
25. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp. 184–200. IEEE (2001)
26. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)

Oblivious Transfer



Equational Security Proofs of Oblivious Transfer Protocols

Baiyu Li^(✉) and Daniele Micciancio

University of California, San Diego, USA
{baiyu,daniele}@cs.ucsd.edu

Abstract. We exemplify and evaluate the use of the equational framework of Micciancio and Tessaro (ITCS 2013) by analyzing a number of concrete Oblivious Transfer protocols: a classic OT transformation to increase the message size, and the recent (so called “simplest”) OT protocol in the random oracle model of Chou and Orlandi (Latincrypt 2015), together with some simple variants. Our analysis uncovers subtle timing bugs or shortcomings in both protocols, or the OT definition typically employed when using them. In the case of the OT length extension transformation, we show that the protocol can be formally proved secure using a revised OT definition and a simple protocol modification. In the case of the “simplest” OT protocol, we show that it cannot be proved secure according to either the original or revised OT definition, in the sense that for any candidate simulator (expressible in the equational framework) there is an environment that distinguishes the real from the ideal system.

1 Introduction

Cryptographic design and analysis is a notoriously hard problem, arguably even harder than standard software design because it requires to build systems that behave robustly in the presence of a malicious adversary that actively tries to subvert their execution. The desirability of precise formalisms to describe and analyze cryptographic constructions is well exemplified by the code-based game-playing framework of [4] to present security definitions and proofs of standard cryptographic functions. But even the detailed framework of [4] offers little help when formalizing more complex cryptographic protocols, due to their interactive nature and underlying distributed execution model. At the semantic level, the gold standard in secure computation protocol design and analysis is the *universally composable (UC)* security model of [5] (or one of its many technical variants [1, 2, 7, 9, 16, 17, 21]), which offers strong compositionality guarantees in fully asynchronous execution environments like the Internet. Unfortunately, the relative lack of structure/abstraction in the traditional formulation of this

This work was supported in part by NSF grant CNS-1528068. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

model¹ makes it rather hard to use in practice, when specifying and analyzing concrete protocols.² These limitations are widely recognized, and have prompted researchers to explore several variants, simplifications and specialization of the general UC security model [6, 19, 22, 30]. In this perspective, a very interesting line of work is represented by the “abstract cryptography” framework of [25], which calls for an axiomatic approach to the description and analysis of cryptographic primitives/protocols, and the “constructive cryptography” [24] and “equational security” [26] frameworks, which can be thought of as logical models of the axioms put forward in [25].

In this work we examine the equational security framework of [26], which provides both a concrete mathematical model of computation/communication, and a concise syntax to formally describe distributed systems by means of a set of mathematical equations. We believe that progress in our ability to describe and analyze cryptographic protocols cannot be achieved simply by formulating frameworks and proving theorems in definitional papers, but it requires putting the frameworks to work on actual example protocols. To this end, we present a detailed case-study where we evaluate the expressiveness and usability of this framework by analyzing a number of concrete *oblivious transfer* protocols, a simple but representative type of security protocols of interest to cryptographers.

Oblivious transfer (OT), in its most commonly used 1-out-of-2 formulation [12], is a two party protocol involving a sender transmitting two messages m_0, m_1 and a receiver obtaining only one of them m_b , in such a way that the sender does not learn which message $b \in \{0, 1\}$ was delivered and the receiver does not learn anything about the other message m_{1-b} . OT is a classic example of secure computation [12, 27], and an important (in fact, complete) building block for the construction of arbitrary security protocols [11, 14, 18, 20, 23, 32]. In Sects. 3 and 4 we investigate a well known transformation often used to increase the message length of OT protocols with the help of a pseudorandom generator. In Sect. 5, we investigate a very efficient OT protocol in the random oracle model recently proposed in [10].

We remark that the primary goal of our work is to exemplify and evaluate the usability of the equational security framework of [26], rather than finding and fixing bugs in specific protocol instances. Still, our findings about the OT protocols under study may be of independent interest, and well illustrate how equational security modeling can offer a convenient and valuable tool for cryptographic protocol specification and analysis. The main findings about the OT protocols are the following:

- The security of the OT protocol transformation, often considered a folklore result in cryptography, does not hold with respect to the naive OT definition

¹ Rooted in computational complexity, the model is usually described as an arbitrary network of (dynamically generated) Turing machines that communicate by means of shared tapes, possibly under the direction of some scheduling process, also modeled as an interactive Turing machine.

² This is analogous to the Turing machine, an excellent model to study computation in general but a rather inconvenient one when it comes to specifying actual algorithms.

typically used (often implicitly) in the cryptographic literature. However, if the OT ideal functionality definition is suitably modified, then the transformation becomes provably secure, and can be readily analyzed using simple equational reasoning.

- The protocol of [10] can be proved secure according to *neither* the classic *nor* the revised OT definitions considered above.

Technical details about our findings, and general comments/conclusions are provided in the next paragraphs.

1.1 Oblivious Transfer Extension

The standard definition of OT is given by a functionality $\text{OT}((m_0, m_1), b) = m_b$ that takes a pair of messages (m_0, m_1) from the sender, a selection bit b from the receiver, gives m_b to the receiver, and gives nothing to the sender. The two messages are assumed to have the same length $|m_0| = |m_1| = \kappa$, which is usually tied to the security parameter of the scheme and the mathematical structures used to implement it. (E.g., $\kappa = \log |H|$ where H is the domain/range of some group-theoretic cryptographic function). A natural and well known method to adapt such OT protocol to one allowing the transmission of longer messages is the following:

1. Use an underlying OT protocol to send two random seeds (s_0, s_1) of length κ ,
2. Use these seeds as keys to encrypt the two messages using a private-key encryption scheme,³ and send both ciphertexts to the receiver over a standard (authenticated, but insecure to eavesdropping) communication channel.

The intuition is that since the receiver gets only one of the two keys, the other message is protected by the encryption scheme. Indeed, the intuition is correct, in the sense that encryption does its job and protects the other message, but the protocol is nevertheless not secure (at least, according to the simulation-based fully asynchronous security definition implied by the OT functionality described above). Our formal analysis shows that, while the protocol is correct, and secure against corrupted senders, it is not secure against corrupted receivers, and for a very simple reason: it contains a subtle timing bug! In a real execution, the sender transmits the encryption of its two messages as soon as the two messages are made available by the environment. However, the simulator can produce the corresponding simulated ciphertexts only after the receiver has chosen her selection bit b . In order to prove security, the sender should delay the transmission of the ciphertexts until after the receiver has provided b to the underlying OT protocol. The problem is that the above OT ideal functionality does not disclose any information to the sender, not even if and when the receiver has selected the bit b .

³ Since each seed s_i is used only once, the secret key encryption scheme can be as simple as stretching s_i using a pseudorandom generator \mathcal{G} , and use the resulting string $\mathcal{G}(s_i)$ as a one-time pad to mask the message m_i .

We also consider a revised OT definition $\text{OT}((m_0, m_1), b) = (f(b), m_b)$, that includes an additional output $f(b) \in \{\perp, \top\}$ disclosing to the sender if b has been chosen yet, without providing the actual value of $b \in \{0, 1\}$. We modify the protocol accordingly (by letting the sender delay the transmission of the ciphertexts until $b > \perp$), and show that the modified protocol can be formally proved secure according to the revised OT definition.

1.2 Oblivious Transfer in the Random Oracle Model

In [10], Chou and Orlandi propose a new OT protocol achieving UC security in the random oracle model [3]. The protocol is very elegant and can be efficiently implemented based on elliptic curve groups. We provide a formal analysis of the protocol using the equational framework. We show that if the naive OT definition is used, then the protocol is insecure against both corrupted senders and corrupted receivers. For the case of corrupted senders, the failure of simulation is due to the fact that in a real protocol execution the sender learns if and when the receiver provides her selection bit b , which is not available to the simulator. For the case of corrupted receivers, the problem is that in a real protocol execution the receiver can delay its random oracle query until after seeing the sender's ciphertexts, but in the ideal protocol execution, if the simulator has to output the ciphertexts before seeing the receiver's random oracle query, then it must be able to guess an external random bit correctly before seeing any inputs, which is impossible to achieve with high probability. However, unlike the case of the OT length extension transformation, these problems are not the only weakness of the protocol, and security cannot be proved by switching to the revised OT definition given above.

1.3 Discussion/Conclusions

Before jumping to conclusions, some remarks about the significance of our results are in order. As already noted, it should be understood that the aim of our work was to illustrate the use of the equational framework, rather than criticizing any specific protocol or definition. In particular, we are not arguing that the revised OT definition given in Sect. 4 is the “correct” one, and everybody should use it. In fact, other alternative definitions are possible. Our main point is that the equational model is a convenient framework to precisely formulate and investigate alternative definitions.

The OT message length transformation studied in Sect. 3 is folklore. We are not aware of any work analyzing its security, and our study is, to the best of our knowledge, the first work even making a formal security claim about it. This is perhaps because doing this using the traditional framework based on the informal use of interactive Turing machines already seemed cumbersome and error prone enough not to be worth the effort. In fact, the transformation is simple enough that at first it is natural to wonder if a formal proof of security is required at all. Our analysis shows that a formal security proof is indeed useful, at very least to unambiguously identify the security property (ideal functionality) for which

the transformation is (proved or claimed to be) correct. We remark that when we set to analyze the OT protocol transformation, we were giving for granted that the transformation was secure, and the analysis was meant primarily as a simple example to illustrate the use of the equational framework. Finding that the protocol does not emulate the traditional OT definition came to us as a surprise, even if in hindsight the timing bug is rather obvious. In this respect, the equational framework proved to be a very convenient tool to carry out a precise formal analysis with relatively modest effort.

As for the protocol of [10], our primary aim is to illustrate the use of the equational framework to analyze a protocol in the random oracle model. We are certainly not concerned about whether the protocol is making a “morally correct” use of the random oracle, or if a “global” random oracle definition [8] should be used instead. We simply use the equational framework to model and analyze the protocol as described in the original paper [10]. Our analysis shows that the protocol is not secure according to the original OT definition (seemingly used in [10]), but even using a revised OT definition still does not allow to prove security in the equational framework, in the technical sense that for any simulator (expressible in the equational framework) there is an environment that distinguishes between the real and the ideal systems.

We believe our analysis highlights the importance of a more rigorous proof style when analyzing secure computation protocols than currently feasible using traditional formulations of the UC framework and its variants. This is especially important when it comes to formally specifying the security properties satisfied (or claimed) by a protocol. Without an unambiguous formal security specification/claim, even the most detailed proof is of little value, as it is not clear what is being proved or claimed. Within the context of our work, the equational framework of [26] proved to be a very convenient and useful formalism to express security definitions (in the form of ideal functionalities) and cryptographic protocols in a concise, yet mathematically precise way. It allowed us to easily explore different definitional variants and put them to good use to spot potential bugs in cryptographic protocols. Exploring the applicability of abstract frameworks along the lines of [24–26] to the specification and analysis of a wider range of cryptographic protocols is likely to be mutually beneficial, both to further develop and refine the models, and to gain useful insight on the security of concrete cryptographic protocols.

2 Background and Notation

In this section we review the equational framework of [26], and define the notation used in this paper. For completeness, we will first recall some background on the (standard) theory that gives a precise meaning to systems of equations as used in [26] and in this paper. This material is important to give a solid mathematical foundation to the equational framework, but is not essential to follow the rest of the paper, and the reader may want to skip directly to the following paragraph describing our computational models and notational conventions.

2.1 Domain Theoretical Background

The mathematical foundation of the equational framework is provided by domain theory. Here we give just enough background to describe the systems studied in this paper, and refer the reader to [15, 28, 29] for a detailed treatment. Recall that a partially ordered set (or poset) is a set X equipped with a reflexive, transitive and antisymmetric relation \leq . All posets in this paper are complete partial orders (CPOs), i.e., any (possibly empty) chain $x_1 < x_2 < \dots$ has a least upper bound $\sup_i x_i$ in X . The Cartesian product $X \times Y$ of two CPOs is also a CPO with the component-wise partial order $(x_1, y_1) \leq (x_2, y_2) \iff x_1 \leq x_2 \wedge y_1 \leq y_2$. These posets are endowed with the *Scott topology*, where a subset $C \subseteq X$ is *closed* if for all $x \in C$, $y \leq x$ implies $y \in C$, and any chain in C has a least upper bound in C . A set is *open* if its complement is closed. The standard topological definition of *continuous function* still applies here, and continuous functions (with respect to the Scott topology) are exactly the functions that preserve limits $f(\sup_i x_i) = \sup_i f(x_i)$. The set of all continuous functions from CPOs X to Y is denoted by $[X \rightarrow Y]$. Any (Scott) continuous function is necessarily *monotone*, i.e., for all $x, y \in X$, if $x \leq y$ then $f(x) \leq f(y)$. All CPOs X have a minimal element $\perp = \sup \emptyset$, called the *bottom*, which satisfies $\perp \leq x$ for all $x \in X$.

For any set A , we can always construct a *flat* CPO $A_\perp = A \cup \{\perp\}$ by including a unique bottom element \perp . The partial order in A_\perp consists of $\perp \leq x$ for all $x \in A$. It should be easy to see that all nonempty closed sets in A_\perp contain \perp , and open sets in A_\perp are exactly the subsets of A and the whole A_\perp . Functions $f: A \rightarrow B$ between sets can be lifted to strict functions $f: A_\perp \rightarrow B_\perp$ between the corresponding flat CPOs by setting $f(\perp) = \perp$. The bottom element usually designates the situation where no (real) input or output is given yet.

For any CPO X , every continuous functions $f: X \rightarrow X$ admits a *least fixed point*, denoted as $\text{fix}(f)$, which is the minimal $x \in X$ such that $f(x) = x$. The least fixed point can be obtained by taking the limit of the sequence $\perp, f(\perp), f^2(\perp), \dots$. A system of mutually recursive equations can be solved via least fixed point computation. Such a solution describes the final outputs of interactive computations between nodes in a network. By Bekič’s theorem [31], the least fixed point of such a system can be computed one component at a time: For example, the system $(x, y) = (f(x, y), g(x, y))$ can be solved by computing first $\hat{x} = \text{fix}(\lambda x. f(x, \text{fix}(\lambda y. g(x, y))))$ and then $\hat{y} = \text{fix}(\lambda y. g(\hat{x}, y))$, and the least fixed point of the system is (\hat{x}, \hat{y}) .

We can also model probabilistic behaviors in equational framework. A *probability distribution* on a CPO X is a function $p: X \rightarrow [0, 1]$ such that⁴ $p(A) + p(B) = p(A \cup B)$ for all disjoint $A, B \subseteq X$ and $p(X) = 1$. As usual, we say that a probability p is *negligible* if for all $x \in X$, $p(x) < n^{-c}$ for any

⁴ In general we should consider the Borel algebra on X when defining probability distributions on X . Here we simply use X instead since we work on finite sets and discrete probabilities.

constant $c > 1$, where n is a *security parameter*.⁵ Similarly, p is *overwhelming* if $1 - p$ is negligible. If X is a CPO, then the *set of probability distributions over X* , denoted by $D(X)$, is also a CPO, where for any two distributions $p \leq q$ (in $D(X)$) if and only if $p(A) \leq q(A)$ for any open subset $A \subseteq X$. *Probabilistic functions* are just (continuous) functions between sets of distributions with respect to this ordering relation.

2.2 Computational Model

We recall that the execution model of [26] consists of a network, with nodes representing computational units, and (directed) edges modeling communication channels. (See below for details.) Each channel is associated with a partially ordered set of channel “histories” or “behaviors”, representing all possible messages or sequences of messages that may be transmitted on the channel over time. The partial order represents temporal evolution, so for any two histories $h_1 \leq h_2$ means that h_2 is a possible extension (or future) of h_1 . The standard example is that of finite sequences $M^* = \{(m_1, \dots, m_k) : k \geq 0, \forall i. m_i \in M\}$ of messages from a ground set M , ordered according to the prefix partial order. By combining the set M^∞ of infinite sequences of messages from M , we get a CPO M^ω . Another common example, modeling a channel capable of delivering only a single message, is the flat partial order M_\perp , consisting of all messages in M and a special bottom element \perp denoting the fact that no message has been transmitted yet. Different incoming and outgoing channels (incident to a single node) are combined taking Cartesian products, so that each node can be thought as having just one input and one output. The computational units at the nodes are modeled as functions $F: X \rightarrow Y$ from the incoming channels to the outgoing channels, satisfying the natural monotonicity requirement that for any $h_1 \leq h_2$ in X , we have $F(h_1) \leq F(h_2)$ in Y . Informally, monotonicity captures the intuition that once a party transmits a message, it cannot go back in time and take it back. A probabilistic computational unit can be modeled as a function of type $X \rightarrow \mathcal{D}(Y)$, where \mathcal{D} is the probability monad. We may also consider units with limited computational power in the monadic approach, which is an important extension to the equational framework. However, as all the protocols considered in this paper run in constant time, for simplicity we do not formalize computational cost (e.g. running time, space, etc.) in our analysis.

Computation units can be connected to a communication network N to form a system, where N is also a monotone function. Such a system is again a monotone function mapping external input channels to external output channels of all the units, and it is modeled as a composition of functions describing all the units and the network. Syntactically, function compositions can be simplified by substitution and variable elimination, and, when recursive definition is involved, by using fixed point operations. In general, we use the notation $(F|G)$ to denote

⁵ In the asymptotic setting, cryptographic protocols are parameterized by a security parameter n . For notational simplicity, we consider this security parameter n as fixed throughout the paper.

the system composed by functions F and G , where the composition operator “ \mid ” is associative. The main advantage of the equational framework is that it has a mathematically clean and well defined semantics, where functions can be completely described by mathematical equations (specifying the relation between the input and the output of the units), and composition simply combines equations together. The equational approach also provides a simple and precise way to reason about relations between systems. For example, equivalent components (in the sense of having equivalent equations) can be replaced by each other, and when considering probabilistic behaviors, if a component is indistinguishable from another component, then they can be used interchangeably with negligible impact on the behavior of the entire system.

2.3 Security

The definition of security in the equational framework follows the well-accepted simulation-based security paradigm. In this paper we consider only OT protocols, which are two-party protocols between a sender program and a receiver program. An ideal functionality F is a function from $X = X_0 \times X_1$ to $Y = Y_0 \times Y_1$, where X_i (Y_i) is the external input (output) of party P_i . An environment is a function $\text{Env} : Y^\omega \rightarrow X^\omega \times \{\top\}_\perp$ such that it takes as input the output history (as a sequence of evolving messages) of a system, and it produces a sequence of evolving inputs to the system and a decision bit t . Here a sequence of messages $x_0x_1\dots$ over X is *evolving* if $x_i \leq_X x_{i+1}$ for all i , where $x_i \in X$ and \leq_X is the partial order of X . An experiment between an environment Env and a system S , is executed as follows: Env generates an evolving sequence of input $x_0x_1\dots$ to S such that S outputs $y_i = S(x_i)$ for each x_i , Env takes as input the sequence $y_0y_1\dots$, and it eventually produces an external decision bit t . We write $\text{Env}[S]$ for the output (distribution) t of this experiment. When all parties are honest, the real system is a composition of the network N and two parties P_0 and P_1 , denoted as $(P_0|P_1|N)$, and it must be equivalent to the ideal functionality F . When a party P_i is *corrupted*, the real system is composed by the remaining honest party and the network, and the ideal system is composed by F and a monotone simulator Sim . We say that a protocol is secure against the corruption of P_i if there exists a simulator Sim as a computation unit such that the systems $(N|P_{(1-i)})$ and $(\text{Sim}|F)$ are indistinguishable by any environment that produces a decision bit in polynomial time in the output length of the system and the security parameter.

A distinctive feature of the equational framework is the ability to specify fully asynchronous systems. An environment might not provide a complete input to a system at once, that is, the input to certain channels might be \perp . So we must consider such asynchronous environments when analyzing the security of a protocol.

It is an very interesting and important open question to compare the equational framework (with the full extension of computational security) with the UC model and its variants (for example, the simplified models of [6, 30]). Due to

space limitation, we do not address such a problem in the current paper and we will study it in future work.

2.4 Notation

Now we briefly mention our notational conventions. In this paper we mainly use flat CPOs, i.e., partially ordered sets \mathbb{X} with a bottom element $\perp \in \mathbb{X}$ such that $x_1 \leq x_2$ iff $x_1 = \perp$ or $x_1 = x_2$. These are used to model simple communication channels that can transmit a single message from $\mathbb{X} \setminus \{\perp\}$, with \perp representing the state of the channel before the transmission of the message. For any CPO \mathbb{X} , we write $\mathbb{X}^{\times 2} = \{(x, y) : x, y \in \mathbb{X}, x \neq \perp, y \neq \perp\}_{\perp}$ for the CPO of *strict* pairs over \mathbb{X} and \perp . The elements of a pair $z \in \mathbb{X}^{\times 2}$ are denoted $z[0]$ and $z[1]$, with $z[i] = \perp$ when $z = \perp$ or $i = \perp$. The operation of combining two elements into a strict pair is written $\langle x, y \rangle$. Notice that $\langle x, \perp \rangle = \langle \perp, y \rangle = \perp$, and therefore $\langle x, \perp \rangle[0] = \langle \perp, y \rangle[1] = \perp$ even when $x, y \neq \perp$. For any set A , we write $x \leftarrow A_{\perp}$ for the operation of selecting an element $x \neq \perp$ uniformly at random from A .

It is easily verified that for any pairs $z, \langle x_0, x_1 \rangle, \langle y_0, y_1 \rangle$, strict function f and strict binary operation \odot ,

$$z = \langle z[0], z[1] \rangle \tag{1}$$

$$f(\langle x_0, x_1 \rangle[i]) = \langle f(x_0), f(x_1) \rangle[i] \tag{2}$$

$$\langle x_0, x_1 \rangle[i] \odot \langle y_0, y_1 \rangle[i] = \langle x_0 \odot y_0, x_1 \odot y_1 \rangle[i] \tag{3}$$

The followings are common CPOs and operations:

- The CPO $\mathbb{T} = \{\top\}_{\perp}$, representing *signals*, i.e., messages with no information content.
- The CPO $\mathbb{B} = \{0, 1\}_{\perp}$ of single bit messages, often used to select an element from a pair.
- The CPO $\mathbb{M}_n = \{0, 1\}_{\perp}^n$ of bit-strings of length n .
- $x!y = \langle x, y \rangle[1]$, the operation of *guarding* an expression y by some other expression x . Notice that $x!y = y$, except when $x = \perp$, and can be used to “delay” the transmission of y until after x is received.
- $x! = x!\top$, testing that $x > \perp$.

As an example, using the notation introduced so far, we can describe the ideal (1-out-of-2) OT functionality by the equations in Fig. 1. (Notice that this functionality is parameterized by a message space \mathbb{M}). The first line specifies the names of the functionality (OT), input channels (m_2, b) and output channel(s) m . This is followed by a specification of the type of each channel: the input interface includes a message pair $m_2 = \langle m_0, m_1 \rangle \in \mathbb{M}^{\times 2}$ from a sender and a selection bit $b \in \mathbb{B}$ from a receiver. The output interface is a single message $m \in \mathbb{M}$ sent to the receiver while the sender does not get any information from the functionality. The last line $m = m_2[b]$ is an equation specifying the value of the output channel(s) as a function of the input channels. The functionality is illustrated by a diagram showing the names of the function and the input/output channels.

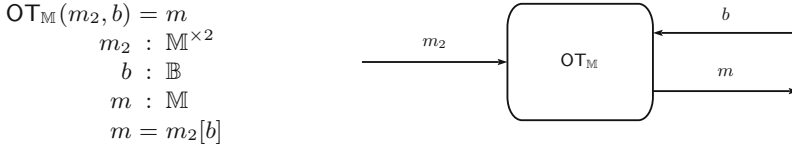


Fig. 1. A naive OT functionality: the receiver gets the selected message $m = m_2[b]$, and the sender does not get anything at all.

In the rest of this paper, equational variables usually belong to unique domains (e.g., $m_2 : \mathbb{M}_n^{\times 2}$). So from now on, we will omit such type specifications when defining functions using equations, and we will follow the convention listed in Table 1 for naming variables.

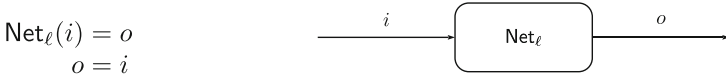
Table 1. Frequently used variables and their domains.

Variable name	Domain	Variable name	Domain
m	\mathbb{M}_n	m'	\mathbb{M}_ℓ
m_2	$\mathbb{M}_n^{\times 2}$	m'_2	$\mathbb{M}_\ell^{\times 2}$
c_0, c_1	\mathbb{M}_ℓ	c_2	$\mathbb{M}_n^{\times 2}$
a, a'	\mathbb{T}	b, b'	\mathbb{B}
i, o	\mathbb{M}_n	i_2, o_2	$\mathbb{M}_\ell^{\times 2}$
k	\mathbb{K}_n	k_2	$\mathbb{K}_n^{\times 2}$
q	$(G^2 \times G)_\perp$	q_2	$(G^2 \times G)_\perp^{\times 2}$
X, Y	G_\perp		

3 Oblivious Transfer Length Extension: A First Attempt

As an abbreviation, when the message space $\mathbb{M} = \{0, 1\}_\perp^n$ is the set of all bit-strings of length n , we write OT_n instead of $\text{OT}_{\mathbb{M}}$. Consider the following OT *length extension* problem: given an OT_n channel for messages of some (sufficiently large) length n , build an OT functionality OT_ℓ for messages of length $\ell > n$. The goal is to implement OT_ℓ making a single use of the basic OT_n functionality, possibly with the help of an auxiliary (unidirectional, one-time) communication channel for the transmission of messages from the sender to the receiver. For simplicity,⁶ we model the communication channel as a functionality Net_ℓ that copies its input of length ℓ to the output of the same length:

⁶ This corresponds to a perfectly secure communication channel. More complex/realistic communication channels are discussed at the end of this section.



The OT length extension protocol is specified by a pair of Sender and Receiver programs, which are interconnected (using the OT_n and $Net_{2\ell}$ functionalities) as shown in Fig. 2. Notice how the external input/output interface of the system corresponding to a real execution of the protocol in Fig. 2 is the same as that of the ideal functionality $OT_\ell(m'_2, b') = m'$ the protocol is trying to implement.

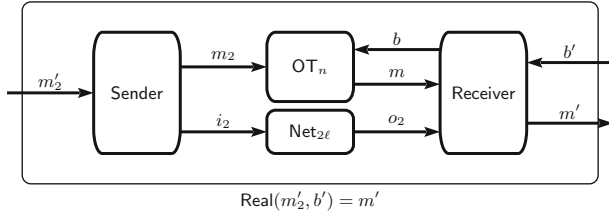
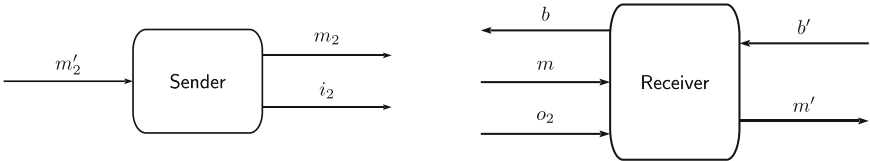


Fig. 2. A real execution of a candidate OT length extension protocol. The protocol consists of a Sender and a Receiver programs that communicate using OT_n and $Net_{2\ell}$ functionalities.

A natural approach to design an OT length extension protocol is to make use of a pseudorandom generator $\mathcal{G} : \mathbb{M}_n \rightarrow \mathbb{M}_\ell$ that stretches a short random seed of length n into a long pseudorandom string of length ℓ . Using such pseudorandom generator, one may define candidate Sender and Receiver programs as follows:

$$\begin{aligned}
 \text{Sender}(m'_2) &= (m_2, i_2) \\
 m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\
 i_2[0] &= m'_2[0] \oplus \mathcal{G}(m_2[0]) \\
 i_2[1] &= m'_2[1] \oplus \mathcal{G}(m_2[1])
 \end{aligned}
 \qquad
 \begin{aligned}
 \text{Receiver}(m, o_2, b') &= (b, m') \\
 b &= b' \\
 m' &= o_2[b'] \oplus \mathcal{G}(m)
 \end{aligned}$$



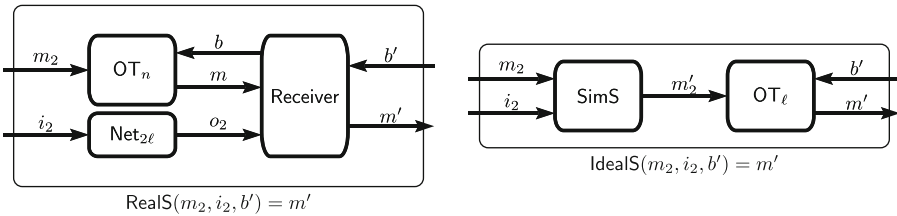
In words, these programs work as follows:

- The sender picks a pair m_2 of two random seeds, and passes (one of) them to the receiver using the OT_n functionality. It then stretches the two seeds using the pseudorandom generator \mathcal{G} , and uses the generator's output as a one-time pad to "mask" the actual messages before they are transmitted to the receiver over the communication channel $Net_{2\ell}$.

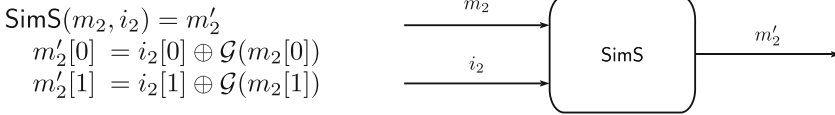
- The receiver selects one of the two seeds from the OT_n functionality, expands it using the pseudorandom generator, and uses the result to “unmask” the corresponding message from $Net_{2\ell}$.

It is easy to show that the protocol is correct, in the sense that combining the equations of OT_n , $Net_{2\ell}$, Sender and Receiver as shown in Fig. 2 results in a system $Real(m'_2, b') = m'$ that is perfectly equivalent to the defining equation $m' = m'_2[b']$ of the ideal functionality OT_ℓ . Intuitively, the protocol also seems secure because only one of the two seeds can be recovered by the receiver, and the unselected message is protected by an unpredictable pseudorandom pad. But security of cryptographic protocols is a notoriously tricky business, and deserves a closer look.

We first consider the security of the protocol when the sender is corrupted. The attack scenario corresponds to the real system obtained by removing the Sender program from the protocol execution in Fig. 2. Following the simulation paradigm, security requires exhibiting an efficient simulator program $SimS$ (interacting, as a sender, with the ideal functionality OT_ℓ) such that the following real and ideal systems are computationally indistinguishable:

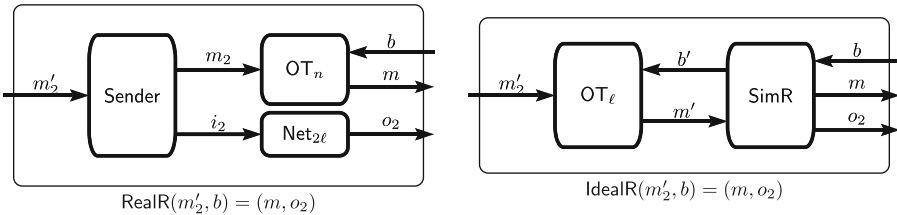


Security is easily proved by defining the following simulator:



We observe that $RealS$ and $IdealS$ are perfectly equivalent because they both simplify to $m' = i_2[b'] \oplus \mathcal{G}(m_2[b'])$. So, the protocol is perfectly secure against corrupted senders.

We now turn to analyzing security against a corrupted receiver. This time we need to come up with a simulator $SimR$ such that the following real and ideal executions are equivalent:

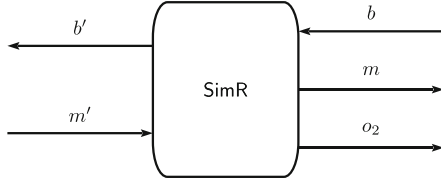


Of course, this time we can only aim at proving computational security, i.e., coming up with a simulator such that RealR and IdealR are computationally indistinguishable. We begin by writing down explicitly the equations that define the real system execution. Combining the equations for Sender , OT_n and $\text{Net}_{2\ell}$, we obtain the following system:

$$\begin{aligned} \text{RealR}(m'_2, b) &= (m, o_2) \\ m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\ o_2[0] &= m'_2[0] \oplus \mathcal{G}(m_2[0]) \\ o_2[1] &= m'_2[1] \oplus \mathcal{G}(m_2[1]) \\ m &= m_2[b] \end{aligned}$$

So, the simulator may proceed by picking m_0, m_1 at random on its own, and set $m = m_2[b]$ just as in the real execution. However, the simulator cannot compute o_2 as in RealR because it does not know m'_2 . This is addressed by using the same message m' twice, counting on the pseudorandom masking to hide this deviation from a real protocol execution. Formally, the simulator SimR is defined as follows:

$$\begin{aligned} \text{SimR}(m', b) &= (b', m, o_2) \\ b' &= b \\ m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\ m &= m_2[b] \\ o_2[0] &= m' \oplus \mathcal{G}(m_2[0]) \\ o_2[1] &= m' \oplus \mathcal{G}(m_2[1]) \end{aligned}$$



Combining SimR with OT_ℓ results in the ideal system:

$$\begin{aligned} \text{IdealR}(m'_2, b) &= (m, o_2) \\ m_2 &\leftarrow \mathbb{M}_n^{\times 2} \\ o_2[0] &= m'_2[b] \oplus \mathcal{G}(m_2[0]) \\ o_2[1] &= m'_2[b] \oplus \mathcal{G}(m_2[1]) \\ m &= m_2[b] \end{aligned}$$

As expected, the two systems IdealR , RealR are indistinguishable for both $b = 0$ and $b = 1$. For example, $\text{RealR}(m'_2, 0)$ and $\text{IdealR}(m'_2, 0)$ are equivalent because they are both computationally indistinguishable from the process that chooses $m \leftarrow \mathbb{M}_n$ and $c \leftarrow \mathbb{M}_\ell$ at random and sets $o_2 = \langle m'_2[0] \oplus \mathcal{G}(m), c \rangle$. The case when $b = 1$ is similar. At this point it would be very tempting to conclude that RealR and IdealR are equivalent, but they are not: they can be easily distinguished by an environment that sets $m'_2 \neq \perp$ and $b = \perp$. In fact, $\text{IdealR}(m'_2, \perp) = (\perp, \perp)$, but $\text{RealR}(m'_2, \perp) = (\perp, o_2)$, where $o_2 \neq \perp$. So, IdealR and RealR are not equivalent, and the simulator SimR is not valid.

Insecurity in general. By generalizing the above idea, we can show that, for any simulator SimR there is an environment Env that can distinguish the two systems RealR and IdealR with nonnegligible probability. We build Env that works in two stages:

$$\begin{aligned} \text{Env}_0(m, o_2) &= (b, m'_2, t) \text{ where} \\ & b = \perp, m'_2 \leftarrow \mathbb{M}_n^{\times 2}, t = (o_2 > \perp) \\ \text{Env}_1(m, o_2) &= (b, m'_2, t) \text{ where} \\ & b \leftarrow \{0, 1\}, m'_2 \leftarrow \mathbb{M}_n^{\times 2}, t = (\mathcal{G}(m) + o_2[b] = m'_2[b]) \end{aligned}$$

Notice that the output of the ideal system $\text{IdealR}(m'_2, b) = (m, o_2)$ is defined by $(b', m, o_2) \leftarrow \text{SimR}(m'_2[b'], b)$, where b' is an internal channel. Since b' ranges over a flat CPO, and $m'_2[\perp] = \perp$, the value of b' resulting from a least fixed point computation is given by $(b', _, _) = \text{SimR}(\perp, b)$. In particular, b' may depend only on the external input b . We denote using $\text{SimR}(b)^{b'}$ the random variable b' computed on input b .

Let $p = \Pr\{\text{SimR}(\perp)^{b'} = \perp\}$ and $q = \Pr\{\text{SimR}(\perp, \perp)^{o_2} = \perp\}$. It is clear that $\Pr\{\text{Env}_i[\text{RealR}] = \top\} = 1$ for all $i \in \{1, 2\}$. For the ideal system, we have

$$\begin{aligned} \Pr\{\text{Env}_0[\text{IdealR}] = \top\} &= \Pr\{\text{SimR}(\perp, \perp)^{o_2} > \perp\} \cdot p \\ &\quad + \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\} \cdot (1 - p) \\ &= (1 - q)p + \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\} \cdot (1 - p). \end{aligned}$$

Since $\Pr\{\text{Env}_0[\text{RealR}] = \top\} = 1$, $\Pr\{\text{Env}_0[\text{IdealR}] = \top\}$ must be overwhelming; and since $\Pr\{\text{SimR}(\perp, \perp)^{o_2} > \perp\} \leq \Pr\{\text{SimR}(\perp, m'_2[b'])^{o_2} > \perp\}$, p must be negligible. Finally, notice that

$$\begin{aligned} \Pr\{\text{Env}_1[\text{IdealR}] = \top\} &= \Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} = \perp\} \cdot p \\ &\quad + \Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} > \perp\} \cdot (1 - p). \end{aligned}$$

If $\text{SimR}(\perp)^{b'} > \perp$, then $\Pr\{b' = b\} = \frac{1}{2}$ and so

$$\Pr\{\mathcal{G}(m) + o_2[b] = m'_2[b] \mid \text{SimR}(\perp)^{b'} > \perp\} = \frac{1}{2}(1 + \frac{1}{2^\ell}).$$

This implies that $\Pr\{\text{Env}_2[\text{IdealR}] = \top\} = \frac{1}{2} + \epsilon$ for some negligible $\epsilon > 0$, and so Env can distinguish the two systems.

The discrepancy between the two systems as shown above highlights a subtle timing bug in the protocol: in order to carry out the simulation, the transmission of i_2 should be delayed until after the receiver has selected her bit b . However, this information is not available to the sender, and fixing the protocol requires revising the definition of OT , as we will do in the next section.

Other communication channels. We conclude this section with a discussion of other possible communication channels and weaker OT variants that leak some information to the environment. For example, one may replace the perfectly secure communication channel $\text{Net}_{\mathbb{M}}$ with an authenticated channel $\text{AuthNet}_{\mathbb{M}}(i, e_i) = (o, e_o)$ that also takes an input $e_i : \mathbb{T}$ and provides an output $e_o : \mathbb{M}$ to the environment. The environment output $e_o = i$ is used to leak the transmitted message as well as the timing information about when the message is transmitted. The environment input e_i is used to allow the environment to delay the transmission of the message $o = e_i!i$ to the receiver.

Similarly, one may consider the OT variants that leak the input timing information $e_o = (m_2!T, b!T)$ to the environment, and allow the environment to delay the OT output $m = e_i!m_2[b]$. This idea is similar to the “message header” in the UC models proposed in [6,30].

We remark that none of these modifications affect the analysis presented in this section. In particular, considering a perfectly secure communication channel *Net* only makes our insecurity result stronger. Also, leaking the signal $b!T$ to the environment does not solve the timing bug in the protocol: in order to fix the bug, the sender needs to delay the transmission of i_2 until $b > \perp$. So, it is not enough to provide this information to the environment. The timing signal $b!T$ needs to be provided as an input to the honest sender.

4 OT Length Extension

We have seen that the “standard” OT definition is inadequate even to model and analyze a simple OT length-extension protocol. In Fig. 3 we provide a revised definition of oblivious transfer that includes an acknowledgment informing the sender of when the receiver has provided her selection bit.

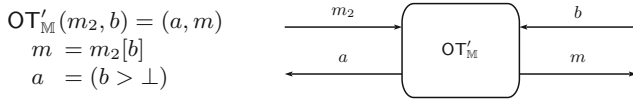
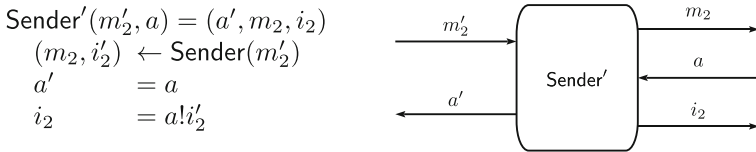


Fig. 3. A revised OT functionality.

We use this revised definition to build and analyze a secure OT length-extension protocol, similar to the one described in the previous section. The OT length extension uses the same Receiver program as defined in Sect. 3, but modifies Sender by using the signal a to delay the transmission of the message i_2 . The new Sender' also forwards the signal a to the environment to match the new OT' definition:



The Sender and Receiver programs are interconnected using OT'_n and $Net_{2\ell}$ as shown in Fig. 4. As in the previous section, it is easy to check that the protocol is correct, i.e., combining and simplifying all the equations from the real system in Fig. 4 produces a set of equations identical to the revised definition of the ideal functionality $OT'(m'_2, b') = (a', m')$. Security when the sender is corrupted is also similar to before. The real and ideal systems in this case are given by

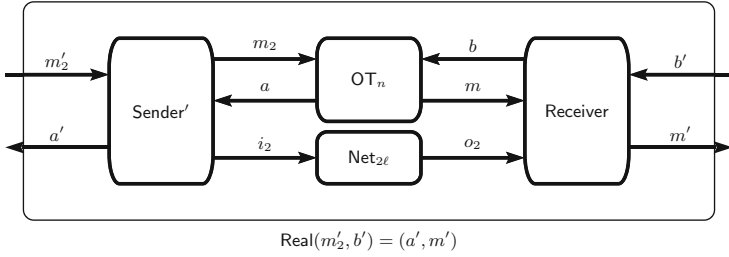
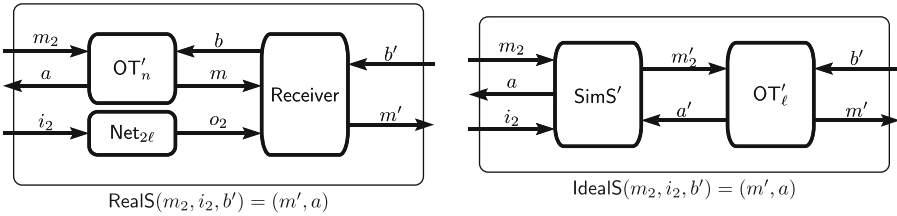
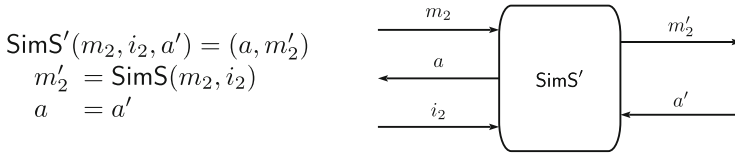


Fig. 4. A normal execution of the OT length extension protocol.

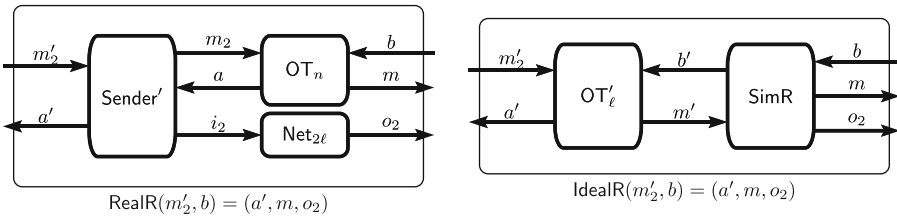


We see that this time SimS' has an additional input a' and output a . We adapt the simulator from the previous section simply by adding an equation that forwards the a' signal from OT' to the external environment:



$\text{RealS}(m_2, i_2, b')$ and $\text{IdealS}(m_2, i_2, b')$ are equivalent because they both output $m' = o_2[b'] \oplus \mathcal{G}(m_2[b'])$ and $a = (b' > \perp)$. So, the protocol is still perfectly secure against corrupted senders according to the revised OT' definition.

We now go back to the analysis of security against corrupted receivers. The real and ideal systems are:



No change to the simulator are required: we use exactly the same “candidate” simulator SimR as defined in Sect. 3. Combining and simplifying the equations, gives the following real and ideal systems:

$$\begin{array}{ll}
 \text{RealR}(m'_2, b) = (a', m, o_2) & \text{IdealR}(m'_2, b) = (a', m, o_2) \\
 m_2 \leftarrow \mathbb{M}_n^{\times 2} & m_2 \mathbf{n} \leftarrow \mathbb{M}_n^{\times 2} \\
 c_0 = m'_2[0] \oplus \mathcal{G}(m_2[0]) & c_0 = m'_2[b] \oplus \mathcal{G}(m_2[0]) \\
 c_1 = m'_2[1] \oplus \mathcal{G}(m_2[1]) & c_1 = m'_2[b] \oplus \mathcal{G}(m_2[1]) \\
 o_2 = b!(c_0, c_1) & o_2 = \langle c_0, c_1 \rangle \\
 m = m_2[b] & m = m_2[b] \\
 a' = (b > \perp) & a' = (b > \perp)
 \end{array}$$

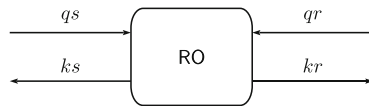
Now, when $b = \perp$, we have $\text{RealR}(m'_2, \perp) = \text{IdealR}(m'_2, \perp) = (\perp, \perp, \perp)$. So, no adversary can distinguish the two systems by not setting b . On the other hand, when $b \neq \perp$, RealR and IdealR are identical to the real and ideal systems from the previous section, augmented with the auxiliary output $a' = (b > \perp) = \top$. As we already observed in Sect. 3, these two distributions are computationally indistinguishable, proving that the length extension protocol is secure against corrupted receivers.

5 The OT Protocol of Chou and Orlandi

In this section we consider the OT protocol proposed by Chou and Orlandi in [10]. In the original paper, this is described as a protocol to execute l instances of 1-out-of- m OT, in parallel, i.e., the sender provides an l -dimensional vector of m -tuples of messages, and the receiver (non-adaptively) selects one message from each tuple. For simplicity, we consider the most basic case where $l = 1$ and $m = 2$, i.e., a single OT execution of a basic OT protocol as defined in the previous sections. This is without loss of generality because our results are ultimately negative. So, fixing $l = 1$ and $m = 2$ only makes our results stronger. Our goal is to show that this protocol is not provably secure in the equational framework according to a fully asynchronous simulation-based security definition. In order to formally analyze security, we begin by giving a mathematical description of the protocol and model of [10] using the equational framework.

The Random Oracle model. The protocol of [10] is designed and analyzed in the random oracle model [3]. So, both parties have access to an ideal functionality RO implementing a random function with appropriately chosen domain Q and range K . Queries from the sender and receiver are answered consistently, and, in general, RO can receive multiple (adaptively chosen) queries from both parties. Formally, the random oracle is modeled by the following functionality, where $f^*(x_1, x_2, \dots) = (f(x_1), f(x_2), \dots)$ is the standard extension of f to sequences:

$$\begin{array}{l}
 \text{RO}_{Q,K}(qs, qr) = (ks, kr) \\
 qs, qr : Q^* \\
 ks, kr : K^* \\
 f \leftarrow [Q \rightarrow K] \\
 ks = f^*(qs) \\
 kr = f^*(qr)
 \end{array}$$



The random oracle starts by picking a function $f: Q \rightarrow K$ uniformly at random, and then it uses f to answer any sequence of queries $qs, qr \in Q^*$ from each party. We give separate channels to access RO to the sender (qs) and receiver (qr) to model the fact that random oracle queries are implemented as local computations, and each party is not aware of if/when other players access the oracle. The Sender and Receiver programs from the protocol of [10] only make a small number of queries (two and one respectively.) Moreover, the two sender queries are chosen simultaneously, non-adaptively. So, for simplicity, we restrict $\text{RO}(q_2, q) = (k_2, k)$ to an oracle that receives just a pair of queries $q_2 = \langle q_0, q_1 \rangle \in Q_{\perp}^{\times 2}$ from the sender and one query $q \in Q_{\perp}$ from the receiver. We remark that in order to prove security, one should consider an arbitrary (still polynomial) number of (sequential, adaptively chosen) queries to model the adversary/environment ability to compute the RO function locally an arbitrary number of times.⁷ However, since our results are negative, fixing the number of queries only makes our result stronger: we show that the protocol is not provably secure even against the restricted class of adversaries that make only this very limited number of random oracle queries.

It has been observed, for example in [8], that a protocol analyzed stand-alone in the traditional random oracle model might lose its security when composed with other instances of protocols in the same random oracle model: either each instance uses an independent random oracle such that the real composed system cannot assume a single hash function, or the composed system suffers from transferability attack. A modified notion called *global random oracle* was proposed in [8] to allow a composed system achieving UC security when all protocols can access a single global random oracle. With respect to this issue, the OT protocol of [10] cannot be claimed UC secure and it should be re-defined in the global random oracle model or an equivalent notion. However, such issue is independent of the negative result we are going to present. Since our motivation is to illustrate the use of equational framework, for simplicity, we still consider the traditional random oracle model as used in [10].

The protocol. In order to facilitate a comparison with the original paper, we use as far as possible the same notation as [10]. Let $G = \langle B \rangle$ be a group generated by an element B of prime order p . Following [10], we use additive group notation, so that the group elements are written as xB for $x = 0, \dots, p - 1$.⁸ In [10] it is assumed that group elements have unique, canonical representations (which allows for equality testing), and group membership can be efficiently checked. Here, for simplicity, we assume that all messages representing group elements are syntactically valid, i.e., whenever a program expects a group element from G as

⁷ This can be modeled by letting qs and qr range over the set of sequences of queries Q^* , partially ordered according to the prefix ordering relation.

⁸ Chou and Orlandi use additive notation to match their efficient implementation based on elliptical curve groups. Here we are not concerned with any specific implementation, but retain the additive notation to match [10] and facilitate the comparison with the original protocol description.

input, it will always receive the valid representation of a such a group element (or \perp if the no message has been sent), even when this value is adversarially chosen. This is easily enforced by testing for group membership, and mapping invalid strings to some standard element, e.g., the group generator B .

The protocol uses a random oracle $\text{RO}(q_2, q) = (k_2, k)$ for functions with domain $Q = G^2 \times G$ and range $K = \{0, 1\}^n$, which receives two (parallel) queries $q_2 = \langle q_0, q_1 \rangle \in Q_{\perp}^{\times 2}$ from the sender and one query $q \in Q_{\perp}$ from the receiver.

The protocol also uses a symmetric encryption scheme (E, D) , with the same message space \mathbb{M}_n as the OT functionality, and key and ciphertext space $\mathbb{K}_n = \{0, 1\}_{\perp}^n$ equal to the range of the random oracle. In addition, the scheme is assumed to satisfy the following properties:

1. Non-committing: There exist PPT $\mathcal{S}_1, \mathcal{S}_2$ such that, for all $m \in \mathbb{M}_n$, the following distributions are identical:⁹

$$\begin{aligned} & \{(e, k) : k \leftarrow K, e \leftarrow E(k, m)\} \\ & \{(e, k) : e \leftarrow \mathcal{S}_1, k \leftarrow \mathcal{S}_2(e, m)\} \end{aligned}$$

2. Robustness: Let S be a set of keys chosen independently and uniformly at random from \mathbb{K}_n . For any PPT algorithms \mathcal{A} , if $e \leftarrow \mathcal{A}(S)$, then the set $V_{S,e} = \{k \in S \mid D(k, e) \neq \perp\}$ of keys under which e can be successfully decrypted has size at most 1 with overwhelming probability (over the choice of S and the randomness of \mathcal{A} .)

A simple encryption scheme satisfying these property is given by $E(m, k) = (m, 0^n) \oplus k$, i.e., padding the message with a string of zeros for redundancy, and masking the result with a one-time pad.

The protocol of [10] can be described by the equations in Fig. 5, and its execution is depicted in Fig. 6. We briefly explain the normal protocol execution: Sender first samples a random group element X and sends it to Receiver; once it receives Y from Receiver, it submits a pair of queries q_2 to RO; and once it receives random keys k_2 from RO, it encrypts messages m_2 under the keys k_2 , and it sends the ciphertext pair c_2 to Receiver. On the other hand, Receiver first samples a random group element yB , and upon receiving X from Sender it computes $Y = bX + yB$ and sends it to Sender; it then submits a query q to RO, and once the random key k and the ciphertexts c_2 are all received, it decrypts $c_2[b]$ using k to get the desired message m .

In the following subsections, we show that this protocol is insecure, both according to the classic OT definition given in Fig. 1, and according to our revised OT' definition of Fig. 3 that includes the signal $a = (b > \perp)$ to the sender. Specifically, first, in Subsects. 5.1 and 5.2 we show that if the definition from Fig. 1 is used, then the protocol is insecure against corrupted senders and corrupted receivers. The sender insecurity is for reasons very similar to

⁹ In fact, computational indistinguishability is enough, but it is easy to achieve perfect security.

$\text{Sender}(m_2, k_2, Y) = (q_2, X, c_2)$ $x \leftarrow \mathbb{Z}_p^*$ $X = xB$ $q_2[0] = ((X, Y), xY)$ $q_2[1] = ((X, Y), xY - xX)$ $c_2[0] \leftarrow \mathbf{E}(k_2[0], m_2[0])$ $c_2[1] \leftarrow \mathbf{E}(k_2[1], m_2[1])$	$\text{Receiver}(k, X, c_2, b) = (q, Y, m)$ $y \leftarrow \mathbb{Z}_p^*$ $Y = bX + yB$ $q = ((X, Y), yX)$ $m = \mathbf{D}(k, c_2[b])$
--	--

Fig. 5. The OT protocol of Chau and Orlandi.

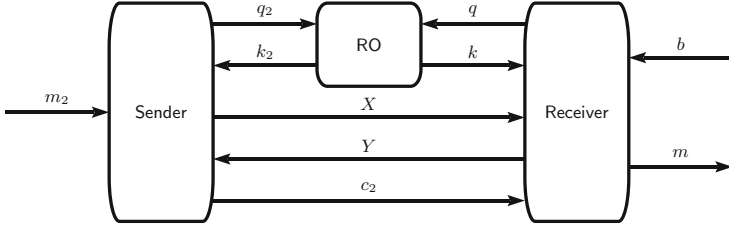
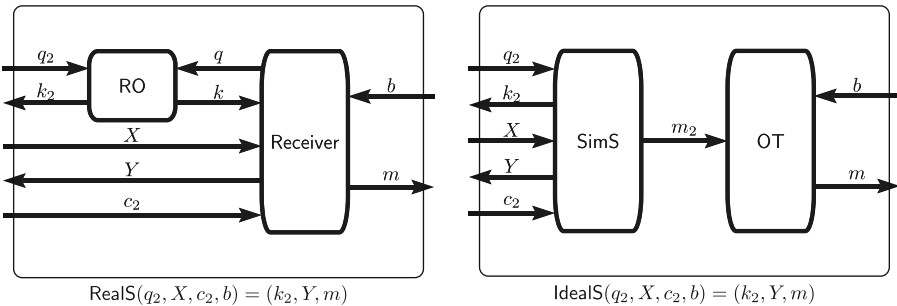


Fig. 6. A normal execution of the OT protocol of Chou and Orlandi.

those leading to the failure simulation in Sect. 3. Unlike the case of OT length extension, when considering the revised OT' definition and modifying the sender program accordingly, we show in Subsect. 5.3 that the modified protocol is still insecure against corrupted senders and corrupted receivers.

5.1 Corrupted Sender

We begin our analysis of the OT protocol with respect to the standard OT functionality, and we first consider the case when the sender is corrupted. The corresponding real and ideal systems are shown in the following diagrams:



For the protocol to be secure, the two systems should be computationally indistinguishable (for some simulator program SimS.) Just like the case of OT length extension, there exists an environment that can distinguish the two systems. We now describe an environment Env that works in two stages Env₀ and Env₁, and show that for any SimS, at least one of Env₀ and Env₁ distinguishes the real and ideal systems with nonnegligible advantage. We recall that

a distinguishing environment connects to all input and output channels of the system, and produces one external output $t \in \{\perp, \top\}$. The distinguishing advantage of Env_i is given by

$$\text{Adv}[\text{Env}_i] = |\Pr\{\text{Env}_i[\text{RealS}] = \top\} - \Pr\{\text{Env}_i[\text{IdealS}] = \top\}|.$$

The two stages of the distinguisher work as follows:

- $\text{Env}_0(k_2, Y, m) = (q_2, X, c_2, b, t)$ sets $q_2 = \perp$, $X = B$, $c_2 = \perp$ and $b = \perp$, and outputs $t = (Y > \perp)$.
- $\text{Env}_1(k_2, Y, m) = (q_2, X, c_2, b, t)$ sets $q_2 = \perp$, $X = B$, $c_2 = \perp$ and $b = 0$, and outputs $t = (Y > \perp)$.

Notice that the only difference between these two stages is in the value of b . Using the equations for the Receiver, we see that in the real system $Y > \perp$ if and only if $b > \perp$. In particular, we have $\Pr\{\text{Env}_0[\text{RealS}] = \top\} = 0$ and $\Pr\{\text{Env}_1[\text{RealS}] = \top\} = 1$. On the other hand, we have

$$\Pr\{\text{Env}_0[\text{IdealS}] = \top\} = \Pr\{\text{Env}_1[\text{IdealS}] = \top\} \tag{4}$$

because when interacting with IdealS , the output value t is independent of b . So, if we let p be the probability in (4), the two stages of Env have advantage $\text{Adv}[\text{Env}_0] = p$ and $\text{Adv}[\text{Env}_1] = 1 - p$. It follows that either Env_0 or Env_1 has distinguishing advantage at least $1/2$.

Intuitively, this environment can distinguish the real and the ideal systems because a corrupted sender (interacting with the real system RealS), learns when the receiver sets $b > \perp$ by observing the incoming message $Y > \perp$, but in the ideal system this timing information is not passed to the simulator.

5.2 Corrupted Receiver

We have seen that when using the standard OT definition, the protocol is not secure against corrupted senders. Now we turn to analyzing the protocol against corrupted receivers with respect to the standard OT definition. The real and ideal system in this case are shown in Fig. 7.

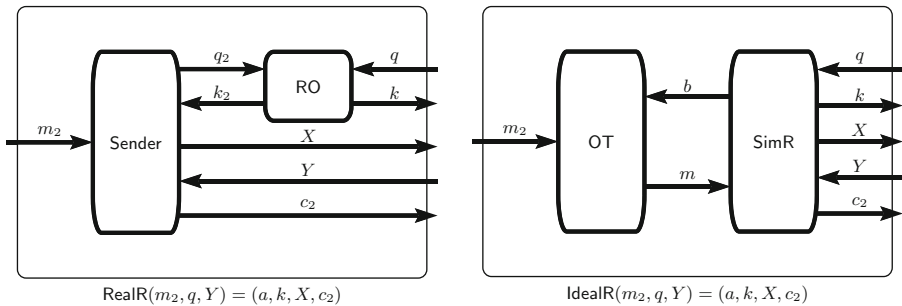


Fig. 7. The real and ideal systems when receiver is corrupted.

Security requires that the real and the ideal systems are indistinguishable for some simulator program SimR . Unfortunately, as we are about to show, no such simulator exists.

Proposition 1. *For the OT protocol in Fig. 5, when the receiver is corrupted, for any receiver simulator SimR , there is an environment that distinguishes the two systems with nonnegligible probability.*

Proof. We build an environment that works in three stages, denoted by Env_i for $i \in \{0, 1, 2\}$:

$$\begin{aligned} \text{Env}_0(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\ & d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 = \perp, Y = dX + yB, q = \perp, t = (c_2 = \perp) \\ \text{Env}_1(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\ & d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 \leftarrow \mathbb{M}_n^{\times 2}, Y = dX + yB, q = \perp, t = (c_2 > \perp) \\ \text{Env}_2(k, X, c_2) &= (m_2, q, Y, t) \text{ where} \\ & d \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_p^*, m_2 \leftarrow \mathbb{M}_n^{\times 2}, Y = dX + yB, q = ((X, Y), yX), \\ & t = (\mathbb{D}(k, c_2[d]) = m_2[d]) \end{aligned}$$

Assume there exists a receiver simulator SimR . With the real system, Env_i outputs $t = \top$ with probability 1 for all $i \in \{0, 1, 2\}$. So $\Pr\{\text{Env}_i[(\text{OT}|\text{SimR})] = \top\}$ must be overwhelming for all $i \in \{0, 1, 2\}$.

Notice that in the ideal system both b and m are internal channels such that $m = m_2[b]$, and we can simplify the output of the ideal system as $(k, X, c_2) \leftarrow \text{SimR}(m_2[b], q, Y)$. For $i = 0, 1, 2$, let u_i denote the (random variable of) the input to SimR when working with Env_i , and let $\text{SimR}(u_i)^b$ denote the (random variable of) the value of b given input u_i . The external input channels to SimR are q and Y , and their values are \perp in both Env_0 and Env_1 . If SimR sets $b = \perp$ when $q = \perp$ and $Y = \perp$, then it cannot tell the difference between Env_0 and Env_1 , and thus at least one of Env_0 and Env_1 has a nonnegligible distinguishing advantage. So $\Pr\{\text{SimR}(u_0)^b > \perp\}$ must be overwhelming. Since SimR is a monotone function, $\Pr\{\text{SimR}(u_i)^b > \perp\}$ is also overwhelming for $i \in \{1, 2\}$. In particular, let $\epsilon = \frac{1}{2} \Pr\{\text{SimR}(u_1)^b = \perp\}$, then ϵ is negligible.

Now consider Env_1 , which sets $q = \perp$ and samples Y from the distribution $\{dX + yB \mid y \leftarrow \mathbb{Z}_p^*\} \equiv \{yB \mid y \leftarrow \mathbb{Z}_p^*\}$. So q and Y are independent of d , and thus $\Pr\{\text{SimR}(u_1)^b = d\} = \Pr\{\text{SimR}(u_1)^b = 1 - d\} = \frac{1}{2} - \epsilon$.

Finally, when working with Env_2 we have

$$\begin{aligned} & \Pr\{\text{Env}_2[(\text{OT}|\text{SimR})] = \top\} = \Pr\{\mathbb{D}(k, c_2[d]) = m_2[d]\} \\ &= \Pr\{\mathbb{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = d\} \Pr\{\text{SimR}(u_2)^b = d\} \\ & \quad + \Pr\{\mathbb{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = 1 - d\} \Pr\{\text{SimR}(u_2)^b = 1 - d\} \\ & \quad + \Pr\{\mathbb{D}(k, c_2[d]) = m_2[d] \mid \text{SimR}(u_2)^b = \perp\} \Pr\{\text{SimR}(u_2)^b = \perp\} \end{aligned}$$

Since SimR is monotone, $\frac{1}{2} - \epsilon = \Pr\{\text{SimR}(u_1)^b = 1 - d\} \leq \Pr\{\text{SimR}(u_2)^b = 1 - d\}$, and thus $\Pr\{\text{SimR}(u_2)^b = d\} \leq \frac{1}{2} + \epsilon$. On the other hand, when $\text{SimR}(u_2)^b = 1 - d$, it holds that $\text{SimR}(u_i)^b \in \{1 - d\}_\perp$ for $i \in \{0, 1\}$ and thus SimR has no access to

$m_2[d]$, and since $m_2[d]$ is independently sampled from \mathbb{M}_n , **SimR** cannot guess it correctly with probability more than $\frac{1}{2^n}$. So we can bound the probability

$$\Pr\{\text{Env}_2[(\text{OT}|\text{SimR})] = \top\} \leq \frac{1}{2} + \epsilon + \frac{1}{2^n} + 2\epsilon,$$

which is close to $\frac{1}{2}$. Therefore the environment can distinguish the real and the ideal systems with nonnegligible probability. \square

5.3 Revised OT Definition

The timing issue with a corrupted sender is similar to the one for OT length extension that is fixed by adding an acknowledgment signal. So it is natural to ask if the insecurity problems can be resolved by modifying the protocol according to the revised functionality OT' . Clearly, changing the definition requires also modifying the sender program to output a signal a in order to match OT' . Since the sender receives only one message (Y) from the receiver, there is only one sensible way to modify the protocol to produce this additional output: setting $a = (Y > \perp)$. Formally, we consider the following modified sender program:

$$\begin{aligned} \text{Sender}'(m_2, k_2, Y) &= (a, q_2, X, c_2) \\ (q_2, X, c_2) &\leftarrow \text{Sender}(m_2, k_2, Y) \\ a &= (Y > \perp) \end{aligned}$$

We leave it to the reader to verify that a real protocol execution ($\text{Sender}' \mid \text{RO} \mid \text{Receiver}$): $(m_2, b) \mapsto (a, m)$ is equivalent to the ideal functionality OT' : $(m_2, b) \mapsto (a, m)$.

For security, we start with the case when the receiver is corrupted. The real and ideal systems are depicted in Fig. 8. Notice that the additional bit a is not provided to the simulator but is instead given to the environment. So any receiver simulator **SimR** that connects to OT' to form the ideal system in the revised OT definition has the same interface as a receiver simulator in the standard OT definition. Thus we obtain the same result as in Proposition 1 that the modified protocol is insecure against corrupted receivers.

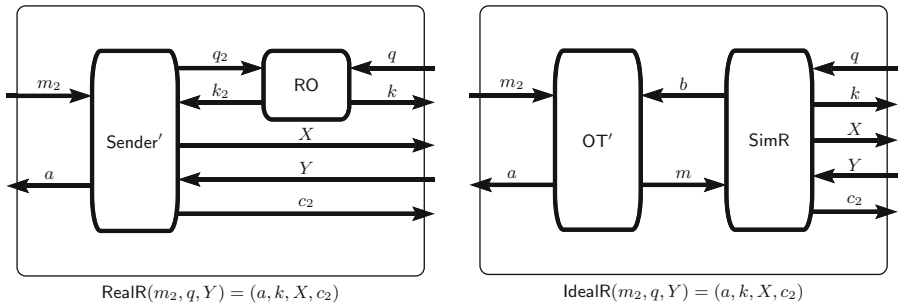


Fig. 8. The real and ideal systems when receiver is corrupted, under revised OT definition.

When the sender is corrupted, the sender simulator is now provided with an additional bit $a = (b > \perp)$, as shown in Fig. 9. This small modification is the key to prove security for the OT length extension protocol, so one might speculate, as we did in the previous version of this paper, that security could also hold for the current protocol in the case of sender corruption. On the contrary, this modification is not enough. As we are exploring the useability of the equational framework, we show in the following why the natural simulation strategy that takes advantage of the signal a fails at proving security.

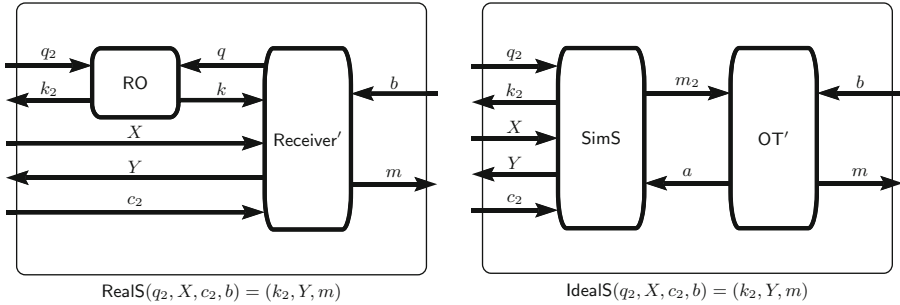


Fig. 9. The real and ideal systems when sender is corrupted, under revised OT definition.

The speculated simulator is shown below. As we are presenting negative results, we limit the power of a corrupted sender such that it can send at most one pair of RO queries q_2 and it obtains at most one pair of keys k_2 .

$$\begin{aligned}
 \text{SimS}(q_2, X, a, c_2) &= (k_2, Y, m_2) \\
 f &\leftarrow [(G^2 \times G) \rightarrow K] \\
 k_2 &= f^*(q_2) \\
 y &\leftarrow \mathbb{Z}_p^* \\
 Y &= X!a!yB \\
 m_2[0] &= \text{if } (\exists i. q_2[i] = ((X, Y), \dots)) \text{ then } D(k_2[i], c_2[0]) \\
 m_2[1] &= \text{if } (\exists i. q_2[i] = ((X, Y), \dots)) \text{ then } D(k_2[i], c_2[1])
 \end{aligned}$$

Let us derive an equation for m . In the real system RealS, the message m satisfies the equation

$$m = D(f((X, bX + yB), yX), c_2[b]), \tag{5}$$

where y is sampled uniformly at random from \mathbb{Z}_p^* by the honest receiver. In the ideal system IdealS = (SimS|OT'), notice that $a = (b > \perp)$, and so

$$m = D(f((X, X!b!yB), W), c_2[b]), \tag{6}$$

where y is sampled uniformly at random from \mathbb{Z}_p^* by the simulator and W is some element of G chosen by the environment. In both Eqs. (5) and (6), $c_2[b]$

is an input to the system given by the environment. By a careful examination, we can see that the value of m as computed in these two equations could be different if the environment sets W to be distinct from yX . We follow this idea to construct the following environment:

$$\begin{aligned} \text{Env}(k_2, Y, m) &= (q_2, X, c_2, b, t) \text{ where} \\ x &\leftarrow \mathbb{Z}_p^*, X = xB, w \leftarrow \mathbb{Z}_p^*, W = wB, b \leftarrow \{0, 1\}, \\ \text{For } i \in \{0, 1\}: \\ q_2[i] &= ((X, Y), W), c_2[i] \leftarrow \mathbf{E}(k_2[i], 0), \\ t &= (m > \perp) \end{aligned}$$

In the real system, Env outputs $t = \top$ only in two cases: either the key $k = f((X, Y), yX)$ obtained by the receiver is same as the key $k_2[b] = f((X, Y), W)$ used by Env to encrypt $m_2[b]$ in the ciphertext $c_2[b]$, where f is a random function sampled by RO , or the decryption succeeds when $k \neq k_2[b]$. For a sufficiently large key space \mathbb{K}_n , since $yX = yxB$ and $W = wB$ are independently sampled and uniformly distributed, the probability ϵ that $k = k_2[b]$ is negligible. Since (\mathbf{E}, \mathbf{D}) is a robust encryption scheme, when $k \neq k_2[b]$ the decryption can succeed with only a negligible probability δ . So Env outputs $t = \top$ with a negligible probability $\epsilon + (1 - \epsilon)\delta$. But in the ideal system, the decryption always succeeds and thus we get $m = 0 > \perp$, which implies that Env outputs $t = \top$ with probability 1. Therefore Env has a nonnegligible distinguishing advantage.

We remark that, if the above simulator SimS has access to a DDH oracle \mathbf{O} that answers on input (X, Y, W) whether $W = yxB$ for $X = xB$ and $Y = yB$, then we can modify the equations for m_2 in SimS to prove sender security with respect to the revised OT definition:

$$\begin{aligned} m_2[0] &= \text{if } (\exists i. q_2[i] = ((X, Y), W) \text{ and } \mathbf{O}(X, Y, W) = \top) \text{ then } \mathbf{D}(k_2[i], c_2[0]) \\ m_2[1] &= \text{if } (\exists i. q_2[i] = ((X, Y), W) \text{ and } \mathbf{O}(X, Y, W) = \top) \text{ then } \mathbf{D}(k_2[i], c_2[1]) \end{aligned}$$

That is, if a RO query contains a triple of group elements satisfying the DDH condition, then SimS uses the corresponding key to decrypt both $c_2[0]$ and $c_2[1]$ and assigns the resulting plaintext to $m_2[0]$ and $m_2[1]$, respectively. As already noted by Genç et al. [13], sender security holds with certain gap-DH groups in which the CDH problem is hard but the DDH problem is easy to solve.

6 Conclusion

We considered two OT protocols within the equational framework in this paper: The OT length extension protocol and the “simplest” OT protocol by Chou and Orlandi [10]. Both examples demonstrated the simplicity and expressive power of the equational framework in analyzing MPC protocols. We found that the traditional formulation of the OT problem does not fit into a fully asynchronous simulation-based security model, and we revised it accordingly to fix it for the OT length extension protocol. Still, the revised formulation does not allow to salvage the OT protocol of Chou and Orlandi. Overall, the equational framework proved to be a convenient formalism to carry out rigorous, yet concise, security analysis of cryptographically interesting protocols.


References

1. Backes, M., Pfizmann, B., Waidner, M.: The reactive simulatability (RSIM) framework for asynchronous systems. *Inf. Comput.* **205**(12), 1685–1720 (2007)
2. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), 17–19 October 2004, Rome, Italy, Proceedings, pp. 186–195 (2004)
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, pp. 62–73. ACM, New York (1993)
4. 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
5. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 2001 42nd IEEE Symposium on Foundations of Computer Science, Proceedings, pp. 136–145, October 2001
6. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 3–22. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_1
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_4
8. Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 597–608. ACM, New York (2014)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, STOC 2002, pp. 494–503. ACM, New York (2002)
10. Chou, T., Orlandi, C.: The simplest protocol for oblivious transfer. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 40–58. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22174-8_3
11. Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 110–123. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_9
12. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. *Commun. ACM* **28**(6), 637–647 (1985)
13. Genç, Z.A., Iovino, V., Rial, A.: “The simplest protocol for oblivious transfer” revisited. IACR Cryptology ePrint Archive 2017, 370 (2017). <http://eprint.iacr.org/2017/370>
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 218–229. ACM, New York (1987)
15. Gunter, C.A., Scott, D.S.: Semantic domains. In: Handbook of theoretical computer science, vol. b, pp. 633–674. MIT Press, Cambridge (1990)

16. Hofheinz, D., Shoup, V.: GNUC: a new universal composability framework. *J. Cryptol.* **28**(3), 423–508 (2015)
17. Hofheinz, D., Unruh, D., Müller-Quade, J.: Polynomial runtime and composability. *J. Cryptol.* **26**(3), 375–441 (2013)
18. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
19. Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: Sahai, A. (ed.) *TCC 2013*. LNCS, vol. 7785, pp. 477–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_27
20. Kilian, J.: Founding cryptography on oblivious transfer. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988*, pp. 20–31. ACM, New York (1988)
21. Küsters, R.: Simulation-based security with inexhaustible interactive turing machines. In: *Computer Security Foundations Workshop, CSFW-19 2006*, pp. 309–320. IEEE Computer Society (2006)
22. Küsters, R., Tuengerthal, M.: The IITM model: a simple and expressive model for universal composability. *IACR Cryptology ePrint Archive 2013*, 25 (2013). <http://eprint.iacr.org/2013/025>
23. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptol.* **25**(4), 680–722 (2011)
24. Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) *TOSCA 2011*. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27375-9_3
25. Maurer, U., Renner, R.: Abstract cryptography. In: *Innovations in Computer Science - ICS 2010, Proceedings*. Tsinghua University, Beijing, China, 7–9 January 2011, pp. 1–21 (2011)
26. Micciancio, D., Tessaro, S.: An equational approach to secure multi-party computation. In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS 2013*, pp. 355–372. ACM, New York (2013)
27. Rabin, M.O.: How to exchange secrets with oblivious transfer, Technical report, TR-81 edn. Harvard University, Aiken Computation Lab (1981)
28. Scott, D.S.: Domains for denotational semantics. In: Nielsen, M., Schmidt, E.M. (eds.) *ICALP 1982*. LNCS, vol. 140, pp. 577–610. Springer, Heidelberg (1982). <https://doi.org/10.1007/BFb0012801>
29. Stoltenberg-Hansen, V., Lindström, I., Griffor, E.R.: *Mathematical Theory of Domains*. Cambridge University Press, New York (1994)
30. Wikström, D.: Simplified universal composability framework. In: Kushilevitz, E., Malkin, T. (eds.) *TCC 2016*. LNCS, vol. 9562, pp. 566–595. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_24
31. Winskel, G.: *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge (1993)
32. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: *Foundations of Computer Science, Proceedings of FOCS 1986*, pp. 162–167. IEEE Computer Society (1986)



Extending Oblivious Transfer with Low Communication via Key-Homomorphic PRFs

Peter Scholl^(✉) 

Aarhus University, Aarhus, Denmark
`peter.scholl@cs.au.dk`

Abstract. We present a new approach to extending oblivious transfer with communication complexity that is logarithmic in the security parameter. Our method only makes black-box use of the underlying cryptographic primitives, and can achieve security against an active adversary with almost no overhead on top of passive security. This results in the first oblivious transfer protocol with sublinear communication and active security, which does not require any non-black-box use of cryptographic primitives.

Our main technique is a novel twist on the classic OT extension of Ishai et al. (Crypto 2003), using an additively key-homomorphic PRF to reduce interaction. We first use this to construct a protocol for a large batch of 1-out-of- n OTs on random inputs, with amortized $o(1)$ communication. Converting these to 1-out-of-2 OTs on chosen strings requires logarithmic communication. The key-homomorphic PRF used in the protocol can be instantiated under the learning with errors assumption with exponential modulus-to-noise ratio.

1 Introduction

In an oblivious transfer protocol, a receiver wishes to learn a subset of some messages held by a sender, whilst hiding exactly which messages are received. A common type of oblivious transfer is 1-out-of-2 OT, where the sender holds messages x_0, x_1 , while the receiver holds a bit b and wishes to learn x_b . The protocol should guarantee that the receiver learns no information on x_{1-b} , whilst the sender learns nothing about b . 1-out-of-2 OT is a key tool in building secure two-party and multi-party computation protocols, and most efficient protocols need to use a very large number of oblivious transfers that scales with the input size [Yao86], or the size of the circuit description of the function being computed [GMW87].

All known protocols for oblivious transfer are much more expensive than standard symmetric-key primitives, as they rely on public-key cryptography. This property seems to be inherent, since it is known that constructing OT from symmetric cryptographic primitives in a black-box manner is impossible [IR89].

An essential technique for reducing the cost of oblivious transfer is *OT extension*, which reduces the cost of carrying out many OTs with amortization. OT extension protocols proceed in two stages: in a setup phase, a small number of ‘seed’ OTs are created using standard public-key techniques; secondly, these are extended to create many more, independent OTs, with a lower cost than the seed OT protocols. Typically the second phase is based only on cheap, symmetric cryptography, so using OT extension allows many OTs to be created with only $O(k)$ public-key operations for security parameter k , greatly reducing the computational costs.

OT Extension: A Brief History. Classically, OT extension refers to evaluating OT using mainly symmetric key cryptography. In this paper we broaden the term to cover any protocol that generates $m = \text{poly}(k)$ OTs in a way which is more efficient than executing m instances of an OT protocol. Reducing communication for the case of (1-out-of-2) *bit-OT*, where the sender’s messages are bits, is of particular importance. This is exactly what is needed in the GMW protocol for secure multi-party computation [GMW87, Gol04], and a bit-OT protocol with $O(1)$ communication complexity implies secure computation with *constant overhead* using GMW, and even with active security [IPS08].

Beaver [Bea96] first showed how to convert $O(k)$ seed OTs into any polynomial number of OTs, using only one-way functions, for security parameter k . In this technique, the parties use a secure two-party computation protocol to evaluate the circuit that takes as input a random seed from each party, then applies a PRG and computes the OT functionality on the expanded, random inputs. With Yao’s protocol [Yao86] this only needs $O(k)$ OTs, since the inputs are of size $O(k)$.

The ‘IKNP’ protocol, by Ishai et al. [IKNP03], lies at the core of all recent, practical OT extension protocols. IKNP was the first protocol to efficiently extend OT in a *black-box* manner, using only a hash function which satisfies a correlation robustness assumption (or a random oracle). The communication complexity of this protocol is $O(k + \ell)$ bits per extended OT with passive security, where ℓ is the bit length of the sender’s strings. Harnik et al. [HIKN08] later showed how to obtain active security with the same asymptotic efficiency. The TinyOT protocol [NNOB12] introduced the first practical, actively secure OT extension, and more recently the overhead of active security has been reduced to almost nothing for both the 1-out-of-2 [ALSZ15, KOS15] and 1-out-of- n cases [OOS17, PSS17].

These protocols are essentially optimal for transferring messages of length $\ell = \Omega(k)$, but when ℓ is short (as in bit-OT where $\ell = 1$) there is still an overhead in $O(k)$. Kolesnikov and Kumaresan [KK13] presented a variant of the IKNP protocol based on 1-out-of- n OT, which can be used to perform 1-out-of-2 bit-OT with $O(k/\log k)$ communication. It is not known how to make this bit-OT protocol actively secure, because it relies on a passively secure reduction from 1-out-of- n to 1-out-of-2 OT [NP99].

Unfortunately, all known methods for achieving *constant-communication* bit-OT use very complex techniques, and often require non-black-box use of

cryptographic primitives. Ishai et al. [IKOS08] combined Beaver’s non-black-box technique [Bea96] for OT extension with a polynomial-stretch PRG in NC0 and randomized encodings, to obtain a passively secure protocol with amortized $O(1)$ computational overhead (implying $O(1)$ communication). As well as needing a strong assumption on the PRG, a major drawback is the use of non-black-box techniques, which lead to a very high constant. The same authors later gave an alternative, black-box approach with constant communication [IKOS09]. However, this still requires heavy machinery such as algebraic geometry codes, randomized encodings and low-communication PIR. Additionally, achieving active security would require generic use of zero-knowledge proofs [GMW86, IKOS07], again with non-black-box use of the underlying primitives. Recently, Boyle et al. [BGI17] showed how to obtain an amortized communication cost of just 4 bits per bit-OT using homomorphic secret-sharing, which can be realised from either DDH [BGI16] or LWE [DHRW16]. As with the previous works, however, this construction makes non-black-box use of PRGs and would be extremely inefficient in practice.

Finally, we remark that using indistinguishability obfuscation and fully homomorphic encryption, it is possible to produce $\text{poly}(k)$ OTs on random inputs with a communication complexity that is independent of the number of OTs, with a general method for reusable correlated randomness in secure computation [HW15, HIJ+16].

1.1 Contributions of This Work

We present a new approach to extending oblivious transfer with low communication. Our protocol, in the random oracle model, makes black-box use of the underlying cryptographic primitives and can achieve security against an active adversary with almost no overhead on top of passive security. This results in the first bit-OT protocol with sublinear communication and active security, making only black-box use of cryptographic primitives. Table 1 compares the characteristics of our protocol with some of the other OT extension protocols discussed earlier.

Our main technique is a novel twist on the classic IKNP OT extension, using an additively key-homomorphic PRF to reduce interaction. The main challenge here is to handle the homomorphism error present in known key-homomorphic PRF constructions, without compromising on correctness or security. We first present a protocol for a large batch of 1-out-of- p_i OTs on random inputs, for multiple, distinct primes p_i . The communication complexity of this protocol is *sublinear in the total number of random OTs*, with an amortized cost of $o(1)$ bits per OT. It was not known previously how to achieve this without using obfuscation,¹ and this primitive may be useful in wider applications. If we want

¹ Even with obfuscation, secure computation with complexity sublinear in the output size and active security is known to be impossible [HW15]. The obfuscation-based protocol of [HIJ+16] circumvents this using a CRS, whilst we use the random oracle model.

Table 1. Various protocols for extending 1-out-of-2 bit-OT (unless otherwise specified) with different assumptions. All passively secure protocols can be transformed to be actively secure using non-black-box zero-knowledge techniques. CRH is a correlation-robust hash function, RO is a random oracle; $\binom{n}{1}$ -ROT is 1-out-of- n OT on random inputs.

Protocol	Communication per OT (bits)	Security	Based on	Black-box
[Bea96]	$\text{poly}(k)$	passive	OWF	✗
[IKNP03]	$O(k)$	passive	CRH/RO	✓
[KK13]	$O(k/\log k)$	passive	CRH/RO	✓
[ALSZ15, KOS15]	$O(k)$	active	CRH/RO	✓
[IKOS08]	$O(1)$	passive	poly-stretch local PRG	✗
[IKOS09]	$O(1)$	passive	Φ -hiding	✓
[BGI17]	$4 + o(1)$	passive	DDH	✗
[HLJ+16] ($\binom{n}{1}$ -ROT)	$o(1)$	active	iO + FHE	✗
This work ($\binom{p_i}{1}$ -ROT ^a)	$o(1)$	active	LWE + RO	✓
This work ($\binom{2}{1}$ -OT)	$O(\log k)$	active	LWE + RO	✓

^a p_i are small distinct primes

to obtain 1-out-of-2 OT on chosen strings, each 1-out-of- p_i random OT can be converted to a 1-out-of-2 OT with $O(\log p_i)$ bits of communication using standard techniques, giving logarithmic communication overhead.

The additively key-homomorphic PRF needed in our protocol can be instantiated based on the learning with errors assumption with an exponential modulus-to-noise ratio. This assumption has previously been used to construct attribute-based encryption [GVW13] and fully key-homomorphic encryption [BGG+14], and is believed to be hard if the LWE dimension is chosen large enough to thwart known attacks. The downside of our approach is that this spectrum of LWE parameters results in fairly heavy computational costs for the parties, so it seems that the main uses of our protocol would be in low bandwidth environments where communication is much more expensive than computation.

As a contribution of independent interest, to implement the base OTs in our protocol we generalise the consistency check used for OT extension by Asharov et al. [ALSZ15], and adapt it for producing *correlated OTs* over any abelian group \mathbb{G} , instead of just XOR correlations over bit strings. These are 1-out-of-2 OTs where the sender’s messages are all guaranteed to be of the form $(x_i, x_i + \Delta)$, for some fixed correlation $\Delta \in \mathbb{G}$. We also identify a crucial security flaw in the original protocol of Asharov et al. [ALSZ15, ALSZ17a], which leaks information on the receiver’s inputs to a passively corrupted sender. After reporting this to the authors, their protocol has been modified to fix this [ALSZ17b], and we use the same fix for our protocol.

1.2 Overview of Techniques

IKNP OT Extension. We first recall the IKNP OT extension protocol [IKNP03] (with optimizations from [ALSZ13, KK13]), which uses k instances of oblivious transfer to construct $m = \text{poly}(k)$ oblivious transfers with only a cryptographic hash function and a pseudorandom generator. The parties begin by performing k 1-out-of-2 OTs on random k -bit strings with their roles reversed. The receiver acts as sender in the base OTs, with input pairs of strings (k_i^0, k_i^1) . The sender, acting as receiver, inputs a random choice bit s_i to the i -th OT and learns $k_i^{s_i}$, for $i = 1, \dots, k$. The receiver then sends over the values

$$u_i = G(k_i^0) \oplus G(k_i^1) \oplus x$$

where $G : \{0, 1\}^k \rightarrow \{0, 1\}^m$ is a pseudorandom generator (PRG) and $x = (x_1, \dots, x_m)$ are the receiver’s m choice bits. After this step, the parties can obtain k correlated OTs on pairs of m -bit strings of the form $(t_i, t_i \oplus x)$, where $t_i = G(k_i^0)$: the receiver knows both t_i and $t_i \oplus x$, while the sender can define

$$\begin{aligned} q_i &= G(k_i^{s_i}) \oplus s_i \cdot u_i \\ &= t_i \oplus s_i \cdot x \\ &= \begin{cases} t_i & \text{if } s_i = 0 \\ t_i \oplus x & \text{if } s_i = 1 \end{cases} \end{aligned}$$

This is a 1-out-of-2 OT on m -bit strings because the other message, $t_i \oplus \overline{s_i} \cdot x$, is computationally hidden to the sender due to the use of a PRG.

Both parties then place these values into matrices $Q, T \in \{0, 1\}^{k \times m}$ containing q_i and t_i (respectively) as rows, where the sender holds Q and the receiver holds T . If $q^j, t^j \in \{0, 1\}^k$ are the columns of Q and T , and $s = (s_1, \dots, s_k)$, then notice that we have

$$t^j = q^j \oplus (x_j \cdot s)$$

So, by transposing the matrix of OTs we obtain m sets of correlated OTs on k -bit strings, with x_j as the receiver’s choice bits. Finally, the two parties can convert these correlated OTs into OTs on random strings using a hash function H that satisfies a notion of correlation robustness (or, modeled as a random oracle): the sender computes the two strings $H(q^j)$ and $H(q^j \oplus s)$, whilst the receiver can only compute one of these with $H(t^j)$; the other string remains unknown to the receiver since it does not know s . This means the parties have converted k initial OTs into $m = \text{poly}(k)$ OTs on random strings, and these random OTs can be used to transfer the sender’s chosen messages by encrypting them with a one-time pad.

Apart from the initial base OTs, the only interaction in this process is sending the u_i values at the beginning, which costs $O(mk)$ bits of communication. This gives an overhead of $O(k)$ when the sender’s inputs are bit strings of constant size.

Using a Key-Homomorphic PRF. We observe that if the PRG, G , in the above protocol satisfies $G(x \oplus y) = G(x) \oplus G(y)$, and the *base OTs* are correlated so that $k_i^1 = k_i^0 \oplus r$ for some fixed string r , then the main step of interaction in IKNP can be removed. The homomorphic property of the PRG preserves the correlation, so the parties can obtain the OTs $(t_i, t_i \oplus x)$ (for random choice bits x) without any message from the receiver: the sender simply defines $q_i = G(k_i^{s_i})$ while the receiver defines $t_i = G(k_i^0)$ and $x = G(r)$. We then have

$$q_i = G(k_i^{s_i}) = G(k_i^0 \oplus (s_i \cdot r)) = G(k_i^0) \oplus (s_i \cdot G(r)) = t_i \oplus s_i \cdot x,$$

as previously.

Unfortunately, such XOR-homomorphic PRGs are not known to exist. Instead, we do know how to build *almost*-seed-homomorphic PRGs (and almost-key-homomorphic PRFs) $G : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_p$, which satisfy

$$G(x + y) = G(x) + G(y) + e \pmod{p},$$

where $q > p$ and $|e| \leq 1$ is an error term, based on the *learning with rounding* (LWR) or *learning with errors* (LWE) assumption [BLMR13, BP14]. We remark that it is possible to build an *error-free* key-homomorphic PRF in the random oracle model based on the decisional Diffie-Hellman assumption, with the simple construction $F(k, x) = H(x)^k$ [BLMR13]. However, here the output homomorphism is multiplicative instead of additive, which is more difficult to exploit in constructing OT extension.

Trying to apply these additively homomorphic PRGs (or PRFs) to the IKNP protocol brings about two main challenges. Firstly, since the homomorphism maps into \mathbb{Z}_p and not \mathbb{F}_2^k , we obtain matrices Q and T containing \mathbb{Z}_p elements instead of bits, which means there is no natural way of ‘transposing’ the OT matrix whilst preserving the correlated OT property. Secondly, the homomorphism error means that all of the OTs will be incorrect with high probability.

To handle the first problem, we choose p to be a *primorial modulus* which is the product of ℓ primes, and then decompose the correlated OTs via the Chinese Remainder Theorem.² This gives us an alternative to transposing the bit-matrix in IKNP; however, it means that instead of constructing 1-out-of-2 OT, we end up with 1-out-of- p_i random OTs, for each prime factor p_i in the modulus. The second problem of eliminating the error is more difficult. We observe that the receiver can always *compute* a homomorphism error e , such that the resulting error in the OT is given by $e' = e \cdot s_i$, where s_i is one of the sender’s choice bits in the base OTs. It seems tempting to let the receiver just send over e so that the sender can correct the error, but this may not be secure: each error leaks information about the unknown PRG key, and a large number of errors could leak information on the secret PRG outputs. To securely correct the error, the receiver instead samples some uniform noise u , which is used to mask e . To ensure that both parties still obtain the correct result, we must *obviously*

² Ball et al. used a primorial modulus for a different application of constructing arithmetic garbled circuits [BMR16].

transfer the masked error $u + s_i \cdot e'$ to the sender. Since s_i is a choice bit in the original base OTs, this can be done by without any additional OTs, by extending the base OTs once more with a (standard) PRG.

This last error-correction step introduces some interaction into the basic OT extension protocol, which is otherwise non-interactive. Importantly, the amount of data that needs sending only depends on the security parameter, and not the modulus. Since each distinct prime factor in the modulus produces one additional OT in the OT extension phase, choosing a sufficiently large modulus allows us to obtain an amortized communication cost of $o(1)$ bits per random OT. If we wish to construct 1-out-of-2 OTs, each random 1-out-of- p_i OTs can be converted to a single 1-out-of-2 OT with $\log p_i = O(\log k)$ bits of communication (see Appendix A).

Active security. To obtain active security, the above protocol needs to be modified in two ways. Firstly, we need to ensure that the correlation in the base OTs is created correctly, and secondly, we need to ensure that a malicious receiver does not cheat in the error-correction stage, which would cause incorrect outputs.

We first consider the error-correction step. A common technique for dealing with this is to compute random linear combinations of all the correlated OTs, then open the result [KOS15] and check correctness. However, this only achieves negligible cheating probability when the correlation is over a *large field*. In our case we use a ring with many zero divisors, and this method cannot be applied in general. Nevertheless, our situation is slightly different because the *size* of the adversarial deviations can be bounded by some value B that is much smaller than the modulus. This means that for some error $d < B$ introduced by a cheating receiver, and random challenge $r \leftarrow \mathbb{Z}_p$, the product $dr \bmod p$ is *statistically close* to uniform in \mathbb{Z}_p (for arbitrary p), which suffices to prove security when taking random linear combinations.

For the base OTs, the receiver can cheat in an arbitrary way when creating the correlations, which means the above check is not enough to prevent deviations here. It might be tempting to work around this problem by choosing a *prime* modulus q for the base OTs (before applying the PRG/PRF to convert mod p). The problem here is that we then wouldn't be able to *transpose* the base OTs, which is necessary for checking consistency via random linear combinations. Instead, we adopt a different approach used for OT extension in [ALSZ15], where the receiver sends hashes of every pair of base OTs, which are then checked for consistency by the sender. We show that this approach still works to prove that the OTs are correlated over an *arbitrary* abelian group, instead of just XOR correlations over bit strings. If the receiver cheats, they may guess a few bits of the sender's secret choice bits. This does not cause a problem for the OT extension phase, and we model this possibility in our setup functionality.

Instantiation based on DDH. It is possible to modify the above protocol to use a key-homomorphic PRF in the random oracle model based on the decisional

Diffie-Hellman assumption, instead of using LWE or LWR. This has the advantage of avoiding the problems with homomorphism error, since the DDH-based PRF $F(k, x) = H(x)^k$ is noiseless. However, the drawback is that this protocol produces random 1-out-of- p OTs, where p is the order of a group in which DDH is hard. Since DDH is not hard if p has small factors, these cannot be decomposed into smaller random OTs using the CRT, so this does not lead to any improvements to 1-out-of-2 OT extension. Nevertheless, random 1-out-of- p OT for exponentially large p (sometimes referred to as batch related-key oblivious PRF evaluation) can be used to construct private equality test and private set intersection protocols [PSZ18, KKRT16, OOS17], so this variation could be useful in these applications to reduce interaction at the cost of requiring exponentiations instead of only symmetric-key operations. More details on this protocol are given in the full version of this work.

2 Preliminaries

2.1 Universally Composable Security

We present ideal functionalities and security proofs in the universal composability framework [Can01], and assume some familiarity with this.

Informally speaking, for a protocol Π which implements a functionality \mathcal{F} in the \mathcal{G} -hybrid model, we let $\text{HYBRID}_{\Pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{G}}$ denote the view of an environment \mathcal{Z} in an execution of the real protocol with the adversary \mathcal{A} controlling the corrupted parties, in a hybrid model where all parties have access to the ideal functionality \mathcal{G} . We let $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$ denote the view of \mathcal{Z} in the ideal execution, where the simulator \mathcal{S} plays the role of the corrupted parties in Π and interacts with the functionality \mathcal{F} . When the context is clear, we sometimes abbreviate the two executions by HYBRID and IDEAL .

We say that the protocol Π securely realises the functionality \mathcal{F} in the \mathcal{G} -hybrid model, if for every adversary \mathcal{A} there exists a simulator \mathcal{S} , such that for every environment \mathcal{Z} ,

$$\text{HYBRID}_{\Pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{G}} \stackrel{c}{\approx} \text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$$

where $\stackrel{c}{\approx}$ is the standard notion of computational indistinguishability.

As well as the standard, computational security parameter k , we often use a statistical security parameter λ . This means that the advantage of any probabilistic $\text{poly}(k)$ -time environment in distinguishing the two executions is at most $\text{negl}(k) + O(2^{-\lambda})$.

2.2 Key-Homomorphic Pseudorandom Functions

We now recall the definitions of additively key-homomorphic pseudorandom functions [BLMR13, BP14], and discuss the distribution of the homomorphism error in LWE-based constructions. Let n , p and $q > p$ be integers.

Definition 1 (Key-homomorphic PRF). A function $F : \mathbb{Z}_q^n \times \{0, 1\}^\ell \rightarrow \mathbb{Z}_p$ is a key-homomorphic PRF if it is a PRF, and for all $k_1, k_2 \in \mathbb{Z}_q$ and $x \in \{0, 1\}^\ell$ it holds that:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) \in \mathbb{Z}_p$$

We do not know of any PRFs satisfying the above property, where the homomorphism is additive over both the inputs and the outputs, so instead use the following, weaker definition.

Definition 2 (Almost key-homomorphic PRF). A function $F : \mathbb{Z}_q^n \times \{0, 1\}^\ell \rightarrow \mathbb{Z}_p$ is an almost key-homomorphic PRF if it is a PRF, and for all $k_1, k_2 \in \mathbb{Z}_q$ and $x \in \{0, 1\}^\ell$ it holds that:

$$F(k_1 + k_2, x) = F(k_1, x) + F(k_2, x) + e \in \mathbb{Z}_p$$

where $|e| \leq 1$.

To realise this, we use the *rounding* function

$$\lfloor x \rfloor_p = \lfloor x \cdot (p/q) \rfloor$$

which scales $x \in \mathbb{Z}_q$ to lie in the interval $[0, p)$ and then rounds to the nearest integer. We now define the learning with rounding assumption [BPR12].

Definition 3 (Learning with Rounding). Let $n \geq 1$ and $q \geq p \geq 2$ be integers. For a vector $\mathbf{s} \in \mathbb{Z}_q^n$, define the distribution $\text{LWR}_{\mathbf{s}}$ to be the distribution over $\mathbb{Z}_q^n, \mathbb{Z}_p$ obtained by sampling $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, and outputting $(\mathbf{a}, b = \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rfloor_p)$.

The decisional- $\text{LWR}_{n,q,p}$ problem is to distinguish any desired number of samples $(\mathbf{a}_i, b_i) \leftarrow \text{LWR}_{\mathbf{s}}$ from the same number of samples taken from the uniform distribution on $(\mathbb{Z}_q^n, \mathbb{Z}_p)$, where the secret \mathbf{s} is uniform in \mathbb{Z}_q^n .

With this we can easily construct a key-homomorphic PRF in the random oracle model, with the function $F : \mathbb{Z}_q^n \times \{0, 1\}^\ell \rightarrow \mathbb{Z}_p$ defined by

$$F(k, x) = \lfloor \langle k, H(x) \rangle \rfloor_p$$

where $H : \{0, 1\}^\ell \rightarrow \mathbb{Z}_q^n$ is a random oracle. This is an almost-key-homomorphic PRF under the $\text{LWR}_{n,q,p}$ assumption [BPR12].

There are also constructions in the standard model based on learning with rounding or learning with errors [BLMR13, BP14]. All these constructions inherit the same error term, which comes from first computing a function that is linear in the key k , and then rounding the result.

Can we Remove the Homomorphism Error? Applications such as distributed PRFs can work around the error term by suitably rounding the output [BLMR13]. However, in some applications, particularly those in this work, it would be very useful to have a *noise-free* PRF satisfying Definition 1. Two previous works [BV15,BFP+15] claimed that their LWE-based constructions can achieve a slightly weaker notion, where it is *computationally hard* for an adversary to come up with a query x that violates key-homomorphism. Unfortunately, these claims are not correct,³ and it seems difficult to modify these PRFs to satisfy this.

To see why the error seems to be inherent in these PRFs, consider an experiment where we sample random values $r_1, r_2 \leftarrow \mathbb{Z}_q$, and then test whether $[r_1 + r_2]_p = [r_1]_p + [r_2]_p$. This gives us the same result as testing for homomorphism error in F , since H is a random oracle and the two keys are random.⁴ Let $x_1 = [r_1]_p, x_2 = [r_2]_p$ and define the relevant fractional components $e_1 = r_1 \cdot p/q \pmod{1}, e_2 = r_2 \cdot p/q \pmod{1}$, where the reduction modulo 1 is mapped to the interval $[-1/2, 1/2)$. If p divides q then it holds that e_1, e_2 are uniformly random in the set $[-1/2, 1/2) \cap (p/q)\mathbb{Z}$.

If there is no homomorphism error then we have

$$[e_1 + e_2] = [e_1] + [e_2]$$

Clearly when $e_1 \geq 0$ there will be no error as long as $e_2 \leq 0$. Similarly, when $e_2 \geq 0$ and $e_1 \leq 0$ there is no error. These two error-free possibilities cover approximately half of the space of possible choices of $(e_1, e_2) \in [-1/2, 1/2)^2$. For the remaining cases, if we condition on $e_1 \geq 0$ and $e_2 > 0$ then there will be an error whenever $e_1 + e_2 \geq 1/2$, which happens with probability around 1/2. Symmetrically, in the remaining case of $e_2 \geq 0$ and $e_1 > 0$ the error probability is around 1/2, and combining these cases we get an overall error probability of approximately 1/4. The exact error rate depends on whether p divides q and if q/p is even or not, but is nevertheless always close to 1/4.

3 OT Extension Protocol

We now describe our main protocol for extending oblivious transfer.

3.1 Setup Functionality

We use the setup functionality $\mathcal{F}_{\Delta\text{-ROT}}$, shown in Fig. 1, to implement the base OTs in our main protocol. This functionality creates k random, correlated OTs over an abelian group \mathbb{G} (in our protocol we instantiate this with $\mathbb{G} = \mathbb{Z}_q$) where

³ We have confirmed this through personal communication with an author of [BFP+15]. This does not affect the main results of that work or [BV15].

⁴ This method also applies to known standard model KH-PRFs based on LWE, as these constructions all have the form $F(k, x) = [L_x(k)]_p$ for some linear function $L_x : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$.

the sender inputs $\Delta \in \mathbb{G}^n$ and obtains messages of the form $(b_i, b_i + \Delta)$ for randomly sampled $b_i \in \mathbb{G}^n$. The receiver inputs the choice bits $s_i \in \{0, 1\}$ in a setup phase, and during the correlated OT phase it learns a_i , which is either b_i or $b_i + \Delta$, depending on s_i . This Δ -OT stage of the functionality also allows a corrupt sender to attempt to guess (a subset of) the bits s_i , but if the guess fails then the functionality aborts. This leakage is necessary so that we can efficiently implement $\mathcal{F}_{\Delta\text{-ROT}}$, using the protocol we give in Sect. 4.

$\mathcal{F}_{\Delta\text{-ROT}}$ also includes a **Chosen OTs** command, which further extends the base OTs on chosen (but not necessarily correlated) inputs from the sender, using the same choice bits from the receiver.

Functionality $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{G}}$

Parameters: n , the length; k , the number of OTs; \mathbb{G} , a finite, additive abelian group.

Initialize: On receiving $(sid, \text{init}, s_1, \dots, s_k)$ from P_A , where $s_i \in \{0, 1\}$, and (sid, init) from P_B , store $\{s_i\}_i$ and ignore any subsequent (sid, init) messages.

Δ -OTs: On input $(sid, \text{correlated}, \Delta)$ from P_B , where $\Delta \in \mathbb{G}^n$, send $(sid, \text{correlated})$ to P_A and do the following.

1. Sample $b_i \leftarrow \mathbb{G}^n$, for $i \in [k]$. If P_B is corrupted, instead receive b_i from \mathcal{A} .
2. Compute $a_i = b_i + s_i \cdot \Delta$
3. If P_A is corrupted, receive $a_i \in \mathbb{G}^n$ from \mathcal{A} and recompute $b_i = a_i - s_i \cdot \Delta$.
4. If P_B is corrupted, wait for \mathcal{A} to input a message (guess, S) , where S efficiently describes a subset of $\{0, 1\}^k$. If $(s_1, \dots, s_k) \in S$ then send (success) to \mathcal{A} . Otherwise, send \perp to both parties and terminate.
5. Output a_i to P_A and b_i to P_B .

Chosen OTs: On input $(sid, \text{chosen}, B, (u_i^0, u_i^1)_{i=1}^k)$ from P_B , and (sid, chosen, B) from P_A , where $u_i^0, u_i^1 \in [0, B - 1]$, send $(sid, \text{chosen}, (u_i^{s_i})_{i=1}^k)$ to P_A .

Fig. 1. Extended correlated random oblivious transfer functionality over a group \mathbb{G}

3.2 Random OT Protocol

The functionality we implement is shown in Fig. 2. This produces a batch of $m \cdot \ell$ random OTs at once, consisting of m sets of random 1-out-of- p_i OTs, for each $i = 1, \dots, \ell$, where p_i is the i -th prime and ℓ is a parameter of the protocol. Let $P_\ell = 2 \cdot 3 \cdot 5 \cdots p_\ell$ be the product of the first ℓ primes.

The protocol, shown in Fig. 3, starts with a setup phase where the parties perform k correlated OTs using $\mathcal{F}_{\Delta\text{-ROT}}$ over \mathbb{Z}_q^n , where n is the key length of the almost-key-homomorphic PRF $F : \mathbb{Z}_q^n \times \{0, 1\}^k \rightarrow \mathbb{Z}_{P_\ell}$. After this phase, P_R holds random values $\mathbf{r}, \mathbf{b}_i \in \mathbb{Z}_q^n$, whilst P_S holds random $s_i \in \{0, 1\}$ and $\mathbf{a}_i = \mathbf{b}_i + s_i \cdot \mathbf{r}$, for $i = 1, \dots, k$.

In the OT extension phase, the parties expand the base OTs using F , such that the key-homomorphic property of F preserves the correlation between \mathbf{a} and \mathbf{b} , except for a small additive error. We have:

Functionality $\mathcal{F}_{p_i\text{-ROT}}^{m,\ell,k}$

After receiving a message (**send**) from P_S and (**receive**) from P_R , ignore all other messages from both parties and do as follows:

1. Sample choices $c_1^i, \dots, c_m^i \leftarrow \mathbb{Z}_{p_i}$, for $i \in [\ell]$. If P_R is corrupted, instead receive $c_j^i \in \mathbb{Z}_{p_i}$ from \mathcal{A} .
2. For each $j \in [m]$, sample $\sum_i p_i$ sets of strings $(y_j^{i,0}, \dots, y_j^{i,p_i-1})_{i=1}^\ell$, where each $y_j^{i,c} \leftarrow \{0, 1\}^k$.
3. Output $(y_j^{i,0}, \dots, y_j^{i,p_i-1})$ to P_S and (c_j^i, y_j^{i,c_j^i}) to P_R , for $i \in [\ell]$ and $j \in [m]$.

Fig. 2. Functionality for m sets of random $\{1\text{-out-of-}p_i\}_{i=1}^\ell$ OTs on k -bit strings

$$F(\mathbf{a}_i, j) = F(\mathbf{b}_i + s_i \cdot \mathbf{r}, j) = F(\mathbf{b}_i, j) + s_i \cdot (F(\mathbf{r}, j) + e_i)$$

where $|e_i| \leq 1$ (note that e_i depends on j , but we often omit the subscript- j to simplify notation).

Since P_R knows both \mathbf{b}_i and \mathbf{r} , it can compute e_i , and then use the base OTs to obliviously transfer either $u_i + e_i$ (if $s_i = 1$) or u_i (if $s_i = -1$) to P_S , where u_i is a uniformly random value in $\{0, \dots, B - 1\}$, and B is superpolynomial in the security parameter so that u_i statistically masks e_i .

After step 2f, if $u_i + e_i \notin \{-1, B\}$ then we have:

$$\begin{aligned} q_i &= q'_i - v_i = t'_i + s_i \cdot (x_j + e_i) - u_i - s_i \cdot e_i \\ &= t'_i - u_i + s_i \cdot x_j \\ &= t_i + s_i \cdot x_j \pmod{P_\ell} \end{aligned}$$

For each $j \in [m + 1]$, these k sets of correlated OTs are then placed into vectors \mathbf{q}_j and \mathbf{t}_j , which satisfy $\mathbf{q}_j = \mathbf{t}_j + x_j \cdot \mathbf{s}$. To convert these into random 1-out-of- p_i OTs, for $i = 1, \dots, \ell$, each \mathbf{t}_j is reduced modulo p_i and then hashed with the random oracle to produce the receiver’s output string. The c -th output of the sender, for $c \in \{0, \dots, p_i - 1\}$, in the (i, j) -th OT is defined as:

$$H(\mathbf{q}_j - c \cdot \mathbf{s} \pmod{p_i}) = H(\mathbf{t}_j + (x_j - c) \cdot \mathbf{s} \pmod{p_i})$$

which for $c = x_j$ is equal to the receiver’s output, as required. The sender’s other outputs are computationally hidden to the receiver, since to compute them it would have to query $\mathbf{q}_j - x' \cdot \mathbf{s} \pmod{p_i}$ to the random oracle for some $x' \neq x_j$, but this requires guessing \mathbf{s} .

The only opportunity for a malicious receiver to misbehave in the protocol is when sending the $(u_i, u_i + e_i)$ values to the base OT functionality, to correct the errors. The consistency check phase prevents this, by opening a random linear combination of the correlated OTs and checking that the linear relation still holds. We must then discard the $(m + 1)$ -th set of OTs so that the opened values do not reveal anything about the receiver’s outputs. This check allows a corrupt

Protocol $\Pi_{p_i\text{-ROT}}^{m,\ell,k}$

Let $F : \mathbb{Z}_q^n \times \{0, 1\}^k \rightarrow \mathbb{Z}_{P_\ell}$ be an almost-key-homomorphic PRF.
 Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ be a random oracle and $G : \{0, 1\}^k \rightarrow \mathbb{Z}_{P_\ell}^m$ be a PRG.

1. *Setup phase.*

- (a) P_S samples $s_1, \dots, s_k \leftarrow \{0, 1\}$, and P_R samples $\mathbf{r} \leftarrow \mathbb{Z}_q^n$
- (b) Both parties initialize $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$, where P_S inputs (s_1, \dots, s_k) . P_R then calls the functionality again with input (**correlated**, \mathbf{r}).
- (c) P_R receives $\mathbf{b}_1, \dots, \mathbf{b}_k$ and P_S receives $\mathbf{a}_1, \dots, \mathbf{a}_k$ such that $\mathbf{a}_i = \mathbf{b}_i + s_i \cdot \mathbf{r}$.

2. *Extension phase.*^a P_S initializes a zero matrix $\mathbf{Q} \in \mathbb{Z}_{P_\ell}^{k \times (m+1)}$. P_R initializes a zero matrix $\mathbf{T} \in \mathbb{Z}_{P_\ell}^{k \times (m+1)}$, and a vector $\mathbf{x} \in \mathbb{Z}_{P_\ell}^{m+1}$.

For each $j \in [m + 1]$:

- (a) P_S computes $q'_i = F(\mathbf{a}_i, j) \in \mathbb{Z}_{P_\ell}$, for $i \in [k]$
- (b) P_R computes $t'_i = F(\mathbf{b}_i, j)$, $x_j = F(\mathbf{r}, j) \in \mathbb{Z}_{P_\ell}$
- (c) P_R computes the errors

$$e_i = F(\mathbf{b}_i + \mathbf{r}, j) - t'_i - x_j \in \{0, 1\}, \quad \text{for } i \in [k]$$

and samples $u_i \leftarrow [0, B - 1]$

- (d) P_R calls $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$ on input (**chosen**, B , $(u_i, u_i + e_i \bmod B)$)
- (e) P_S receives $v_i = u_i + s_i \cdot e_i \bmod B$ from $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$
- (f) P_S defines $q_i = q'_i - v_i$, and P_R defines $t_i = t'_i - u_i$

It should now hold that

$$q_i = t_i + s_i \cdot x_j \quad \text{mod } P_\ell$$

- (g) P_S stores $\mathbf{q}_j = (q_1, \dots, q_k)$ in column j of the matrix \mathbf{Q} .
- (h) P_R stores $\mathbf{t}_j = (t_1, \dots, t_k)$ in column j of \mathbf{T} , and x_j in entry j of the vector \mathbf{x} .

3. *Consistency check.*

- (a) P_S samples a seed $\sigma \leftarrow \{0, 1\}^k$ and sends this to P_R
- (b) Both parties compute $\boldsymbol{\alpha} = (G(\sigma) \| 1)$.
- (c) P_R computes and sends

$$\tilde{\mathbf{t}} = \mathbf{T}\boldsymbol{\alpha}, \quad \tilde{\mathbf{x}} = \mathbf{x}^\top \boldsymbol{\alpha}$$

- (d) P_S checks that $\mathbf{Q}\boldsymbol{\alpha} = \tilde{\mathbf{t}} + \tilde{\mathbf{x}}\mathbf{s}$

4. *Output:* P_S samples a seed $\rho \leftarrow \{0, 1\}^k$ and sends this to P_R . The parties then compute their outputs as follows:

- P_S outputs, for $j \in [m]$ and $i \in [\ell]$:

$$H(i, \rho, \mathbf{q}_j^i), H(i, \rho, \mathbf{q}_j^i - \mathbf{s}), \dots, H(i, \rho, \mathbf{q}_j^i - (p_i - 1) \cdot \mathbf{s}), \quad \text{where } \mathbf{q}_j^i = \mathbf{q}_j \bmod p_i$$

- P_R outputs, for $j \in [m]$ and $i \in [\ell]$:

$$x_j^i, H(i, \rho, \mathbf{t}_j^i), \quad \text{where } x_j^i = x_j \bmod p_i, \quad \mathbf{t}_j^i = \mathbf{t}_j \bmod p_i$$

^a Steps 2–4 can be iterated by maintaining j as a counter.

Fig. 3. Random 1-out-of- p_i OT extension protocol

receiver to attempt to guess a few of the s_i bits by cheating in only a few OT instances. However, this is exactly the same behaviour that is already allowed by the $\mathcal{F}_{\Delta\text{-ROT}}$ functionality for the base OTs. It does not pose a problem for security because in the output phase \mathbf{s} is put through a random oracle, and the whole of \mathbf{s} must be guessed to break security.

3.3 Security

Theorem 1. *Let $B = \Theta(2^\lambda)$, $\ell = \Omega(k\lambda)$, F be an almost key-homomorphic PRF and H be a random oracle. Then protocol $\Pi_{p_i\text{-ROT}}^{m,\ell,k}$ securely realises the functionality $\mathcal{F}_{p_i\text{-ROT}}^{m,\ell,k}$ with static security in the $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$ -hybrid model.*

We prove this by considering separately the cases of a corrupt sender and a corrupt receiver. Security when both parties are honest, or both corrupt, is straightforward.

Corrupt sender. This is the simpler of the two cases. We construct an ideal-world simulator, \mathcal{S}_S , shown in Fig. 4. The simulator uses random values to simulate the v_i messages sent to P_S during the OT extension phase, then samples \tilde{x} at random to respond to the consistency check, computing $\tilde{\mathbf{t}}$ such that the check will always pass. The random oracle queries are responded to using knowledge of the sender’s bits \mathbf{s} from the setup phase, so as to be consistent with the random sender messages obtained from $\mathcal{F}_{p_i\text{-ROT}}$. All other queries are responded honestly, at random. The security of the protocol against a corrupt sender rests on two key points: (1) $B = \Theta(2^\lambda)$, so that $u_i + e_i$ statistically masks the errors e_i in the protocol, and (2) The security of the key-homomorphic PRF, which implies the x_j outputs of the honest receiver are pseudorandom, and also the simulated \tilde{x} is indistinguishable from the real value in the protocol.

Lemma 1. *For every adversary \mathcal{A} who corrupts P_S , and for every environment \mathcal{Z} , it holds that*

$$\text{IDEAL}_{\mathcal{F}_{p_i\text{-ROT}}, \mathcal{S}_S, \mathcal{Z}} \stackrel{c}{\approx} \text{HYBRID}_{\Pi_{p_i\text{-ROT}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\Delta\text{-ROT}}}$$

Proof. Recall that as well as seeing the view of \mathcal{A} during the protocol execution, the environment \mathcal{Z} learns the outputs of both parties. We prove security by defining a sequence of hybrid executions, where H_0 is defined to be the ideal process and each successive hybrid modifies the previous execution in some way.

Hybrid H_1 : Instead of sampling v_i at random, \mathcal{S}_S sends $v_i = u_i + s_i \cdot e_i \bmod B$, where $u_i \leftarrow [0, B - 1]$ and e_i is computed as in the protocol, using randomly sampled $\mathbf{r} \in \mathbb{Z}_q^n$ and $\mathbf{b}_i := \mathbf{a}_i - s_i \cdot \mathbf{r}$.

Hybrid H_2 : Instead of sampling \tilde{x} at random, let x_1, \dots, x_m be the outputs of the honest receiver (from $\mathcal{F}_{p_i\text{-ROT}}$), and sample $x_{m+1} \leftarrow \mathbb{Z}_{P_\ell}$. \mathcal{S}_S then sends $\tilde{x} = \mathbf{x}^\top \boldsymbol{\alpha}$.

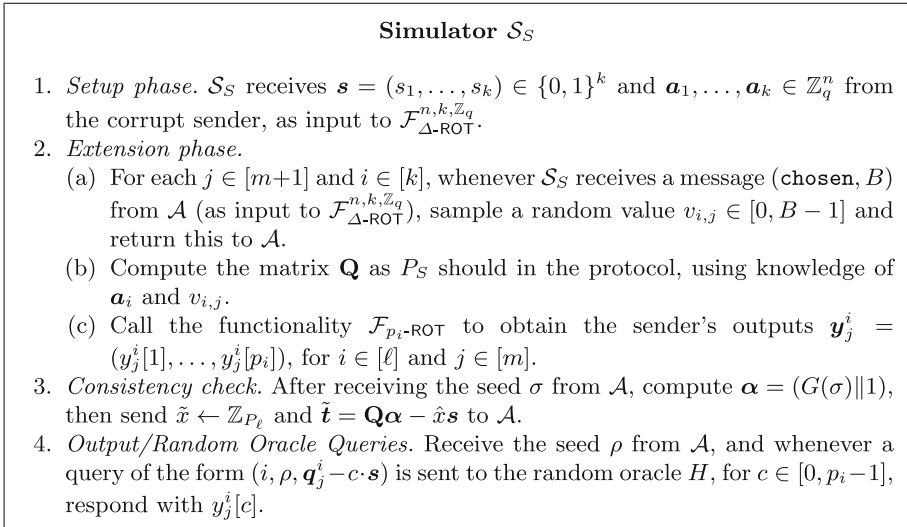


Fig. 4. Simulator for a corrupted sender

Hybrid H_3 : This is defined the same as H_2 , except the random choices x_1, \dots, x_{m+1} are replaced with values computed from the key-homomorphic PRF as $x_j = F(\mathbf{r}, j)$, using the previously sampled \mathbf{r} .

Note that all of the simulated messages in the final hybrid, H_3 , are identically distributed to messages sent in the real execution, and the outputs of the sender and receiver are computed exactly as in the protocol as outputs of the random oracle H and PRF F , respectively. Therefore, $H_3 \equiv \text{HYBRID}$.

Hybrids H_0 and H_1 are identically distributed as long as $u_i + s_i \cdot e_i \notin \{-1, B\}$, since $e_i \in \{0, \pm 1\}$. If the reduction modulo B overflows then \mathcal{Z} could distinguish because the outputs in H_1 will be incorrect. This occurs with probability at most $2/B$ for each i , so when $B = \Omega(2^\lambda)$ we have $H_0 \stackrel{s}{\approx} H_1$.

In hybrid H_2 , the value \tilde{x} is masked by x_{m+1} so is uniformly random in the view of \mathcal{Z} . Therefore, $H_1 \equiv H_2$.

Hybrids H_2 and H_3 are computationally indistinguishable, by a standard reduction to the key-homomorphic PRF, since \mathbf{r} is uniformly random and not seen by the environment, so the x_j values are pseudorandom. This completes the claim that $\text{IDEAL} \stackrel{c}{\approx} \text{HYBRID}$. □

Corrupt receiver. The simulator \mathcal{S}_R , given in Fig. 5, essentially runs an internal copy of the sender and honestly generates messages as P_S would. The main challenge is to extract the inputs of the corrupt P_R , and also show that \mathcal{Z} cannot query the random oracle H on a value corresponding to one of the sender's random outputs that was not chosen by the receiver.

Simulator \mathcal{S}_R

1. *Setup phase.* \mathcal{S}_R receives \mathbf{r} and $\mathbf{b}_1, \dots, \mathbf{b}_k$ from the corrupt receiver, as input to $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$. It then samples $\mathbf{s} \leftarrow \{0, 1\}^k$ and defines $\mathbf{a}_i = \mathbf{b}_i + s_i \mathbf{r}$.
For any key query guess submitted by \mathcal{A} to $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$, \mathcal{S} responds according to the secret \mathbf{s} . If any guess is unsuccessful, \mathcal{S} aborts.
2. *Extension phase.*
 - (a) \mathcal{S} waits for P_R to send a message of the form $(\text{chosen}, B, u_i, u'_i)$ as input to $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{Z}_q}$. \mathcal{S} then defines $e'_i = u'_i - u_i \bmod B$ and $v_i = u_i + s_i \cdot e'_i$.
 - (b) \mathcal{S} computes q_i as an honest sender would, and defines the matrix \mathbf{Q} .
3. *Consistency Check.*
 - (a) \mathcal{S} samples $\sigma \leftarrow \{0, 1\}^k$ and sends this to P_R .
 - (b) \mathcal{S} receives $\tilde{\mathbf{t}}, \tilde{\mathbf{x}}$ and checks that $\mathbf{Q}\boldsymbol{\alpha} = \tilde{\mathbf{t}} + \tilde{\mathbf{x}}\sigma$. If the check fails, abort.
 - (c) Extract the inputs x'_j using Proposition 1.
 - (d) Call $\mathcal{F}_{p_i\text{-ROT}}$ and send the choices $\{x'_j \bmod p_i\}_{i \in [\ell]}$, for $j \in [m]$. Receive back the OT outputs y_j^i , for $i \in [\ell], j \in [m]$.
4. *Output phase/Random Oracle Queries.* \mathcal{S} sends a random seed $\rho \leftarrow \{0, 1\}^k$, and responds to the random oracle queries as follows:
 - (a) If a query is $(i, \rho, (\mathbf{q}_j - x'_j \mathbf{s}) \bmod p_i)$ for some $i \in [\ell], j \in [m]$ then respond with y_j^i . Otherwise, respond at random (consistent with previous queries).

Fig. 5. Simulator for a corrupted receiver

We use the following technical lemma to analyse the soundness of the consistency check when taking random linear combinations over the ring \mathbb{Z}_{P_ℓ} .

Lemma 2. *Let $\mathbf{E} \in \mathbb{Z}_{P_\ell}^{k \times (m+1)}$. Suppose that every column \mathbf{e}_i of \mathbf{E} satisfies $\|\mathbf{e}_i\|_\infty \leq B$, and further that there is at least one column not in $\text{span}(\mathbf{1})$. Then,*

$$\Pr_{\alpha \leftarrow \mathbb{Z}_{P_\ell}^m \times \{1\}} [\mathbf{E}\boldsymbol{\alpha} \in \text{span}(\mathbf{1})] \leq 2B/P_\ell$$

Proof. From the assumption that at least one column of \mathbf{E} not in $\text{span}(\mathbf{1})$, there exist two rows \mathbf{a}, \mathbf{b} of \mathbf{E} with $\mathbf{a} \neq \mathbf{b}$. If $\mathbf{E}\boldsymbol{\alpha} \in \text{span}(\mathbf{1})$ then $\langle \mathbf{a}, \boldsymbol{\alpha} \rangle = \langle \mathbf{b}, \boldsymbol{\alpha} \rangle$ and so $\langle \mathbf{a} - \mathbf{b}, \boldsymbol{\alpha} \rangle = 0$. Let $\mathbf{d} = \mathbf{a} - \mathbf{b}$, and j be an index where $d_j \neq 0$. Then $\langle \mathbf{d}, \boldsymbol{\alpha} \rangle = 0$ if and only if $d_j \alpha_j = -\sum_{i \neq j} d_i \alpha_i$.

First consider the case that $j \neq m + 1$, so α_j is uniform in \mathbb{Z}_{P_ℓ} . For any fixed choice of d_j , the number of distinct possibilities for $d_j \alpha_j \bmod P_\ell$ is given by the order of d_j in the additive group \mathbb{Z}_{P_ℓ} , which equals $P_\ell / \gcd(d_j, P_\ell)$. Since $|a_j|, |b_j| \leq B$, we have $d_j \in [0, 2B] \cup [P_\ell - 2B, P_\ell - 1]$, from which it follows that $\gcd(d_j, P_\ell) \leq 2B$. Therefore, since α_j is random and independent of $\sum_{i \neq j} \alpha_i d_i$, we have that $\Pr[\langle \mathbf{d}, \boldsymbol{\alpha} \rangle = 0] \leq 2B/P_\ell$.

On the other hand, if $j = m + 1$ then $\alpha_j = 1$. But this means $d_j = -\sum_{i \neq j} d_i \alpha_i$, and because $d_j \neq 0$ there must be another index j' with $d_{j'} \neq 0$. We then apply the previous argument on j' to obtain the same probability. \square

If $\boldsymbol{\alpha}$ is sampled using a PRG with a public seed, instead of uniformly at random, the previous statement still holds except with negligible probability.

Lemma 3. *Let \mathbf{E} be as in Lemma 2 and let $G : \{0, 1\}^k \rightarrow \mathbb{Z}_{P_\ell}^m \times \{1\}$ be a PRG. Then,*

$$\Pr_{\sigma \leftarrow \{0,1\}^k} [\mathbf{E}\boldsymbol{\alpha} \in \text{span}(\mathbf{1}) : \boldsymbol{\alpha} = G(\sigma)] \leq 2B/P_\ell + \text{negl}(k)$$

Proof. Define a distinguisher, D , for the PRG G , which on input a challenge $\boldsymbol{\alpha}$, outputs 1 if $\mathbf{E}\boldsymbol{\alpha} \in \text{span}(\mathbf{1})$ and 0 otherwise. From Lemma 2 we know that $\Pr[D = 1]$ given that $\boldsymbol{\alpha}$ is uniformly random is $\leq 2B/P_\ell$. On the other hand, the advantage of D is $\text{negl}(k)$, so if $\boldsymbol{\alpha}$ is an output of G then it must be that D outputs 1 with probability at most $2B/P_\ell + \text{negl}(k)$. \square

We now show indistinguishability of the simulation.

Lemma 4. *For every adversary \mathcal{A} who corrupts P_R , and for every environment \mathcal{Z} , it holds that*

$$\text{IDEAL}_{\mathcal{F}_{P_i\text{-ROT}}, \mathcal{S}_R, \mathcal{Z}} \stackrel{c}{\approx} \text{HYBRID}_{\Pi_{P_i\text{-ROT}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\Delta\text{-ROT}}}$$

Proof. We first show how \mathcal{S}_R (Fig. 5) extracts the corrupt receiver’s inputs in step 3c. \mathcal{S}_R received the values $u_i, u'_i = u_i + e'_i$ which \mathcal{A} used as input to $\mathcal{F}_{\Delta\text{-ROT}}$. \mathcal{S}_R can also compute the actual errors e_i (which would equal e'_i if P_R was honest), since it knows \mathbf{r} and \mathbf{b}_i . For each $j \in [m + 1]$ and $i \in [k]$, \mathcal{S}_R defines a value (omitting j subscripts) $q_i = q'_i - (u_i + s_i e'_i)$, and then puts all these values into the vector \mathbf{q}_j . We also compute the values x_j and \mathbf{t}_j as an honest P_R would do according to the protocol.

Now, if P_R was honest we would have $\mathbf{q}_j = \mathbf{t}_j + x_j \cdot \mathbf{s}$, but it actually holds that

$$\mathbf{q}_j = \mathbf{t}_j + x_j \cdot \mathbf{s} + \mathbf{e}_j * \mathbf{s}$$

where \mathbf{e}_j contains the values $(e_1 - e'_1, \dots, e_k - e'_k)$ from iteration j of this phase, and $*$ denotes component-wise product. Note that since $e_i \in \{0, \pm 1\}$ and $e'_i \in \{0, \dots, B - 1\}$ we have $\|\mathbf{e}_j\|_\infty \leq B$ for all j .

At this point, although we have computed values x_j which could be used to define the inputs of P_R , these may *not* be the correct inputs \mathcal{S}_R should send to $\mathcal{F}_{P_i\text{-ROT}}$. This is because \mathcal{A} could choose, for instance, $\mathbf{e}_j = \mathbf{1}$, and then the actual inputs would correspond to $x_j + 1$ and not x_j . Proposition 1 shows how we obtain the correct inputs. Let $\text{view}(\mathcal{A})$ denote the view of the corrupt receiver at this point in the execution.

Proposition 1. *Suppose the consistency check passes, and \mathcal{A} makes no (guess) queries to $\mathcal{F}_{\Delta\text{-ROT}}$. Then with overwhelming probability there exists a set $S \subset [k]$ and values x'_j, \mathbf{e}'_j , for $j \in [m]$ such that:*

1. $\mathbf{q}_j = \mathbf{t}_j + x'_j \cdot \mathbf{s} + \mathbf{e}'_j * \mathbf{s}$.
2. For all $i \in [k] \setminus S$, $\mathbf{e}'_j[i] = 0$
3. $H_\infty((s_i)_{i \in S} | \text{view}(\mathcal{A})) = 0$
4. $H_\infty((s_i)_{i \in [k] \setminus S} | \text{view}(\mathcal{A})) = k - |S|$.

Proof. Recall that $\tilde{\mathbf{t}}, \tilde{x}$ are the values received by \mathcal{S}_R from P_R during the consistency check, and we have $\mathbf{q}_j = \mathbf{t}_j + x_j \cdot \mathbf{s} + \mathbf{e}_j * \mathbf{s}$.

This means we can write

$$\mathbf{Q} = \mathbf{T} + \underbrace{(x_1 \cdot \mathbf{1} + \mathbf{e}_1 \parallel \cdots \parallel x_m \cdot \mathbf{1} + \mathbf{e}_m)}_{=\mathbf{Y}} * \mathbf{s}$$

where we extend the $*$ operator to apply to every column of \mathbf{Y} in turn.

Define the vectors, in $\mathbb{Z}_{P_\ell}^k$,

$$\delta_x = \mathbf{1}\tilde{x} - \mathbf{Y}\alpha, \quad \delta_t = \mathbf{T}\alpha - \tilde{\mathbf{t}}$$

We can think of these as representing the deviation between what P_R sent, and the values P_R *should have* sent, given \mathbf{Y}, \mathbf{T} . If the check succeeds, then we know that

$$\mathbf{Q}\alpha = \tilde{\mathbf{t}} + (\mathbf{1}\tilde{x}) * \mathbf{s}$$

and so

$$\begin{aligned} (\mathbf{T} + \mathbf{Y} * \mathbf{s})\alpha &= \tilde{\mathbf{t}} + (\mathbf{1}\tilde{x}) * \mathbf{s} \\ \Leftrightarrow \delta_t &= (\mathbf{1}\tilde{x}) * \mathbf{s} - (\mathbf{Y} * \mathbf{s})\alpha \\ &= \delta_x * \mathbf{s} \end{aligned}$$

For each index $i \in [k]$, if the check passes then it must hold that either $\delta_t[i] = \delta_x[i] = 0$ —which essentially means there was no deviation at position i —or $\delta_x[i] \neq 0$. In the latter case, because $s_i \in \{0, 1\}$, the cheating receiver must guess s_i in order to pass the check.

Define $S \subset [k]$ to be the set of all indices i for which $\delta_x[i] \neq 0$. From the above, we have that the probability of passing the check (over the random choice of \mathbf{s}) is at most $2^{-|S|}$. If the check passes, this also implies the last two claims of the proposition, that $H_\infty((s_i)_{i \in S} | \text{view}(\mathcal{A})) = 0$, and $H_\infty((s_i)_{i \in [k] \setminus S} | \text{view}(\mathcal{A})) = k - |S|$.

Let \mathbf{Y}_{-S} denote the matrix \mathbf{Y} with its rows corresponding to indices in the set S removed. Note that for any $i \notin S$, we have $\delta_x[i] = 0$ and so $(\mathbf{Y}\alpha)_{-S} = (\mathbf{Y}_{-S})\alpha = \mathbf{1}\tilde{x}$, which lies in $\text{span}(\mathbf{1})$. Since column j of \mathbf{Y} is equal to $\mathbf{1}x_j + \mathbf{e}_j$, it must also hold that $(\mathbf{E}_{-S})\alpha \in \text{span}(\mathbf{1})$, where $\mathbf{E} = (\mathbf{e}_1 \parallel \cdots \parallel \mathbf{e}_{m+1})$.

Applying Lemma 3 with \mathbf{E}_{-S} , it then holds that every column of \mathbf{E}_{-S} is in $\text{span}(\mathbf{1})$ with overwhelming probability, provided $2B/P_\ell$ is negligible. Therefore, for every $j \in [m]$, we can compute x'_j and \mathbf{e}'_j such that the j -th column of \mathbf{Y} satisfies $\mathbf{y}_j = x'_j \cdot \mathbf{1} + \mathbf{e}'_j$, where $(\mathbf{e}'_j)_{-S} = 0$. These are values needed to satisfy points 1 and 2. \square

The set S in the proposition represents the indices where P_R cheated, corresponding to the set of bits of \mathbf{s} which P_R must guess to pass the consistency check. Passing the check also guarantees that the error vectors \mathbf{e}'_j can only be non-zero in the positions corresponding to S , which is crucial for the next stage.

After extracting the corrupt receiver’s inputs, we need to show that the random oracle calls made by \mathcal{Z} cannot allow it to distinguish. In particular, if \mathcal{Z} queries

$$(i, \rho, (\mathbf{q}_j - y_j \mathbf{s}) \bmod p_i)$$

for some $y_j \neq x'_j \bmod p_i$ then \mathcal{Z} will be able to distinguish, since the simulator’s response will be random, whereas the response in the real world will be one of the sender’s OT outputs.

From Proposition 1, we know that if no (guess) queries were made to $\mathcal{F}_{\Delta\text{-ROT}}$ then there are exactly $k - |S|$ bits of the secret \mathbf{s} that are unknown in the view of \mathcal{A} , and these correspond to the index set $[k] \setminus S$.

Now, from the first part of the proposition, we can rewrite a ‘bad’ query of the form given above as

$$(i, \rho, (\mathbf{t}_j + \mathbf{e}'_j * \mathbf{s} + (x'_j - y_j) \mathbf{s}) \bmod p_i)$$

Since \mathbf{t}_j and $\mathbf{e}'_j * \mathbf{s}$ are fixed in the view of \mathcal{Z} , it must be the case that coming up with such a query requires knowing all of \mathbf{s} . This happens with probability at most $(p_i - 1) \cdot 2^{|S|-k}$ per query with index i . Taking into account the probability of passing the consistency check, we get an overall success probability bounded by $Q \cdot (p_\ell - 1) \cdot 2^{-k}$, where Q is the number of random oracle queries, which is negligible. Making key queries to $\mathcal{F}_{\Delta\text{-ROT}}$ cannot help guess \mathbf{s} because any incorrect guess causes an abort, so this does not affect the distinguishing probability. \square

3.4 Choosing the Parameters

We first show how to securely choose parameters to optimize communication, and then discuss instantiating the key-homomorphic PRF. After the setup phase, and ignoring the short seeds sent in the consistency check, the only communication is to the (chosen) command of $\mathcal{F}_{\Delta\text{-ROT}}$, which can be implemented with λ bits of communication when $B = 2^\lambda$ (see Sect. 4). This gives an overall complexity of λkm bits to generate $m\ell$ random OTs. If $\ell = \omega(\lambda k)$ then we obtain an amortized cost per random OT of $o(1)$, which gets smaller as ℓ increases.

To realise 1-out-of-2 bit-OT on chosen strings, each random 1-out-of- p_i OT must be converted to a 1-out-of-2 OT, at a cost of sending $\log p_i$ bits from the receiver and 2 bits from the sender (see Appendix A). This adds a cost of $T_{m,\ell} = m \sum_{i=1}^\ell (\log_2 p_i + 2)$ bits, and from the prime number theorem we have $p_i = O(i \log i)$, so $T_{m,\ell} = m \sum_{i=1}^\ell O(\log i) = O(m\ell \log \ell)$, giving an overall, amortized cost of $O(\log \ell) = O(\log k)$ bits per OT when $\ell = \Omega(\lambda k)$.

Instantiating the Key-Homomorphic PRF. We can instantiate F using the random oracle-based construction from Sect. 2 based on learning with rounding, or standard model constructions from LWE [BLMR13, BP14]. With LWR, the parameters affecting security are the dimension n and moduli p, q . In our case we

fix $p = P_\ell$ and can choose n, q to ensure security. With an exponential modulus, we know that LWR is at least as hard as LWE with the same dimension n and modulus q , where the LWE error distribution is bounded by $\beta = q/(2^\lambda P_\ell)$, and λ is a statistical security parameter [BPR12, AKPW13]. This gives a modulus-to-noise ratio of $q/\beta = O(2^\lambda P_\ell)$. LWE with an exponential modulus-to-noise ratio has previously been used to construct attribute-based encryption [GVW13] and fully key-homomorphic encryption [BGG+14], and is believed to be hard if $q/\beta \leq 2^{n^\epsilon}$, for some constant $0 < \epsilon < 1/2$ chosen to resist known attacks based on lattice reduction and BKW. To achieve optimal communication in our protocol, we need $\ell = \omega(\lambda k)$, which from the prime number theorem implies that $\log P_\ell = \omega(\lambda k \log k)$. This gives $\log(q/\beta) = \omega(\lambda^2 k \log k)$, so we can have a dimension of $n = \omega((\lambda^2 k \log k)^{1/\epsilon})$ and ensure security.

4 Actively Secure Base OTs

We now show how to implement the functionality $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{G}}$ (Fig. 1), which creates k correlated base OTs over \mathbb{G}^n , for an additive abelian group \mathbb{G} . We achieve active security using a modification of the consistency check from the OT extension protocol by Asharov et al. [ALSZ15, ALSZ17a]. Additionally, in Sect. 4.1 we identify a crucial security flaw in their protocol, whereby a passively corrupted sender can obtain an ‘oracle’ that allows brute-force guessing of the receiver’s choice bits by computing hash values. This bug has since been fixed in a revised version [ALSZ17b], and we apply the same fix to our protocol.

We let $\mathcal{F}_{\text{OT}}^{k,k}$ denote the standard functionality for k sets of 1-out-of-2 OTs on k -bit strings. In the correlated OT phase of our protocol, shown in Fig. 6, the parties first extend the base OTs from \mathcal{F}_{OT} using a PRF, and the sender P_S (who would be receiver when running the main OT extension protocol) then sends the u_i values which create the correlation over the group \mathbb{G} . The consistency check is based on the check in the OT extension protocol by Asharov et al. [ALSZ15], which is used to verify the sender’s inputs are bit strings of the form $(b_i, b_i \oplus \Delta)$. We adapt this to ensure they have the form $(b_i, b_i + \Delta)$, where Δ and each b_i are vectors of length n over any finite abelian group \mathbb{G} . In our protocol the parties then output the correlated base OTs, instead of transposing and hashing them to perform OT extension as in [ALSZ15]. This means we need to account for some leakage on the s_i choice bits of P_R , caused by the consistency check, which is modeled by the key query feature of $\mathcal{F}_{\Delta\text{-ROT}}$.

We also have an additional (non-correlated) **Chosen OTs** phase, which extends the base OTs further with arbitrary inputs from the sender, P_S , and the same choice bits from P_R , in a standard manner using the PRF. Both of these phases can be called repeatedly after the setup phase has run.

4.1 Security

We prove the following theorem by considering separately the two cases of a corrupted P_R and corrupted P_S .

Protocol $\Pi_{\Delta\text{-ROT}}^{n,k,\mathbb{G}}$

Let $F : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \mathbb{G}^{n+k'}$ be a PRF and $H : \mathbb{G}^{n+k'} \times \mathbb{G}^{n+k'} \rightarrow \{0, 1\}^k$ be a random oracle, where k' is chosen so that $|\mathbb{G}|^{k'} \geq 2^k$.

The protocol consists of two main commands, which can be repeatedly called after the **Initialize** stage. Both parties maintain a counter $c := 0$.

Initialize: On input $(\text{init}, s_1, \dots, s_k)$ from P_R and (init) from P_S , where $(s_1, \dots, s_k) \in \{0, 1\}^k$:

1. P_S samples $k_i^0, k_i^1 \leftarrow \{0, 1\}^k$ for $i \in [k]$
2. The parties run $\mathcal{F}_{\text{OT}}^{k,k}$, where P_S is sender with input $(k_i^0, k_i^1)_i$ and P_R is receiver with input s_i .
3. P_R receives $k_i^{s_i}$, for $i = 1, \dots, k$.

Δ -OTs: On input (correlated) from P_R , and $(\text{correlated}, \Delta)$ from P_S , where $\Delta \in \mathbb{G}^n$:

1. *Create correlation:*
 - (a) P_S samples $\rho \leftarrow \mathbb{G}^{k'}$ and sets $\Delta' := (\Delta \parallel \rho)$.
 - (b) P_S computes $t_i^0 = F(k_i^0, c)$ and $t_i^1 = F(k_i^1, c)$, then computes

$$u_i = t_i^0 + t_i^1 + \Delta', \quad i = 1, \dots, k$$

and sends these to P_R .

- (c) P_R computes $a_i = (-1)^{s_i} \cdot F(k_i^{s_i}, c) + s_i \cdot u_i$ ($= t_i^0 + s_i \cdot \Delta'$)
- (d) Set $c := c + 1$.
2. *Consistency Check:* For every pair $(\alpha, \beta) \in [k]^2$:
 - (a) P_S computes

$$\begin{aligned} h_{\alpha,\beta}^{0,0} &= H(t_\alpha^0 - t_\beta^0), & h_{\alpha,\beta}^{0,1} &= H(t_\alpha^0 - t_\beta^1) \\ h_{\alpha,\beta}^{1,0} &= H(t_\alpha^1 - t_\beta^0), & h_{\alpha,\beta}^{1,1} &= H(t_\alpha^1 - t_\beta^1) \end{aligned}$$

and sends these to P_R .

- (b) P_R checks that:
 - i. $h_{\alpha,\beta}^{s_\alpha, s_\beta} = H(t_\alpha^{s_\alpha} - t_\beta^{s_\beta})$
 - ii. $h_{\alpha,\beta}^{\bar{s}_\alpha, \bar{s}_\beta} = H(u_\alpha - u_\beta - t_\alpha^{s_\alpha} + t_\beta^{s_\beta})$
 - iii. $u_\alpha \neq u_\beta$

3. *Output:* P_R outputs the first n components of a_i , and P_S outputs n components of $b_i := t_i^0$.

Chosen OTs: On input $(\text{chosen}, B, (u_i^0, u_i^1)_{i \in [k]})$ from P_S and (chosen, B) from P_R , where each $u_i^b \in [0, B - 1]$ and $B \leq 2^k$:

1. P_S sends $d_i^0 = F(k_i^0, c) \oplus u_i^0$ and $d_i^1 = F(k_i^1, c) \oplus u_i^1$ to P_R , for $i \in [k]$
2. P_R outputs $v_i = d_i^{s_i} \oplus F(k_i^{s_i}, c)^a$
3. Set $c := c + 1$

^a Only the first $\log_2 B$ output bits of the PRF are used in this stage.

Fig. 6. Base OT protocol for correlated OTs over an additive abelian group \mathbb{G}

Theorem 2. *If F is a secure PRF, H is a random oracle and $\lambda \leq k/2$ then protocol $\Pi_{\Delta\text{-ROT}}^{n,k,\mathbb{G}}$ securely realises the functionality $\mathcal{F}_{\Delta\text{-ROT}}^{n,k,\mathbb{G}}$ in the $\mathcal{F}_{\text{OT}}^{k,k}$ -hybrid model with static security.*

Corrupt P_R . To be secure against a corrupted P_R , it is vital that P_S appends the additional randomness ρ to the input Δ in step 1a, before creating the correlated OTs. Without this, P_R can obtain an ‘oracle’ that allows guessing whether a candidate value $\tilde{\Delta}$ equals the input of P_S or not by just computing one hash value. For example, let α, β be indices where $s_\alpha = s_\beta = 0$. Given $t_\alpha^0, t_\beta^0, u_\alpha$ and the hash values sent by P_S , P_R can compute $\tilde{t}_\alpha^1 := u_\alpha - t_\alpha^0 - \tilde{\Delta}$, and then test whether $h_{\alpha,\beta}^{1,0} = H(\tilde{t}_\alpha^1 - t_\beta^0)$. This only holds if $\Delta = \tilde{\Delta}$, so allows testing any candidate input $\tilde{\Delta}$. Including the extra randomness ρ prevents this attack by ensuring that $\Delta' = (\Delta \parallel \rho)$ always has at least k bits of min-entropy (as long as $|\mathbb{G}|^{k'} \geq k$), so P_R can only guess Δ' with negligible probability⁵.

Note that this step was missing in the published versions of [ALSZ15, ALSZ17a], which leads to an attack on their actively secure OT extension protocol. This has now been fixed in a revised version [ALSZ17b].

To formally prove security against a corrupted P_R , we construct a simulator \mathcal{S}_R , who interacts with $\mathcal{F}_{\Delta\text{-ROT}}$ whilst simulating the communication from the honest P_S and the \mathcal{F}_{OT} functionality to the adversary, \mathcal{A} . \mathcal{S}_R is described below.

1. In the **Initialize** phase, \mathcal{S}_R receives the inputs $\{s_i\}_{i \in [k]}$ from \mathcal{A} to $\mathcal{F}_{\text{OT}}^{k,k}$, then samples random strings $k_i^{s_i} \leftarrow \{0, 1\}^k$ and sends these to \mathcal{A} .
2. Whenever the Δ -OTs phase is called, \mathcal{S}_R starts by sampling $u_i \leftarrow \mathbb{G}^{n+k'}$, for $i \in [k]$, and sends these to \mathcal{A} .
3. In the consistency check, \mathcal{S}_R computes and sends the hash values $h_{\alpha,\beta}^{s_\alpha, s_\beta} = H(t_\alpha^{s_\alpha} - t_\beta^{s_\beta})$ and $h_{\alpha,\beta}^{\overline{s_\alpha}, \overline{s_\beta}} = H(u_\alpha - u_\beta - t_\alpha^{s_\alpha} + t_\beta^{s_\beta})$. The other two values $h_{\alpha,\beta}^{s_\alpha, \overline{s_\beta}}, h_{\alpha,\beta}^{\overline{s_\alpha}, s_\beta}$ are sampled uniformly at random.
4. \mathcal{S}_R then sends $\{s_i\}_i$ to $\mathcal{F}_{\Delta\text{-ROT}}$, and computes the values $\{a_i\}_i$ as an honest P_R would in the protocol. \mathcal{S}_R then sends $\{a_i\}_i$ to $\mathcal{F}_{\Delta\text{-ROT}}$ and increments the counter c .
5. Whenever the **Chosen OTs** phase is called, \mathcal{S}_R calls $\mathcal{F}_{\Delta\text{-ROT}}$ with input (chosen, B) , and receives $\{v_i\}_{i=1}^k$. \mathcal{S}_R computes $d_i^{s_i} = F(k_i^{s_i}, c) \oplus v_i$, samples a random string $d_i^{\overline{s_i}}$, then sends d_i^0, d_i^1 to \mathcal{A} and increments c .

Lemma 5. *If H is a (non-programmable, non-observable) random oracle and F is a secure PRF, then for every adversary \mathcal{A} who corrupts P_R , and for every environment \mathcal{Z} , it holds that*

$$\text{IDEAL}_{\mathcal{F}_{\Delta\text{-ROT}}, \mathcal{S}_R, \mathcal{Z}} \stackrel{c}{\approx} \text{HYBRID}_{\Pi_{\Delta\text{-ROT}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{OT}}}$$

⁵ This modification is not strictly needed for our application to the protocol in Sect. 3, because P_S 's input to $\mathcal{F}_{\Delta\text{-ROT}}$ is always uniformly random and cannot be guessed. However, making this change allows for a simpler, more modular exposition and the functionality may be useful in other applications.

Proof. We consider a sequence of hybrid distributions, going from the ideal execution to the real execution, defined as follows. The first hybrid H_0 is identical to the ideal execution with \mathcal{S}_R and $\mathcal{F}_{\Delta\text{-ROT}}$.

Hybrid H_1 : This is identical to H_0 , except that both sets of keys k_i^0, k_i^1 are sampled by \mathcal{S}_R , instead of just $k_i^{s_i}$. We also modify the **Chosen OTs** phase so that both values d_i^0, d_i^1 are computed according to the protocol, using the PRF keys and the real inputs of the honest P_S .

Hybrid H_2 : Here we modify H_1 further, so that the u_i values in the $\Delta\text{-OTs}$ stage are also computed according to the real protocol, using P_S 's real input Δ and a random value ρ . These u_i values are then used by \mathcal{S}_R to compute the a_i 's which are sent to $\mathcal{F}_{\Delta\text{-ROT}}$.

Hybrid H_3 : This is the same as H_2 , except the two hash values $h_{\alpha,\beta}^{s_\alpha, \overline{s_\beta}}, h_{\alpha,\beta}^{\overline{s_\alpha}, s_\beta}$ are computed as in the protocol, instead of with random strings.

It is easy to see the view of \mathcal{Z} in H_3 is identical to the real execution, since all messages are computed as an honest P_S would using the real inputs, and the outputs computed by $\mathcal{F}_{\Delta\text{-ROT}}$ are the same as in the protocol.

Hybrids H_0 and H_1 are computationally indistinguishable because the keys $k_i^{s_i}$ are unknown to \mathcal{Z} , which means the values $d_i^{s_i}$ are indistinguishable from the previously uniform values, by a standard reduction to the PRF security. Similarly, H_1 and H_2 are computationally indistinguishable because $t_i^{s_i}$ is pseudorandom based on the PRF, so masks P_S 's input in u_i .

Regarding H_2 , and H_3 , notice that the two relevant hash values in H_3 are equal to

$$\begin{aligned} h_{\alpha,\beta}^{s_\alpha, \overline{s_\beta}} &= H(t_\alpha^{s_\alpha} - t_\beta^{\overline{s_\beta}}) = H(t_\alpha^{s_\alpha} - u_\beta + t_\beta^{s_\beta} + \Delta') \\ h_{\alpha,\beta}^{\overline{s_\alpha}, s_\beta} &= H(t_\alpha^{\overline{s_\alpha}} - t_\beta^{s_\beta}) = H(u_\alpha - t_\alpha^{s_\alpha} - \Delta' - t_\beta^{s_\beta}) \end{aligned}$$

Looking at the values on the right-hand side, P_R knows everything in both sets of inputs to H except for $\Delta' = (\Delta \parallel \rho)$.

The only way \mathcal{Z} can distinguish between H_2 and H_3 is by querying H on one of the two inputs above, which occurs with probability at most $q \cdot |G|^{-k'} \leq q \cdot 2^{-k}$, where q is the number of random oracle queries, since ρ is uniformly random in $\mathbb{G}^{k'}$ and unknown to \mathcal{Z} . This completes the proof. \square

Corrupt P_S . When P_S is corrupt, the main challenge is to analyse soundness of the consistency check, similarly to [ALSZ15] (with a corrupt receiver in that protocol). Most of the analysis turns out to be identical, so we focus on the differences and state the main properties that we need from that work to show that our protocol securely realises $\mathcal{F}_{\Delta\text{-ROT}}$. For the proof to go through here we need to assume that the statistical security parameter λ is no more than $k/2$, but can always increase k to ensure this holds.

Note that the main way a corrupt P_S may cheat in the protocol is by using different Δ' values when sending the u_i values. To account for this, for each $i \in [k]$ we define $\Delta_i = u_i - t_i^0 - t_i^1$; if P_S behaves honestly then we have $\Delta_1 = \dots = \Delta_k = \Delta'$, otherwise they may be different.

The following lemma is analogous to [ALSZ15, Lemma 3.1], except we work over \mathbb{G} instead of bit strings, and implies that the rest of the analysis of the consistency check from that work also applies in our case. Using the terminology of Asharov et al., if the consistency check passes for some set of messages $\mathcal{T} = \{\{k_i^0, k_i^1, u_i\}_i, \{H_{\alpha,\beta}\}_{\alpha,\beta}\}$ and some secret \mathbf{s} , we say that \mathcal{T} is *consistent* with \mathbf{s} . If the check fails then it is *inconsistent*.

Lemma 6. *Let $\mathcal{T}_{\alpha,\beta} = \{k_\alpha^0, k_\alpha^1, k_\beta^0, k_\beta^1, u_\alpha, u_\beta, H_{\alpha,\beta}\}$ and suppose that H is a collision-resistant hash function. Then, except with negligible probability:*

1. *If $\Delta_\alpha \neq \Delta_\beta$ and $\mathcal{T}_{\alpha,\beta}$ is consistent with (s_α, s_β) then $\mathcal{T}_{\alpha,\beta}$ is inconsistent with $(\overline{s_\alpha}, \overline{s_\beta})$.*
2. *If $\Delta_\alpha = \Delta_\beta$ and $\mathcal{T}_{\alpha,\beta}$ is consistent with (s_α, s_β) then $\mathcal{T}_{\alpha,\beta}$ is also consistent with $(\overline{s_\alpha}, \overline{s_\beta})$.*

Proof. For the first claim, suppose that $\Delta_\alpha \neq \Delta_\beta$, and $\mathcal{T}_{\alpha,\beta}$ is consistent with both (s_α, s_β) and $(\overline{s_\alpha}, \overline{s_\beta})$. Then from the consistency with (s_α, s_β) we have

$$h_{\alpha,\beta}^{s_\alpha, s_\beta} = H(t_\alpha^{s_\alpha} - t_\beta^{s_\beta}), \quad h_{\alpha,\beta}^{\overline{s_\alpha}, \overline{s_\beta}} = H(u_\alpha - u_\beta - t_\alpha^{s_\alpha} + t_\beta^{s_\beta})$$

On the other hand, consistency with $(\overline{s_\alpha}, \overline{s_\beta})$ implies

$$h_{\alpha,\beta}^{\overline{s_\alpha}, \overline{s_\beta}} = H(t_\alpha^{\overline{s_\alpha}} - t_\beta^{\overline{s_\beta}}), \quad h_{\alpha,\beta}^{s_\alpha, s_\beta} = H(u_\alpha - u_\beta - t_\alpha^{\overline{s_\alpha}} + t_\beta^{\overline{s_\beta}})$$

By the collision resistance of H , except with negligible probability it must then hold that

$$\begin{aligned} t_\alpha^{s_\alpha} - t_\beta^{s_\beta} &= u_\alpha - u_\beta - t_\alpha^{\overline{s_\alpha}} + t_\beta^{\overline{s_\beta}} \\ &= t_\alpha^{s_\alpha} - t_\beta^{s_\beta} + (\Delta_\alpha - \Delta_\beta) \end{aligned}$$

This means $\Delta_\alpha = \Delta_\beta$, which is a contradiction.

For the second claim, if $\Delta_\alpha = \Delta_\beta$ then $u_\alpha - u_\beta = t_\alpha^0 + t_\alpha^1 - t_\beta^0 - t_\beta^1$, and it can be seen from the above equations that the checks for (s_α, s_β) and $(\overline{s_\alpha}, \overline{s_\beta})$ are equivalent. \square

For the case of a corrupted P_S , we construct a simulator \mathcal{S} , who interacts with \mathcal{A} and plays the role of the honest P_R and the \mathcal{F}_{OT} functionality. \mathcal{S}_S is described below.

1. \mathcal{S}_S receives all the keys k_i^0, k_i^1 as inputs to \mathcal{F}_{OT} , and then the messages u_i .
2. Using these it computes t_i^0, t_i^1 as in the protocol, and $\Delta_i = u_i - t_i^0 - t_i^1$. If P_S is honest then $\Delta_1 = \dots = \Delta_k$.
3. \mathcal{S}_S defines Δ' to be the most common of the Δ_i 's, and sends the first n components of this as P_S 's input to $\mathcal{F}_{\Delta\text{-ROT}}$.

4. \mathcal{S}_S then receives the sets of 4 hash values $H_{\alpha,\beta} = \{h_{\alpha,\beta}^{0,0}, h_{\alpha,\beta}^{0,1}, h_{\alpha,\beta}^{1,0}, h_{\alpha,\beta}^{1,1}\}$, for each $\alpha, \beta \in [k]$, as part of the consistency check.
5. \mathcal{S}_S then uses the transcript of \mathcal{A} to define a set of constraints on the secret \mathbf{s} that must be satisfied for the consistency check to pass, by running Algorithm 1 from [ALSZ15]. Note that each of the constraints produced by this algorithm either fixes individual bits of \mathbf{s} , or the XOR of two bits of \mathbf{s} , so from this we can efficiently define a set $S(\mathcal{T}) \subset \{0, 1\}^k$ which describes the set of all \mathbf{s} that are consistent with the messages from \mathcal{A} .
6. \mathcal{S}_S queries $\mathcal{F}_{\Delta\text{-ROT}}$ with $(\text{guess}, S(\mathcal{T}))$. If the query is successful, \mathcal{S}_S continues as if the consistency check passed, otherwise, \mathcal{S} aborts.
7. If the check passed, \mathcal{S}_S defines values b'_i , for $i \in [k]$ as specified below. These are sent to $\mathcal{F}_{\Delta\text{-ROT}}$.
8. Whenever the **Chosen OTs** phase is run, \mathcal{S}_S uses its knowledge of the keys to extract P_S 's inputs (u_i^0, u_i^1) , and sends these to $\mathcal{F}_{\Delta\text{-ROT}}$.

The key part of the simulation is step 5, which uses the hash values sent in the consistency check to define the exact bits (or XOR of bits) of the honest P_R 's secret \mathbf{s} which the corrupt P_S tried to guess from cheating. Note that \mathcal{S}_S does not define its own secret \mathbf{s} , as this is already sampled internally in the functionality $\mathcal{F}_{\Delta\text{-ROT}}$. Therefore, \mathcal{S}_S sends a description of all the possible consistent values of \mathbf{s} to the (guess) command of $\mathcal{F}_{\Delta\text{-ROT}}$ to see if the cheating attempt was successful or not.

Lemma 7. *If $\lambda \leq k/2$ and H is collision-resistant then for every adversary \mathcal{A} who corrupts P_S , and for every environment \mathcal{Z} , it holds that*

$$\text{IDEAL}_{\mathcal{F}_{\Delta\text{-ROT}}, \mathcal{S}_S, \mathcal{Z}} \stackrel{c}{\approx} \text{HYBRID}_{\Pi_{\Delta\text{-ROT}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{OT}}}$$

Proof (sketch): Define the transcript of the simulation up until the consistency check by $\mathcal{T} = \{\{k_i^0, k_i^1, u_i\}_i, \{H_{\alpha,\beta}\}_{\alpha,\beta}\}$, and define $\text{consistent}(\mathcal{T}, \mathbf{s})$ to be 1 if the consistency check would pass if \mathbf{s} is the secret of P_R , and 0 otherwise. From Algorithm 1 in step 5, recall that the we defined set of all possible secrets $\mathbf{s} \in \{0, 1\}^k$ of an honest P_R for which the check would pass to be $S(\mathcal{T}) = \{\mathbf{s} \in \{0, 1\}^k : \text{consistent}(\mathcal{T}, \mathbf{s}) = 1\}$, where \mathcal{T} is the transcript produced by \mathcal{A} . Note that from the definition of $S(\mathcal{T})$, the probability that the consistency check passes is $|S(\mathcal{T})|/2^k$ in both the real and ideal executions. To complete the proof we just need to show how to extract the correct values b'_i defining P_S 's output.

Below we give the key results from [ALSZ15] needed to analyse the consistency check.

Lemma 8. *For a given transcript \mathcal{T} , let U be the largest set of indices such that for all $i, j \in U$, $\Delta_i = \Delta_j$, and let $B = [k] \setminus U$ be the complementary set. We have:*

1. *If $B > \lambda$ then the probability of passing the consistency check is $\leq 2^{-\lambda}$.*
2. *If the consistency check passes, then for all $\mathbf{s}' \in S(\mathcal{T})$, it holds that either the bits $\{\mathbf{s}'_i\}_{i \in B}$ are fixed, or the bits $\{\mathbf{s}'_i\}_{i \in U}$ are fixed.*

Proof. The first claim follows from Claim B.3 of [ALSZ15] and Lemma 6. The second can be seen from inspection of the proof of Claim B.4 in the same work. \square

From the first item, we conclude that $|B| \leq \lambda$, except with negligible probability. We claim that this means we first be in the first case of item 2 in the lemma. If the bits $\{s'\}_{i \in U}$ were fixed then the adversary must have guessed $|U|$ bits of the secret to pass the check, but since $|U| \geq k - \lambda \geq \lambda$, this can only happen with probability $\leq 2^{-\lambda}$.

This implies that (except with negligible probability) the bits of s sampled by $\mathcal{F}_{\Delta\text{-ROT}}$ are fixed at the positions $i \in B$, so \mathcal{S} can define a value $b'_i = t_i^0 + s_i \cdot (\Delta_i - \Delta')$ from the fact that s_i is fixed, for all $i \in B$. We then have $b'_i = a_i - s_i \cdot \Delta$, so this defines the correct output that \mathcal{S} sends to $\mathcal{F}_{\Delta\text{-ROT}}$ in step 7. Note that for all $i \in U$ we have $\Delta_i = \Delta'$, so we can just let $b'_i = t_i^0$. These outputs are then identically distributed to the outputs of P_S in the real protocol, so (accounting for the negligible failure events) the simulation is statistically close. \square

Acknowledgements. I would like to thank Claudio Orlandi for the observation that an XOR-homomorphic PRG would imply non-interactive OT extension, which inspired this work. I am also grateful to the PKC 2018 reviewers for helpful comments.

This work has been supported by the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731583 (SODA), and the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC).

A Conversion to 1-out-of-2 OTs

The main protocol in Sect. 3 produces a batch of random 1-out-of- p_i OTs, for multiple small primes p_i . If an application requires 1-out-of-2 OTs (as is common) then we can convert each of the 1-out-of- p_i OTs to a 1-out-of-2 OT at a cost of $O(\log p_i)$ bits of communication using standard techniques, with active security⁶.

Protocol Π_{Conv}

Sender input: strings s_0, \dots, s_{N-1}
Receiver input: choice $b \in \{0, 1\}$

1. P_S obtains random strings r_0, \dots, r_{N-1} from $\mathcal{F}_{p_i\text{-ROT}}$
2. P_R obtains a random choice $c \in \{0, \dots, N - 1\}$, and the string r_c .
3. P_R sends $d = c - b \pmod N$ to P_S
4. P_S sends $e_0 = s_0 \oplus r_d$ and $e_1 = s_1 \oplus r_{d+1 \pmod N}$
5. P_R recovers s_b by computing $e_b \oplus r_c$

Fig. 7. Conversion from random 1-out-of- N OT to chosen 1-out-of-2 OT

⁶ It is possible to convert a 1-out-of- N OT into $\log_2 N$ 1-out-of-2 OTs [NP99, KK13], but this technique would not improve the asymptotic communication cost in our case, and is only passively secure.

In the protocol, shown in Fig. 7, the receiver first converts the random choice c from the 1-out-of- N OT into its chosen input bit b by sending the difference $d = c - b \pmod N$. The sender, who initially has random strings r_0, \dots, r_{N-1} , then uses r_d and $r_{d+1 \pmod N}$ to one-time-pad encrypt its two input messages. The receiver can recover exactly one of these, corresponding to $r_c = r_{d+b}$.

Security of the protocol is straightforward. The only concern is that if the receiver is corrupt, P_R might choose an inconsistent value $b \in \{2, \dots, p-1\}$ instead of a bit, so learns a random string instead of one of the sender's two inputs. However, the fact that a corrupt P_R may not learn a valid output does not pose a problem, since in this case, in the security proof the simulator can just send an arbitrary choice bit to the \mathcal{F}_{OT} functionality and simulate the e_0, e_1 messages from the sender with random strings.

References

- [AKPW13] Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 57–74. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_4
- [ALSZ13] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 535–548. ACM Press, November 2013
- [ALSZ15] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_26
- [ALSZ17a] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions. *J. Cryptol.* **30**(3), 805–858 (2017)
- [ALSZ17b] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. *Cryptology ePrint Archive: Report 2015/061*. Version 20171121:125742 (2017). <https://eprint.iacr.org/2015/061>
- [Bea96] Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: 28th ACM STOC, pp. 479–488. ACM Press, May 1996
- [BFP+15] Banerjee, A., Fuchsbauer, G., Peikert, C., Pietrzak, K., Stevens, S.: Key-homomorphic constrained pseudorandom functions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 31–60. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_2
- [BGG+14] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30

- [BGI16] Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 509–539. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_19
- [BGI17] 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. LNCS, vol. 10211, pp. 163–193. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_6
- [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
- [BMR16] Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for boolean and arithmetic circuits. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 565–577. ACM Press, October 2016
- [BP14] Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 353–370. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_20
- [BPR12] 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
- [BV15] Brakerski, Z., Vaikuntanathan, V.: Constrained key-homomorphic PRFs from standard lattice assumptions - or: how to secretly embed a circuit in your PRF. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 1–30. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_1
- [Can01] Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
- [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
- [GMW86] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS, pp. 174–187. IEEE Computer Society Press, October 1986
- [GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC, pp. 218–229. ACM Press, May 1987
- [Gol04] Goldreich, O.: The Foundations of Cryptography - Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
- [GVW13] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 545–554. ACM Press, June 2013

- [HIJ+16] Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Sudan, M. (ed.) ITCS 2016, pp. 157–168. ACM, January 2016
- [HIKN08] Harnik, D., Ishai, Y., Kushilevitz, E., Nielsen, J.B.: OT-combiners via secure computation. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 393–411. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_22
- [HW15] Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) ITCS 2015, pp. 163–172. ACM, January 2015
- [IKNP03] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
- [IKOS07] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC, pp. 21–30. ACM Press, June 2007
- [IKOS08] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 433–442. ACM Press, May 2008
- [IKOS09] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Extracting correlations. In: 50th FOCS, pp. 261–270. IEEE Computer Society Press, October 2009
- [IPS08] Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
- [IR89] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, pp. 44–61. ACM Press, May 1989
- [KK13] Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_4
- [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, pp. 818–829. ACM Press, October 2016
- [KOS15] Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 724–741. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_35
- [NNOB12] Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_40
- [NP99] Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: 31st ACM STOC, pp. 245–254. ACM Press, May 1999
- [OOS17] Orrù, M., Orsini, E., Scholl, P.: Actively secure 1-out-of- N OT extension with application to private set intersection. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 381–396. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_22

- [PSS17] Patra, A., Sarkar, P., Suresh, A.: Fast actively secure OT extension for short secrets. In: NDSS 2017. Internet Society (2017)
- [PSZ18] Pinkas, B., Schneider, T., Zohner, M.: Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.* **21**(2), 7:1–7:35 (2018)
- [Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167. IEEE Computer Society Press, October 1986

Multiparty Computation



Committed MPC

Maliciously Secure Multiparty Computation from Homomorphic Commitments

Tore K. Frederiksen¹✉, Benny Pinkas², and Avishay Yanai² 

¹ Security Lab, Alexandra Institute, Aarhus, Denmark
tore.frederiksen@alexandra.dk

² Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
benny@pinkas.net, ay.yanay@gmail.com

Abstract. We present a new multiparty computation protocol secure against a static and malicious dishonest majority. Unlike most previous protocols that were based on working on MAC-ed secret shares, our approach is based on computations on homomorphic commitments to secret shares. Specifically we show how to realize MPC using any additively-homomorphic commitment scheme, even if such a scheme is an interactive two-party protocol.

Our new approach enables us to do arithmetic computation over arbitrary finite fields. In addition, since our protocol computes over committed values, it can be readily composed within larger protocols, and can also be used for efficiently implementing committing OT or committed OT. This is done in two steps, each of independent interest:

1. Black-box extension of any (possibly interactive) two-party additively homomorphic commitment scheme to an additively homomorphic multiparty commitment scheme, only using coin-tossing and a “weak” equality evaluation functionality.
2. Realizing multiplication of multiparty commitments based on a lightweight preprocessing approach.

Finally we show how to use the fully homomorphic commitments to compute any functionality securely in the presence of a malicious adversary corrupting any number of parties.

1 Introduction

Secure computation (MPC) is the area of cryptography concerned with mutually distrusting parties who wish to compute some function f on private input from each of the parties, yielding some private output to each of the parties. If we consider p parties, P_1, \dots, P_p where party P_i has input x_i the parties then wish to

T. K. Frederiksen—Majority of work done while at Bar-Ilan University, Israel.

All authors were supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Ministers Office. Tore has also received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 731583.

© International Association for Cryptologic Research 2018

M. Abdalla and R. Dahab (Eds.): PKC 2018, LNCS 10769, pp. 587–619, 2018.

https://doi.org/10.1007/978-3-319-76578-5_20

learn their respective output y_i . We can thus describe the function to compute as $f(x_1, x_2, \dots, x_p) = (y_1, y_2, \dots, y_p)$. It was shown in the 80's how to realize this, even against a malicious adversary taking control over a majority of the parties [24]. With feasibility in place, much research has been carried out trying to make MPC as efficient as possible. One specific approach to efficient MPC, which has gained a lot of traction is based on secret sharing [3, 5, 24]: Each party secretly shares his or her input with the other parties. The parties then parse f as an arithmetic circuit, consisting of multiplication and addition gates. In a collaborative manner, based on the shares, they then compute the circuit, to achieve shares of the output which they can then open.

Out Contributions. Using the secret sharing approach opens up the possibility of malicious parties using “inconsistent” shares in the collaborative computation. To combat this, most protocols add a MAC on the true value shared between the parties. If someone cheats it is then possible to detect this when verifying the MAC [14, 16, 33].

In this paper we take a different approach to ensure correctness: We have each party commit to its shares towards the other parties using an additively homomorphic commitment. We then have the collaborative computation take place on the commitments instead of the pure shares. Thus, if some party tries to change its share during the protocol then the other parties will notice when the commitments are opened in the end (as the opening will be invalid).

By taking this path, we can present the following contributions:

1. An efficient and black-box reduction from random multiparty homomorphic commitments, to two-party additively homomorphic commitments.
2. Using these multiparty commitments we present a new secret-sharing based MPC protocol with security against a majority of malicious adversaries. Leveraging the commitments, our approach does not use any MAC scheme and does not rely on a random oracle or any specific number theoretic assumptions.
3. The new protocol has several advantages over previous protocols in the same model. In particular our protocol works over fields of arbitrary characteristic, independent of the security parameter. Furthermore, since our protocol computes over committed values it can easily be composed inside larger protocols. For example, it can be used for computing committed OT in a very natural and efficient way.
4. We suggest an efficient realization of our protocol, which only relies on a PRG, coin-tossing and OT¹. We give a detailed comparison of our scheme with other dishonest majority, secret-sharing based MPC schemes, showing that the efficiency of our scheme is comparable, and in several cases preferable, over state-of-the-art.

¹ OT can be efficiently realized using an OT extension, without the usage of a random oracle, but rather a type of correlation robustness, as described in [2].

High Level Idea. We depart from any (possibly interactive) two-party additively homomorphic commitment scheme. To achieve the most efficient result, without relying on a random oracle or specific number theoretic assumptions, we consider the scheme of [18], since has been shown to be highly efficient in practice [35,36]. This scheme, along with others [9–11] works on commitments to vectors of m elements over some field \mathbb{F} . For this reason we also present our results in this setting. Thus any of these schemes could be used.

The first part of our protocol constructs a large batch of commitments to random values. The actual value in such a commitment is unknown to any party, instead, each party holds an additive share of it. This is done by having each party pick a random message and commit to it towards every other party, using the two-party additively homomorphic commitment scheme. The resulted multiparty commitment is the sum of all the messages the parties committed to, which is uniformly random if there is at least one honest party. We must ensure that a party commits to the same message towards all other parties, to this end the parties agree on a few (random) linear combinations over the commitments, which are then opened and being checked.

Based on these random additively shared commitments, the parties execute a preprocessing stage to construct random multiplication triples. This is done in a manner similar to MASCOT [29], yet a bit different, since our scheme supports computation over arbitrary small fields and MASCOT requires a field of size exponential in the security parameter. More specifically the Gilboa protocol [23] for multiplication of additively shared values is used to compute the product of two shares of the commitments between each pair of parties. However, this is not maliciously secure as the result might be incorrect and a few bits of information on the honest parties' shares might be leaked. To ensure correctness cut-and-choose and sacrificing steps are executed. First, a few triples are opened and checked for correctness. This ensures that not all triples are incorrectly constructed. Next, the remaining triples are mapped into buckets, where some triples are sacrificed to check correctness of another triple. At this point all the triples are correct except with negligible probability. Finally, since the above process grants the adversary the ability to leak some bits from the honest parties shares, the parties engage in a combining step, where triples are randomly “added” together to ensure that the result will contain at least one fully random triple.

As the underlying two-party commitments are for *vectors* of messages, our protocol immediately features single-instruction multiple-data (SIMD), which allows multiple simultaneously executions of the same computation (over different inputs). However, when performing only a single execution we would like to use only one element out of the vector and save the rest of the elements for a later use. We do so by preprocessing reorganization pairs, following the same approach presented in MiniMAC [12,15,16], which allows to perform a linear transformation over a committed vector.

With the preprocessing done, the online phase of our protocol proceeds like previous secret-sharing based MPC schemes such as [14,16,29]. That is, the

parties use their share of the random commitments to give input to the protocol. Addition is then carried out locally and the random multiplication triples are used to interactively realize multiplication gates.

Efficiency. In Table 1 we count the amount of OTs, two-party commitments and coin tossing operations required in the different commands of our protocol (specifically, in the **Rand**, **Input**, **ReOrg**, **Add** and **Mult** commands).

The complexities describe what is needed to construct a vector of m elements in the underlying field in the amortized sense. When using the commitment scheme of [18] it must hold that $m \geq s/\lceil \log_2(|\mathbb{F}|) \rceil$ where s is the statistical security parameter.

Table 1. Amortized complexity of each instruction of our protocol (Rand, Input, ReOrg, Add and Mult), when constructing a batch of 2^{20} multiplication triples, each with m independent components among p parties. The quadratic complexity of the number of two-party commitments reflects the fact that our protocol is constructed from any two-party commitment scheme in a black-box manner, and so each party independently commits to all other party for every share it possesses.

	Rand, Input	ReOrg	Add	Mult
OTs	0	0	0	$27m \log(\mathbb{F})p(p - 1)$
Two-party commitments	$p(p - 1)$	$3p(p - 1)$	0	$81p(p - 1)$
Random coins	$\log(\mathbb{F})$	$4 \log(\mathbb{F})$	0	$108 \log(\mathbb{F})$

1.1 Related Work

Comparison to SPDZ and TinyOT. In general having the parties commit to their shares allows us to construct a secret-sharing based MPC protocol ala SPDZ [14, 29], but without the need of shared and specific information theoretic MACs. This gives us several advantages over the SPDZ approach:

- We get a light preprocessing stage of multiplication triples as we can base this on commitments to random values, which are later adjusted to reflect a multiplication. Since the random values are additively homomorphic and committed, this limits the adversary’s possible attack vector. In particular we do not need an authentication step.
- Using the commitment scheme of [18] we get the possibility of committing to messages in any field \mathbb{F} among p parties, using communication of only $O(\log(|\mathbb{F}|) \cdot p^2)$ bits, amortized. This is also the case when \mathbb{F} is the binary field² or of different characteristic than 2. In comparison, SPDZ requires the underlying field to be of size $\Omega(2^s)$ where s is a statistical security parameter.
- The TinyOT protocol [7, 30, 33] on the other hand only works over $\text{GF}(2)$ and requires a MAC of $\tilde{O}(s)$ bits on each secret bit. Giving larger overhead than in SPDZ, MiniMAC and our protocol and limiting its use-case to evaluation of Boolean circuits.

² This requires a commitment to be to a vector of messages in \mathbb{F} .

Comparison to MiniMAC. The MiniMAC protocol [16] uses an error correcting code over a vector of data elements. It can be used for secure computation over small fields without adding long MACs to each data element. However, unfortunately the authors of [16] did not describe how to realize the preprocessing needed. Neither did the follow up works [12, 15]. The only efficient³ preprocessing protocol for MiniMAC that we know of is the one presented in [19] based on OT extension. However this protocols has it quirks:

- It only works over fields of characteristic 2.
- The ideal functionality described is different from the ones in [12, 15, 16]. Furthermore, it is non-standard in the sense that the corruption that an adversary can apply to the shares of honest parties can be based on the inputs of the honest parties.
- There is no proof that this ideal functionality works in the online phase of MiniMAC.
- There seems to be a bug in one of the steps of the preprocessing of multiplication triples. We discuss this in further detail in the full version [20].

OT Extensions. All the recent realizations of the preprocessing phase on secret shared protocols such as SPDZ, MiniMAC and TinyOT are implemented using OT. The same goes for our protocol. Not too long ago this would have not been a practically efficient choice since OT generally requires public key operations. However, the seminal work of Beaver [4] showed that it was possible to extend a few OTs, using only symmetric cryptography, to achieve a practically unbounded amount of OTs. Unfortunately Beaver’s protocol was not practically efficient, but much research has been carried out since then [1, 2, 25, 28, 33], culminating with a maliciously secure OT extension where a one-out-of-two OT of 128 bit messages with $s = 64$ can be done, in the amortized sense, in $0.3 \mu\text{s}$ [28].

Commitment Extensions. Using additive homomorphic commitments for practical MPC is a path which would also not have been possible even just a few years ago. However, much study has undergone in the area of “commitment extension” in the recent years. All such constructions that we know of require a few OTs in a preprocessing phase and then construction and opening of commitments can be done using cheap symmetric or information theoretic primitives. The work on such extensions started in [22] and independently in [11]. A series of follow-up work [6, 9, 10, 18, 35] presented several improvements, both asymptotically and practically. Of these works [35] is of particular interest since it presents an implementation (based on the scheme of [18]) and showed that committing and opening 128 bit messages with $s = 40$ can be done in less than $0.5 \mu\text{s}$ and $0.2 \mu\text{s}$ respectively, in the amortized sense for a batch of 500,000 commitments⁴.

³ I.e. one that does not use a generic MPC protocol to do the preprocessing.

⁴ Note that this specific implementation unfortunately uses a code which does not have the properties our scheme require. Specifically its product-code has too low minimum distance.

It should be noted that Damgård *et al.* [11] also achieved both additively and multiplicative homomorphic commitments. They use this to get an MPC protocol cast in the client/server setting. We take some inspiration from their work, but note that their setting and protocols are quite different from ours in that they use verifiable secret sharing to achieve the multiplicative property and so their scheme is based on threshold security, meaning they get security against a constant fraction of servers in a client/server protocol.

Relation to [13]. The protocol by Damgård and Orlandi also considers a maliciously secure secret-sharing based MPC in the dishonest majority setting. Like us, their protocol is based on additively homomorphic commitments where each party is committed to its share to thwart malicious behavior. However, unlike ours, their protocol only works over large arithmetic fields and uses a very different approach. Specifically they use the cut-and-choose paradigm along with packed secret sharing in order to construct multiplication triples. Furthermore, to get random commitments in the multiparty setting, they require usage of public-key encryption for each commitment. Thus, the amount of public-key operations they require is linear in the amount of multiplication gates in the circuit to compute. In our protocol it is possible to limit the amount of public-key operations to be asymptotic in the security parameter, as we only require public-key primitives to bootstrap the OT extension.

Other Approaches to MPC. Other approaches to maliciously secure MPC in the dishonest majority setting exist. For example Yao’s garbled circuit [31,32,37], where the parties first construct an encrypted Boolean circuit and then evaluate it locally. Another approach is “MPC-in-the-head” [26,27] which efficiently combines any protocol in the malicious honest majority settings and any protocol in the semi-honest dishonest majority settings into a protocol secure in the malicious dishonest majority settings.

2 Preliminaries

Parameters and Notation. Throughout the paper we use “negligible probability in s ” to refer to a probability $o(1/\text{poly}(s))$ where $\text{poly}(s)$ indicates some polynomial in $s \in \mathbb{N}$. Similarly we use “overwhelming probability in s ” to denote a probability $1 - o(1/\text{poly}(s))$, where s is the statistical security parameter.

There are $p \in \mathbb{N}$ parties P_1, \dots, P_p participating in the protocol. The notation $[k]$ refers to the set $\{1, \dots, k\}$. We let vector variables be expressed with **bold** face. We use square brackets to select a specific element of a vector, that is, $\mathbf{x}[\ell] \in \mathbb{F}$ is the ℓ ’th element of the vector $\mathbf{x} \in \mathbb{F}^m$ for some $m \geq \ell$. We assume that vectors are column vectors and use $\|$ to denote concatenation of rows, that is, $\mathbf{x}\|\mathbf{y}$ with $\mathbf{x}, \mathbf{y} \in \mathbb{F}^m$ is a $m \times 2$ matrix. We use $*$: $\mathbb{F}^m \times \mathbb{F}^m \rightarrow \mathbb{F}^m$ to denote component-wise multiplication and \cdot : $\mathbb{F} \times \mathbb{F}^m \rightarrow \mathbb{F}^m$ to denote a scalar multiplication. We will sometimes abuse notation slightly and consider \mathbb{F} as a set of elements and thus use $\mathbb{F} \setminus \{0\}$ to denote the elements of \mathbb{F} , excluding the additive neutral element 0.

If S is a set we assume that there exists an arbitrary, but globally known deterministic ordering in such a set and let $S[i] = S_i$ denote the i th element under such an ordering. In general we always assume that sets are stored as a list under such an ordering. When needed we use (a, b, \dots) to denote a list of elements in a specific order. Letting A and B be two sets s.t. $|A| = |B|$ we then abuse notation by letting $\{(a, b)\} \in (A, B)$ denote $\{(A[i], B[i])\}_{i \in [|A|]}$. I.e. a and b denote the i 'th element in A , respectively B .

All proof and descriptions of protocols are done using the Universally Composable framework [8].

Ideal Functionalities. We list the ideal UC-functionalities we need for our protocol. Note that we use the standard functionalities for Coin Tossing, Secure Equality Check, Oblivious Transfer and Multiparty Computation.

We need a coin-tossing functionality that allows all parties to agree on uniformly random elements in a field. For this purpose we describe a general, maliciously secure coin-tossing functionality in Fig. 1.

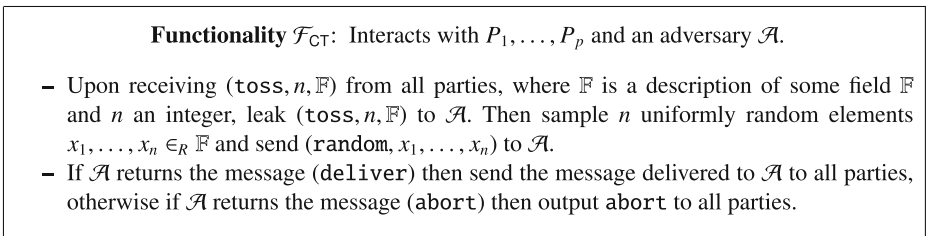


Fig. 1. Ideal functionality \mathcal{F}_{CT}

Furthermore we need to be able to securely evaluate equality of values. This functionality is described in Fig. 2. Notice that this functionality allows the adversary to learn the honest parties' inputs *after* it supplies its own. Furthermore, we allow the adversary to learn the result of the equality check before any honest parties, which gives him the chance to abort the protocol. Thus this function should only be used on data that is not private. The functionality can for example be implemented using a commitment scheme where each party commits to its input towards every other party. Once all parties have committed, the parties open the commitments and each party locally evaluates if everything is equal.

We also require a standard 1-out-of-2 functionality denoted by \mathcal{F}_{OT} as described in Fig. 3.

Finally, a fully fledged MPC functionality, very similar to the one described in previous works such as SPDZ and MiniMAC, is described in Fig. 4. Note that the functionality maintains the dictionary id that maps indices to values stored by the functionality. The expression $\text{id}[k] = \perp$ means that no value is stored by the functionality at index k in that dictionary. Also note that the functionality

Functionality \mathcal{F}_{EQ} : Interacts with P_1, \dots, P_p and an adversary \mathcal{A} . It proceeds as follows:

Equality: Upon receiving $(\text{equal}, i, \mathbf{x}^i)$ from party P_i for all $i \in [p]$ where $\mathbf{x}^i \in \mathbb{F}^m$ (if P_i is corrupted then \mathbf{x}^i is selected by \mathcal{A}), proceed as follows: If $\mathbf{x}^1 = \mathbf{x}^2 = \dots = \mathbf{x}^p$ then send $(\text{equal}, \text{accept})$ to \mathcal{A} , otherwise send $(\text{equal}, \mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^p, \text{reject})$ to \mathcal{A} . Proceed as follows:

- Upon receiving $(\text{equal}, i, \mathbf{x}^i)$ from party P_i for all $i \in [p]$ (if P_i is corrupted then \mathbf{x}^i is selected by \mathcal{A}) if $\mathbf{x}^1 = \mathbf{x}^2 = \dots = \mathbf{x}^p$ then send $(\text{equal}, \text{accept})$ to \mathcal{A} , otherwise send $(\text{equal}, \mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^p, \text{reject})$ to \mathcal{A} .
- If \mathcal{A} returns (deliver) and $\mathbf{x}^1 = \mathbf{x}^2 = \dots = \mathbf{x}^p$ then send the message $(\text{equal}, \text{accept})$ to all parties. If instead $\mathbf{x}^i \neq \mathbf{x}^j$ for some $i, j \in [p]$, then send $(\text{equal}, \mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^p, \text{reject})$ to all parties.
- If \mathcal{A} instead returns (abort) then output abort to all parties.

Fig. 2. Ideal functionality \mathcal{F}_{EQ}

Functionality \mathcal{F}_{OT} : Interacts with a sender P_i , a receiver P_j and an adversary \mathcal{A} and proceeds as follows:

Sender Input: Upon receiving $(\text{transfer}, x_0, x_1)$ from P_i where $x_0, x_1 \in \{0, 1\}^*$ leak (transfer) to \mathcal{A} .

Receiver Input: Upon receiving $(\text{receive}, b)$ from P_j where $b \in \{0, 1\}$ leak (receive) to \mathcal{A} . If a message of the form $(\text{transfer}, x_0, x_1)$ has been received from P_i then output $(\text{deliver}, x_b)$ to P_j and $(\text{deliver}, \perp)$ to P_i .

Fig. 3. Ideal functionality \mathcal{F}_{OT}

is described as operating over *vectors* from \mathbb{F}^m rather than over elements from \mathbb{F} . This is because our protocol allows up to m simultaneous secure computations of the same function (on different inputs) at the price of a single computation, thus, every operation (such as input, random, add, multiply) are done in a component wise manner to a vector from \mathbb{F}^m . As we describe later, it is indeed possible to perform a single secure computation when needed.

Dependencies between functionalities and protocols. We illustrate the dependencies between the ideal functionalities just presented and our protocols in Fig. 5. We see that \mathcal{F}_{CT} and \mathcal{F}_{EQ} , along with a two-party commitments scheme, $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$ (presented in the next section) are used to realize our multiparty commitment scheme in protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$. Functionalities \mathcal{F}_{CT} and \mathcal{F}_{EQ} are again used, along with $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ and \mathcal{F}_{OT} to realize the augmented homomorphic commitment scheme $\Pi_{\text{AHCOM-}\mathbb{F}^m}$. $\Pi_{\text{AHCOM-}\mathbb{F}^m}$ constructs all the preprocessed material, in particular multiplication triples, needed in order to realize the fully fledged MPC protocol $\Pi_{\text{MPC-}\mathbb{F}^m}$.

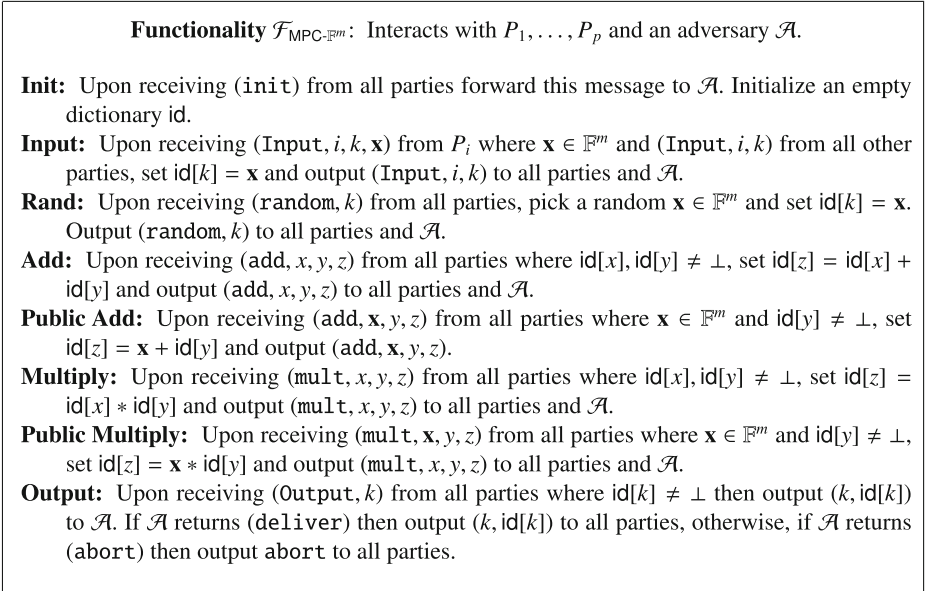


Fig. 4. Ideal functionality $\mathcal{F}_{\text{MPC-}\mathbb{F}^m}$

Arithmetic Oblivious Transfer. Generally speaking, arithmetic oblivious transfer allows two parties P_i and P_j to obtain an additive shares of the multiplication of two elements $x, y \in \mathbb{F}$, where P_i privately holds x and P_j privately holds y .

A protocol for achieving this in the semi-honest settings is presented in [23] and used in MASCO^T [29]. Let $\ell = \lceil \log \mathbb{F} \rceil$ be the number of bits required to represent elements from the field \mathbb{F} , then the protocol goes by having the parties run in ℓ (possibly parallel) rounds, each of which invokes an instance of the general oblivious transfer functionality (\mathcal{F}_{OT}). This is described by procedure ArithmeticOT in Fig. 6.

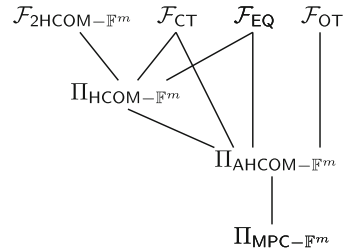


Fig. 5. Outline of functionalities and protocols.

The use of arithmetic OT to construct multiplication triples. In our protocol we use the above procedure to multiply two elements $\mathbf{x}, \mathbf{y} \in \mathbb{F}^m$ such that one party privately holds \mathbf{x} and the other party privately holds \mathbf{y} . Specifically, we can do this using m invocations of the ArithmeticOT procedure, thus, to multiply elements from \mathbb{F}^m we make a total of $m \log(\lceil |\mathbb{F}| \rceil)$ calls to the transfer command of the \mathcal{F}_{CT} functionality.

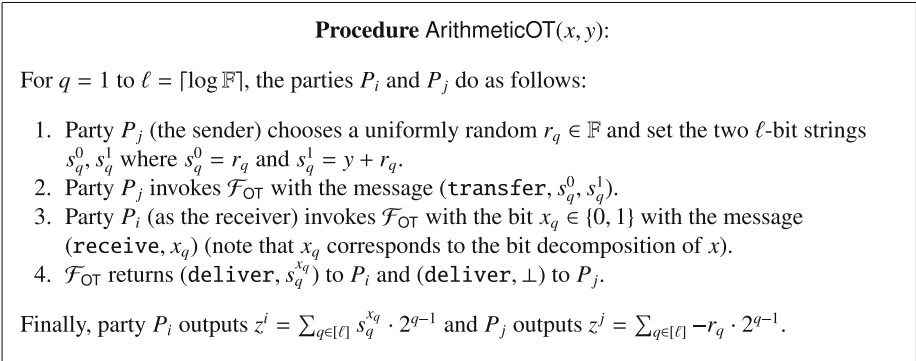


Fig. 6. Procedure ArithmeticOT

Even if using a maliciously secure OT functionality to realize this procedure, it still does not become maliciously secure. We discuss how to handle this in Sect. 4.2.

3 Homomorphic Commitments

In this section we present the functionalities for two-party and multiparty homomorphic commitment schemes, however, we present a realization only to the multiparty case since it uses a two-party homomorphic commitment scheme in a black-box manner and so it is not bound to any specific realization.

For completeness and concreteness of the efficiency analysis we do present a realization to the two-party homomorphic commitment scheme in the full version [20].

3.1 Two-Party Homomorphic Commitments

Functionality $\mathcal{F}_{\text{2HCOM-}\mathbb{F}^m}$ is held between two parties P_i and P_j , in which P_i commits to some value $\mathbf{x} \in \mathbb{F}^m$ toward party P_j , who eventually holds the commitment information, denoted $[\mathbf{x}]^{i,j}$. In addition, by committing to some value \mathbf{x} party P_i holds the opening information, denoted $\langle \mathbf{x} \rangle^{i,j}$, such that having P_i send $\langle \mathbf{x} \rangle^{i,j}$ to P_j is equivalent to issuing the command **Open** on \mathbf{x} by which P_j learns \mathbf{x} .

The functionality works in a batch manner, that is, P_i commits to γ (random) values at once using the **Commit** command. These γ random values are considered as “raw-commitments” since they have not been processes yet. The sender turns the commitment from “raw” to “actual” by issuing either **Input** or **Rand** commands on it: The **Input** command modifies the committed value to the sender’s choice and the **Rand** command keeps the same value of the commitment (which is random). In both cases the commitment is considered as a “actual”

and is not “raw” anymore. Actual commitments can then be combined using the **Linear Combination** command to construct a new actual-commitment.

To keep track of the commitments the functionality uses two dictionaries: **raw** and **actual**. Both map from identifiers to committed values such that the mapping returns \perp if no mapping exists for the identifier. We stress that a commitment is either raw or actual, but not both. That means that either **raw** or **actual**, or both return \perp for every identifier. To issue the **Commit** command, the committer is instructed to choose a set I of γ freshly new identifiers, this is simply a set of identifiers I such that for every $k \in I$ **raw** and **actual** return \perp . The functionality is formally described in Fig. 7.

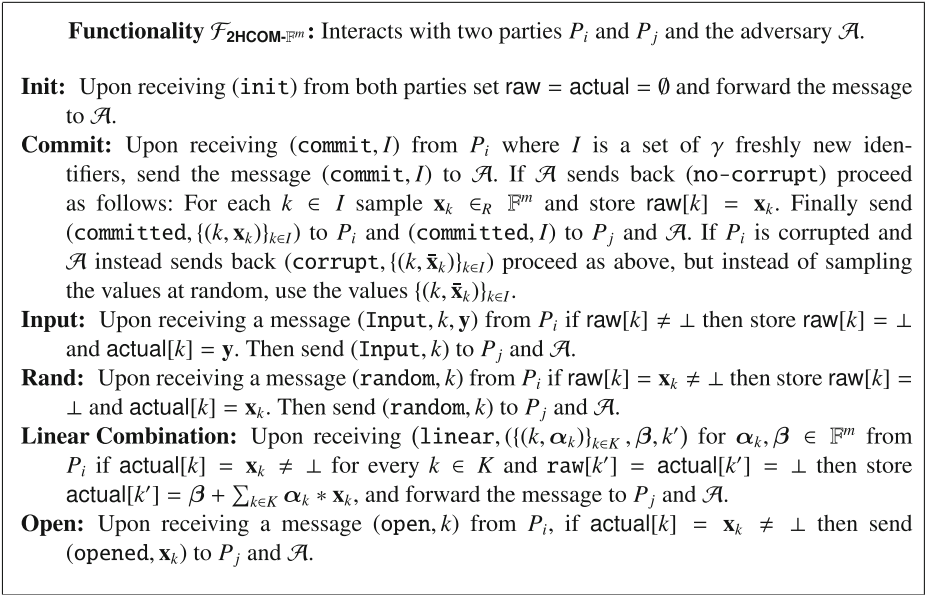


Fig. 7. Ideal functionality $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$

To simplify readability of our protocol we may use shorthands to the functionality’s commands invocations as follows: Let $[\mathbf{x}_k]^{i,j}$ and $[\mathbf{x}_{k'}]^{i,j}$ be two actual-commitments issued by party P_i toward party P_j (i.e. the committed values are stored in **actual**[k] and **actual**[k'] respectively). The **Linear Combination** command of Fig. 7 allows to compute the following operations which will be used in our protocol. The operations are defined over $[\mathbf{x}_k]^{i,j}$ and $[\mathbf{x}_{k'}]^{i,j}$ and result with the actual-commitment $[\mathbf{x}_{k''}]^{i,j}$:

– **Addition.** (Equivalent to the command (**linear**, $\{(k, \mathbf{1}), (k', \mathbf{1})\}, \mathbf{0}, k''$)).

$$[\mathbf{x}_k]^{i,j} + [\mathbf{x}_{k'}]^{i,j} = [\mathbf{x}_k + \mathbf{x}_{k'}]^{i,j} = [\mathbf{x}_{k''}]^{i,j} \quad \text{and} \quad \langle \mathbf{x}_k \rangle^{i,j} + \langle \mathbf{x}_{k'} \rangle^{i,j} = \langle \mathbf{x}_k + \mathbf{x}_{k'} \rangle^{i,j} = \langle \mathbf{x}_{k''} \rangle^{i,j}$$

- **Constant Addition.** (Equivalent to the command $(\mathbf{linear}, \{(k, \mathbf{1})\}, \beta, k'')$.)

$$\beta + [\mathbf{x}_k]^{i,j} = [\beta + \mathbf{x}_k]^{i,j} = [\mathbf{x}_{k''}]^{i,j} \quad \text{and} \quad \beta + \langle \mathbf{x}_k \rangle^{i,j} = \langle \beta + \mathbf{x}_k \rangle^{i,j} = \langle \mathbf{x}_{k''} \rangle^{i,j}$$
- **Constant Multiplication.** (Equivalent to the command $(\mathbf{linear}, \{(k, \alpha)\}, \mathbf{0}, k'')$.)

$$\alpha * [\mathbf{x}_k]^{i,j} = [\alpha * \mathbf{x}_k]^{i,j} = [\mathbf{x}_{k''}]^{i,j} \quad \text{and} \quad \alpha * \langle \mathbf{x}_k \rangle^{i,j} = \langle \alpha * \mathbf{x}_k \rangle^{i,j} = \langle \mathbf{x}_{k''} \rangle^{i,j}$$

Realization of these operations depends on the underlying two-party commitment scheme. In the full version [20] we describe how addition of commitments and scalar multiplication are supported with the scheme of [18], where we also show how to extend this to enable a componentwise multiplication of an actual-commitment with a public vector from \mathbb{F}^m as well. To this end, we show how to extend their scheme to supports this operation as well, as it follows the same approach used in MiniMAC [16]. In the following we assume that public vector componentwise multiplication is supported in the two-party scheme.

3.2 Multiparty Homomorphic Commitments

Functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$, presented in Fig. 8, is a generalization of $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$ to the multiparty setting where the commands **Init**, **Commit**, **Input**, **Rand**, **Open** and **Linear Combination** have the same purpose as before. The additional command **Partial Open** allows the parties to open a commitment to a single party only (in contrast to **Open** that opens a commitment to *all* parties). As before, the functionality maintains the dictionaries **raw** and **actual** to keep track on the raw and actual commitments. The major change in the multiparty setting is that *all* parties take the role of both the committer and receiver (i.e. P_i and P_j from the two-party setting). For every commitment stored by the functionality (either raw or actual), both the commitment information and the opening information are secret shared between P_1, \dots, P_p using a full-threshold secret sharing scheme.

3.3 Realizing $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ in the $(\mathcal{F}_{\text{EQ}}, \mathcal{F}_{\text{CT}}, \mathcal{F}_{2\text{HCOM-}\mathbb{F}^m})$ -hybrid Model

Let us first fix the notation for the multiparty homomorphic commitments: We use $[\mathbf{x}]$ to denote a (multiparty) commitment to the message \mathbf{x} . As mentioned above, both the message \mathbf{x} and the commitment to it $[\mathbf{x}]$ are secret shared between the parties, that is, party P_i holds \mathbf{x}^i and $[\mathbf{x}]^i$ such that $\mathbf{x} = \sum_{i \in [p]} \mathbf{x}^i$ and $[\mathbf{x}]^i$ is composed of the information described in the following. By issuing the **Commit** command, party P_i sends $[\mathbf{x}^i]^{i,j}$ for every $j \neq i$ (by invoking the **Commit** command from $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$). Thus, party P_i holds the opening information for all instances of the commitments to \mathbf{x}^i toward all other parties, that is, it holds $\{ \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [p] \setminus \{i\}}$. In addition, P_i holds the commitment information received from all other parties, \mathbf{x}^j (for $j \neq i$), that is, it holds $\{ [\mathbf{x}^j]^{j,i} \}_{j \in [p] \setminus \{i\}}$.

<p>Functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$: Interacts with parties P_1, \dots, P_p and an adversary \mathcal{A}, who may cause the functionality to abort at any time:</p> <p>Init: Upon receiving (<code>init</code>) from all parties, forward the message to \mathcal{A} and initialize empty dictionaries <code>raw</code> and <code>actual</code>.</p> <p>Commit: Upon receiving (<code>commit</code>, I) where I is a set of γ freshly new identifiers, for every $k \in I$ store <code>raw</code>[k] = \top and send (<code>commit</code>, I) to all parties and \mathcal{A}.</p> <p>Input: Upon receiving a message (<code>Input</code>, i, k, \mathbf{y}) from P_i and (<code>Input</code>, i, k) from all other parties, if <code>raw</code>[k] $\neq \perp$ then assign <code>raw</code>[k] = \perp, assign <code>actual</code>[k] = \mathbf{y} and send (<code>Input</code>, i, k) to all parties and \mathcal{A}.</p> <p>Rand: Upon receiving a message (<code>random</code>, k) from all parties, if <code>raw</code>[k] $\neq \perp$ then pick a random $\mathbf{x}_k \in_R \mathbb{F}^m$, assign <code>raw</code>[k] = \mathbf{x}_k and send (<code>random</code>, k) to all parties and \mathcal{A}.</p> <p>Linear Combination: Upon receiving a message (<code>linear</code>, $\{(k, \alpha_k)\}_{k \in K}, \beta, k'$) for $\alpha_k, \beta \in \mathbb{F}^m$ from all parties, if <code>actual</code>[k] = $\mathbf{x}_k \neq \perp$ for all $k \in K$ and <code>raw</code>[k'] = \perp then store <code>actual</code>[k'] = $\beta + \sum_{k \in K} \alpha_k * \mathbf{x}_k$ and send (<code>linear</code>, $\{(k, \alpha_k)\}_{k \in K}, \beta, k'$) to all parties and \mathcal{A}.</p> <p>Open: Upon receiving a message (<code>open</code>, k) from all parties, if <code>actual</code>[k] = $\mathbf{x}_k \neq \perp$ then send (<code>opened</code>, k, \mathbf{x}_k) to \mathcal{A}. \mathcal{A} may then abort the protocol, otherwise send (<code>opened</code>, k, \mathbf{x}_k) to the honest parties.</p> <p>Partial Open: Upon receiving a message (<code>open</code>, i, k) from all parties, if <code>actual</code>[k] = $\mathbf{x}_k \neq \perp$ then send (<code>opened</code>, i, k, \mathbf{x}_k) to party P_i and (<code>opened</code>, i, k) to all other parties and \mathcal{A}.</p>

Fig. 8. Ideal functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$

All that information that P_i holds with regard to the value \mathbf{x} is denoted by $\llbracket \mathbf{x} \rrbracket^i$, which can be seen as a share to the multiparty commitment $\llbracket \mathbf{x} \rrbracket$.

In protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$ (from Fig. 9) each party has a local copy of the `raw` and `actual` dictionaries described above, that is, party P_i maintains `raw` ^{i} and `actual` ^{i} . In the protocol, P_i may be required to store $\llbracket \mathbf{x} \rrbracket^i$ (i.e. its share of $\llbracket \mathbf{x} \rrbracket$) in a dictionary (either `raw` ^{i} or `actual` ^{i}) under some identifier k , in such case P_i actually assigns `raw` ^{i} [k] = $\{[\mathbf{x}^j]^{j,i}, \langle \mathbf{x}^i \rangle^{i,j}\}_{j \in [p] \setminus \{i\}}$ which may also be written as `raw` ^{i} [k] = $\llbracket \mathbf{x} \rrbracket^i$.

In the following we explain the main techniques used to implement the instructions of functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ (we skip the instructions that are straightforward):

Linear operations. From the linearity of the underlying two-party homomorphic commitment functionality it follows that performing linear combinations over a multiparty commitments can be done locally by every party. We describe the notation in the natural way as follows: Given multiparty commitments $\llbracket \mathbf{x} \rrbracket$ and $\llbracket \mathbf{y} \rrbracket$ and a constant public vector $\mathbf{c} \in \mathbb{F}^m$:

– **Addition.** For every party P_i :

$$\begin{aligned} \llbracket \mathbf{x} \rrbracket^i + \llbracket \mathbf{y} \rrbracket^i &= \{ [\mathbf{x}^j]^{j,i}, \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [p] \setminus i} + \{ [\mathbf{y}^j]^{j,i}, \langle \mathbf{y}^i \rangle^{i,j} \}_{j \in [p] \setminus i} \\ &= \{ [\mathbf{x}^j]^{j,i} + [\mathbf{y}^j]^{j,i}, \langle \mathbf{x}^i \rangle^{i,j} + \langle \mathbf{y}^i \rangle^{i,j} \}_{j \in [p] \setminus i} \\ &= \{ [\mathbf{x}^j + \mathbf{y}^j]^{j,i}, \langle \mathbf{x}^i + \mathbf{y}^i \rangle^{i,j} \}_{j \in [p] \setminus i} = \llbracket \mathbf{x} + \mathbf{y} \rrbracket^i \end{aligned}$$

– **Constant addition.** The parties obtain $\llbracket \beta + \mathbf{x} \rrbracket$ by having P_1 perform $\mathbf{x}^i = \mathbf{x}^i + \beta$, then, party P_1 computes:

$$\beta + \llbracket \mathbf{x} \rrbracket^i = \beta + \{ [\mathbf{x}^j]^{j,i}, \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [2,p]} = \{ [\mathbf{x}^j]^{j,i}, \beta + \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [2,p]} = \llbracket \beta + \mathbf{x} \rrbracket^i$$

and all other parties P_j compute:

$$\begin{aligned} \beta + \llbracket \mathbf{x} \rrbracket^j &= \beta + \{ [\mathbf{x}^i]^{i,j}, \langle \mathbf{x}^j \rangle^{j,i} \}_{j \in [p] \setminus j} = \{ [\mathbf{x}^i]^{i,j}, \langle \mathbf{x}^j \rangle^{j,i} \}_{j \in [2,p] \setminus j} \cup \{ [\beta + \mathbf{x}^1]^{1,j}, \langle \mathbf{x}^j \rangle^{j,1} \} \\ &= \llbracket \beta + \mathbf{x} \rrbracket^j \end{aligned}$$

– **Constant multiplication.** For every party P_i :

$$\alpha * \llbracket \mathbf{x} \rrbracket^i = \alpha * \{ [\mathbf{x}^j]^{j,i}, \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [p] \setminus i} = \{ \alpha * [\mathbf{x}^j]^{j,i}, \alpha * \langle \mathbf{x}^i \rangle^{i,j} \}_{j \in [p] \setminus i} = \llbracket \alpha * \mathbf{x} \rrbracket^i$$

Notice that public addition is carried out by only adding the constant β to *one* commitment (we arbitrarily chose P_1 's commitment). This is so, since the true value committed to in a multiparty commitment is additively shared between p parties. Thus, if β was added to each share, then what would actually be committed to would be $p \cdot \beta!$ On the other hand, for public multiplication we need to multiply the constant α with *each* commitment, so that the sum of the shares will all be multiplied with α .

Commit. As the parties produce a batch of commitments rather than a single one at a time, assume the parties wish to produce γ commitments, each party picks $\gamma + s$ uniformly random messages from \mathbb{F}^m . Each party commit to each of these $\gamma + s$ messages towards each other party using different instances of the **Commit** command from $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$, and thus different randomness.

Note that a malicious party might use the two-party commitment scheme to commit to different messages toward different parties, which leads to an incorrect multiparty commitment. To thwart this, we have the parties execute random linear combination checks as done for batch-opening of commitments in [18]: The parties invoke the coin-tossing protocol to agree on a $s \times \gamma$ matrix, \mathbf{R} with elements in \mathbb{F} . In the following we denote the element in the q th row of the k th column of \mathbf{R} by $\mathbf{R}_{q,k}$. Every party computes s random linear combinations of the opening information that it holds toward every other party. Similarly, every party computes s combinations of the commitments that it obtained from every other party. The coefficients of the q th combination are determined by the q 'th row \mathbf{R} and the q th vector from the s “extra” committed messages added to the combination. That is, let the $\gamma + s$ messages committed by party P_i toward P_j be $\mathbf{x}_1^{i,j}, \dots, \mathbf{x}_{\gamma+s}^{i,j}$ and see that the q th combination computed by P_j is $\left(\sum_{k \in \gamma} \mathbf{R}_{q,k} \cdot \mathbf{x}_k^{i,j} \right) + \mathbf{x}_{\gamma+q}^{i,j}$ and the combination computed by P_i

is $\left(\sum_{k \in \gamma} \mathbf{R}_{q,k} \cdot \langle \mathbf{x}_k^{i,j} \rangle\right) + \langle \mathbf{x}_{\gamma+q}^{i,j} \rangle$. Then P_i open the result to P_j , who checks that it is correct. If P_i was honest it committed to the same values towards all parties and so $\mathbf{x}_k^i = \mathbf{x}_k^{i,j} = \mathbf{x}_k^{i,j'}$ for all $k \in [\gamma + s]$ and $j \neq j' \in [p] \setminus \{i\}$. Likewise for the other parties, so if everyone is honest they all obtain the same result from the opening of the combination. Thus, a secure equality check would be correct in this case. However, if P_i cheated, and committed to different values toward different parties than this is detected with overwhelming probability, since the parties perform s such combinations.

Input. Each party does a partial opening (see below) of a raw, unused commitment towards the party that is supposed to give input. Based on the opened message the inputting party computes a *correction value*. That is, if the raw commitment, before issuing the input command, is a shared commitment to the value \mathbf{x} and the inputting party wish to input \mathbf{y} , then it computes the value $\epsilon = \mathbf{y} - \mathbf{x}$ and sends this value to all parties. All parties then add $\llbracket \mathbf{x} \rrbracket + \epsilon$ to the dictionary *actual* and remove it from the dictionary *raw*. Since the party giving input is the only one who knows the value \mathbf{x} , and it is random, this does not leak.

We prove the following theorem in the full version [20].

Theorem 3.1. *Protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$ (of Fig. 9) UC-securely realizes functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ (of Fig. 8) in the $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$, \mathcal{F}_{CT} , and \mathcal{F}_{EQ} -hybrid model, against a static and malicious adversary corrupting any majority of the parties.*

4 Committed Multiparty Computation

4.1 Augmented Commitments

In the malicious, dishonest majority setting, our protocol, as other protocols, works in the offline-online model. The offline phase consists of constructing sufficiently many multiplication triples which are later used in the online phase to carry out a secure multiplications over committed, secret shared values⁵. To this end, we augment functionality $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ with the instruction **Mult** that uses the multiparty raw-commitments that were created by the **Commit** instruction of Fig. 8 and produces multiplication triples of the form $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$ such that $\mathbf{x} * \mathbf{y} = \mathbf{z}$. Note that a single multiplication triple is actually three multiparty commitments to values from \mathbb{F}^m such that \mathbf{z} is the result of a componentwise multiplication of \mathbf{x} and \mathbf{y} . That actually means that $\mathbf{z}_q = \mathbf{x}_q \cdot \mathbf{y}_q$ for every $q \in [m]$. Hence, this features the ability to securely evaluate up to m instances of the circuit at the same cost of evaluation of a single instance (i.e. in case the parties want to evaluate some circuit m times but with different inputs each time) where all m instances are being evaluated simultaneously. If the parties wish to evaluate only $m' < m$ instances of the circuit, say $m' = 1$, they do so by using only the values stored in the first component of the vectors, while ignoring

⁵ Typically a secure addition can be carried out locally by each party.

Protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$. Interacts between p parties.

Init: On input (init) from all parties each pair of parties P_i and P_j invoke the command (init) of functionality $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$ to initialize an instances denoted by $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}^{i,j}$.

Commit: To obtain a multiparty commitment to γ random values from \mathbb{F}^m :

1. The parties agree on a set I' of $\gamma + s$ freshly new identifiers.
2. Every party P_i engages in $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}^{i,j}$ for all $j \neq i$ by sending the message (commit, I') and receiving the message (committed, $\{(k, \mathbf{x}_k^{i,j})\}_{k \in I'}$). As a result, P_j receives the message (committed, I') from $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}^{i,j}$ for all $j \neq i$.
3. Every party P_i chooses $\mathbf{x}_k^i \in_R \mathbb{F}^m$ for every $k \in I'$. This is the value that is going to be committed from P_i toward all other parties.
4. Every party P_i engages in $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}^{i,j}$ for all $j \neq i$ by sending the message (Input, k, \mathbf{x}_k^i) and receives back $\llbracket \mathbf{x} \rrbracket^i = \{[\mathbf{x}_k^j]^{i,j}, \langle \mathbf{x}_k^i \rangle^{i,j}\}_{j \in [p] \setminus \{i\}}$ for every $k \in I'$.
5. The parties agree on sets I and S such that $|I| = \gamma$, $|S| = s$, $I \cap S = \emptyset$ and $I \cup S = I'$.
6. The parties issue the command (toss, $\mathbb{F}^{s \times \gamma}$) to functionality \mathcal{F}_{CT} , by which they receive (random, \mathbf{R}) where $\mathbf{R} \in \mathbb{F}^{s \times \gamma}$. We denote the element of the q th row of the k th column by $\mathbf{R}_{q,k}$.
7. For every $q \in S$ every party P_i computes $\langle \mathbf{s}_q^i \rangle^{i,j} = \langle \mathbf{x}_q^i \rangle^{i,j} + \sum_{k \in I} \mathbf{R}_{q,k} \cdot \langle \mathbf{x}_k^i \rangle^{i,j} = \langle \mathbf{x}_q^i + \sum_{k \in I} \mathbf{R}_{q,k} \cdot \mathbf{x}_k^i \rangle^{i,j}$ and sends $\langle \mathbf{s}_q^i \rangle^{i,j}$ to P_j . P_j then computes $[\mathbf{s}_q^j]^{i,j} = [\mathbf{x}_q^j]^{i,j} + \sum_{k \in I} \mathbf{R}_{q,k} \cdot [\mathbf{x}_k^i]^{i,j} = [\mathbf{x}_q^i + \sum_{k \in I} \mathbf{R}_{q,k} \cdot \mathbf{x}_k^i]^{i,j}$ and reveals \mathbf{s}_q^j .
8. For every $q \in I$ every party P_i computes $\mathbf{c}_q^i = \sum_{j \in [p]} \mathbf{s}_q^j$ and inputs (equal, i, \mathbf{c}_q^i) to \mathcal{F}_{EQ} . If \mathcal{F}_{EQ} responds with abort or (equal, $\mathbf{s}_q^1, \dots, \mathbf{s}_q^p$, reject) in any of these calls then abort, otherwise output (committed, I). For every $k \in I$ store $\text{raw}^i[k] = \llbracket \mathbf{x}_k \rrbracket^i$.

Input: Upon input (Input, i, k, \mathbf{y}) from party i and (Input, i, k) from all other parties:

1. Party P_j (for $j \neq i$) aborts if $\text{raw}^j[k] = \perp$. Otherwise P_j sends $\langle \mathbf{x}_k^j \rangle^{i,j}$ to P_i (using (open, k)), who learns \mathbf{x}_k^j .
2. Party P_i computes $\mathbf{x}_k = \sum_{j \in [p]} \mathbf{x}_k^j$ and broadcasts $\epsilon_k = \mathbf{y} - \mathbf{x}_k$ to all other parties.
3. Party P_i updates $\llbracket \mathbf{x}_k \rrbracket^i$ by setting the opening values to $\langle \mathbf{x}^i + \epsilon_k \rangle^{i,j} = \langle \mathbf{x}_k^i \rangle^{i,j} + \epsilon_k$ for all $j \in [p]$. Similarly, party P_j (for $j \neq i$) updates $\llbracket \mathbf{x}_k \rrbracket^j$ by setting the i th commitment information to be $[\mathbf{x}_k^i + \epsilon_k]^{i,j} = [\mathbf{x}_k^i]^{i,j} + \epsilon_k$.
4. Party P_j (for all $j \in [p]$) assigns $\text{raw}^j[k] = \perp$ and $\text{actual}^j = \llbracket \mathbf{x}_k \rrbracket^j$.

Rand: The parties agree on an arbitrary k such that $\text{raw}^i[k] = \llbracket \mathbf{x}_k \rrbracket^i \neq \perp$ for all $i \in [p]$, set $\text{raw}^i[k] = \perp$ and $\text{actual}^i[k] = \llbracket \mathbf{x}_k \rrbracket^i$.

Linear Combination: The parties agree on a set of indices K and the public vectors $\{\alpha_k\}_{k \in K}$ such that $\text{actual}[k] \neq \perp$ and $\alpha_k \in \mathbb{F}^m$ for every $k \in K$. In addition, the parties agree on a public vector $\beta \in \mathbb{F}^m$ and an index k' such that $\text{raw}[k'] = \text{actual}[k'] = \perp$. Finally, every party P_i stores $\text{actual}[k'] = \beta + \sum_{k \in K} \alpha_k * \llbracket \mathbf{x}_k \rrbracket^i$.

Open: To open $\llbracket \mathbf{x}_k \rrbracket$, every party P_i sends $\langle \mathbf{x}_k^i \rangle^{i,j}$ to P_j for all $j \neq i$. Then, party P_i obtains \mathbf{x}_k^j from the commitment and the opening information $[\mathbf{x}_k^j]^{i,j}$ and $\langle \mathbf{x}_k^i \rangle^{i,j}$ respectively. Finally P_i computes $\mathbf{x}_k = \sum_{j \in [p]} \mathbf{x}_k^j$.

Partial Open: To open $\llbracket \mathbf{x}_k \rrbracket$ to party P_i , every party P_j (for $j \neq i$) sends $\langle \mathbf{x}_k^j \rangle^{i,j}$ to P_i . Then, party P_i obtains \mathbf{x}_k^j from the commitment and the opening information $[\mathbf{x}_k^j]^{i,j}$ and $\langle \mathbf{x}_k^i \rangle^{i,j}$ respectively. Finally P_i computes $\mathbf{x}_k = \sum_{j \in [p]} \mathbf{x}_k^j$.

Fig. 9. Protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$

the rest of the components. However, using a multiplication triple wastes *all* components of \mathbf{x} , \mathbf{y} and \mathbf{z} even if the parties wish to use only their first component. To avoid such a loss we augment $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ with the instruction **ReOrg**. The **ReOrg** instruction preprocesses *reorganization pairs* which are used to compute a linear operator over a multiparty commitment. For example this enable the parties to “copy” the first component to another, new, multiparty commitment, such that all components of the new multiparty commitment are equal to the first component of the old one. For instance, the linear operator $\phi \in \mathbb{F}^{m \times m}$ such that its first column is all 1 and all other columns are all 0, transforms the vector \mathbf{x} to $\mathbf{x}' = \mathbf{x}_1, \dots, \mathbf{x}_1$ (m times). Applying ϕ to \mathbf{y} and \mathbf{z} as well results in a new multiplication triple $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$ where only the first component of $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ got used (rather than all their m components). We note that the construction of reorganization pairs are done in a batch for *each* function ϕ resulting in the additive destruction of s extra raw commitments (i.e. an additive overhead). In the **ReOrg** command, described in Fig. 10, the linear operator ϕ is applied to L raw commitments in a batch manner. The inputs to ϕ are the messages stored by the functionality under identifiers from the set X and the outputs override the messages stored by the functionality under identifiers from the set Y . The messages stored under identifiers from the set R are being destroyed (this reflects the additive overhead of that command).

Adding instructions **Mult** and **ReOrg** to the $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$ functionality, we get the augmented functionality $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ formally presented in Fig. 10.

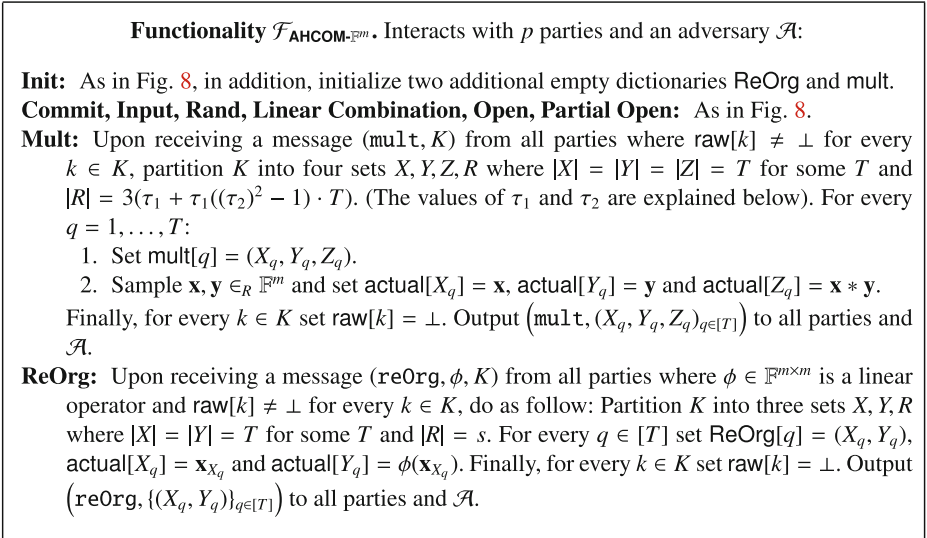


Fig. 10. Ideal functionality $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$

Realizing $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$. The protocol $\Pi_{\text{AHCOM-}\mathbb{F}^m}$ is formally presented in Figs. 12 and 13. In the following we describe the techniques used in $\Pi_{\text{AHCOM-}\mathbb{F}^m}$

and show the analysis that implies the number of multiplication triples that should be constructed in one batch for the protocol to be secure. Specifically, in Sect. 4.2 we describe how to implement the **Mult** command and in Sect. 4.3 we describe how to implement the **ReOrg** command.

4.2 Generating Multiplication Triples

Secure multiplication in our online phase, similar to previous works in the field, is performed using multiplication triples (AKA Beaver triples). In our work a multiplication triple is of the form $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$ where $\llbracket \mathbf{x} \rrbracket$, $\llbracket \mathbf{y} \rrbracket$ and $\llbracket \mathbf{z} \rrbracket$ are multiparty commitments of messages \mathbf{x} , \mathbf{y} and \mathbf{z} respectively as defined in Sect. 3.3 and $\mathbf{z} = \mathbf{x} * \mathbf{y}$. The construction of triples is done in a batch and consists of four parts briefly described below (and further explained and analyzed soon afterward):

1. **Construction.** Using the arithmetic OT procedure formalized in Sect. 2 the parties first *construct* multiplication triples that may be “malformed” and “leaky” in case of a malicious adversary. Here malformed means that they are incorrect, i.e. $\mathbf{x} * \mathbf{y} \neq \mathbf{z}$ and “leaky” means that the adversary has tried to guess the value of the share of an honest party (the term is further explained below).
2. **Cut-and-Choose.** The parties select τ_1 triples at random which they check for correctness. If any of these triples are malformed then they abort. Otherwise, when mapping the remaining triples into buckets, with overwhelming probability all buckets will contain at least one correct triple.
3. **Sacrificing.** The remaining triples (from the cut-and-choose) are mapped to buckets, τ_1 triples in each bucket such that at least one of the triples is correct. Each bucket is then tested to check its correctness where by this check only a single multiplication is being output while the other $\tau_1 - 1$ are being discarded. This step guarantees that either the output triple is correct or a malformed triple is detected, in which case the protocol aborts.
4. **Combining.** As some of the triples may be “leaky” this allows the adversary to carry a selective attack, that is, to probe whether its guess was correct or not. If the guess is affected by the input of an honest party then it means that the adversary learns that input. Thus, as the name suggests, the goal of this step is to produce a non-leaky triple by combining τ_2 triples, which are the result of the sacrificing step (and thus are guaranteed to be correct), where at least one of the τ_2 is non-leaky. As we will see later, this condition is satisfied with overwhelming probability.

Construction. The triples are generated in a batch, that is, sufficiently many triples are generated at once. However, the construction of each triple is independent of the others. Thus, we proceed by describing how to generate a single triple. The parties select three raw-commitments, denoted $\llbracket \mathbf{x} \rrbracket$, $\llbracket \mathbf{y} \rrbracket$, $\llbracket \mathbf{z}' \rrbracket$, that were generated by $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$. The goal of this step is to change $\llbracket \mathbf{z}' \rrbracket$ to $\llbracket \mathbf{z} \rrbracket$ such that $\llbracket \mathbf{z} \rrbracket = \llbracket \mathbf{x} * \mathbf{y} \rrbracket$.

Recall that for a message \mathbf{x} that is committed to by all parties, we have that each party P_i knows \mathbf{x}^i such that $\mathbf{x} = \sum_{i \in [p]} \mathbf{x}^i$. Thus, the product $\mathbf{x} * \mathbf{y}$ equals $(\sum_{i \in [p]} \mathbf{x}^i) * (\sum_{i \in [p]} \mathbf{y}^j) = \sum_{i \in [p]} \mathbf{x}^i * (\sum_{j \in [p]} \mathbf{y}^j)$. In order to have each party P_i hold the value \mathbf{z}^i such that $\sum_{i \in [p]} \mathbf{z}^i = \mathbf{x} * \mathbf{y}$ we let party P_i use the arithmetic OT procedure (as describe in Sect. 2) to have a share of the multiplication $\mathbf{x}^i * \mathbf{y}^j$ for every $j \in [p]$ where P_i inputs \mathbf{x}^i and P_j inputs \mathbf{y}^j . After P_i multiplied its share \mathbf{x}^i with all other parties' shares \mathbf{y}^j the sum of all the shares is $\mathbf{x}^i * (\sum_{j \in [p]} \mathbf{y}^j)$ (assuming honest behavior). If all parties do the same, then every party ends up holding a share of $\mathbf{x} * \mathbf{y}$ as required. Remember that we want P_i to hold a share to $[\mathbf{x} * \mathbf{y}]$ and not just a share to $\mathbf{x} * \mathbf{y}$ (i.e. we want all shares to be committed). To this end, every party broadcasts the difference \mathbf{t} between the new share and the old share, that is, P_i broadcasts $\mathbf{t}^i = \mathbf{z}^i - \mathbf{z}''^i$, then, the parties perform a constant addition to the old commitments, that is, they compute $[\mathbf{z}] = [\mathbf{z}'] + (\sum_{i \in [p]} \mathbf{t}^i)$.

Discussion. As described above, party P_i (for $i \in [p]$) participates in $p - 1$ instantiations of the arithmetic OT functionality with every other party P_j (for $j \neq i$). The arithmetic OT functionality is of the form $(\mathbf{x}^i, (\mathbf{y}^j, \mathbf{r}^j)) \mapsto (\mathbf{x}^i * \mathbf{y}^j + \mathbf{r}^j, \perp)$, that is, P_i inputs its share \mathbf{x}^i of \mathbf{x} , party P_j inputs its share \mathbf{y}^j of \mathbf{y} and a random value \mathbf{r}^j . The functionality outputs $\mathbf{x}^i * \mathbf{y}^j + \mathbf{r}^j$ to P_i and nothing to P_j . Then, to get a sharing of $\mathbf{x}^i * \mathbf{y}^j$ we instruct P_i to store $\mathbf{x}^i * \mathbf{y}^j + \mathbf{r}^j$ and P_j to store $-\mathbf{r}^j$ (see Sect. 2). Even if this arithmetic OT subprotocol is maliciously secure, it will only give semi-honest security in our setting when composed with the rest of the scheme. Specifically, there are two possible attacks that might be carried out by a malicious adversary:

1. Party P_j may input $\tilde{\mathbf{y}}^j \neq \mathbf{y}^j$ such that $\mathbf{e} = \tilde{\mathbf{y}}^j - \mathbf{y}^j$, in the instantiation of the arithmetic OT with every other P_i , where \mathbf{y}^j is the value it is committed to. This results with the parties obtaining a committed share of the triple $([\mathbf{x}], [\mathbf{y}], [\mathbf{x} * (\mathbf{y} + \mathbf{e})])$. We call such a triple a “malformed” triple.
2. In the arithmetic OT procedure party P_j may impact the output of P_i such that P_i obtains $\mathbf{x}^i * \mathbf{y}^j + \mathbf{r}^j$ only if the k 'th value of \mathbf{x}^i is equal to some value “guessed” by P_j , otherwise P_i obtains some garbage $\mathbf{x}^i * \tilde{\mathbf{y}}^i \in \mathbb{F}^m$. A similar attack can be carried out by P_i on \mathbf{y}^j when computing over a “small” field (see the description of the malicious behavior in Sect. 2). In both cases, the parties obtain committed shares of the triple $([\mathbf{x}], [\mathbf{y}], [\mathbf{x} * \mathbf{y}])$ only if the malicious party made a correct guess on an honest party's share, and an incorrect triple otherwise. Thus, when using this triple later on, the malicious party learns if it guessed correctly depending on whether the honest parties abort, thus, it is vulnerable to a “selective attack”. We call such a triple “leaky”, since it might leak private bits from the input of an honest party.

We take three countermeasures (described in the next items) to produce correct and non-leaky triples:

1. In the *Cut-and-Choose* step we verify that a few (τ_1) randomly selected triples have been constructed correctly. This is done, by having each party open

his committed shares associated with these triples and all parties verifying that the triples has been constructed according to the protocol. This step is required to ensure that not all triples were malformed as a preliminary for the sacrificing step (below) in which the triples are mapped to buckets. When working over $\mathbb{F} = \text{GF}(2)$, this step is strictly needed to eliminate the case that all triples are malformed. For other fields, this step improves the amount of triples to be constructed in the batch.

2. In the *Sacrificing* step we make sure that a triple is correct (i.e. not malformed) by “sacrificing” $\tau_1 - 1$ other triples which are being used as a “one-time-pads” of the correct triple. As we treat a bunch of triples at once, the probability of an incorrect triple to pass this step without being detected is negligible in s (analysis is presented below). Having the parties committed (in the construction step) to $\tau_1 \cdot T$ triples, by the end of this step there will be T correct triples.
3. In the *Combining* step we partition the constructed (correct but possibly leaky) triples into buckets of τ_2 triples each, and show that for a sufficiently big number of triples that are the outcome of the sacrificing step, the probability that there exist a bucket in which all triples are leaky in a specific component is negligible in s . We show how to combine the τ_2 triples in a bucket and produce a new triple which is non-leaky. This is done twice, first to remove leakage on the \mathbf{x} component and second to remove leakage on the \mathbf{y} component.

Cut-and-Choose. The parties use \mathcal{F}_{CT} to randomly pick τ_1 triples to check. Note that τ_1 is the bucket-size used in *Sacrificing* below and in practice could be as low as 3 or 4. It was shown in [21] that when partitioning the triples into buckets of size τ_1 (where many of them may be malformed) then by sampling and checking only τ_1 triples, the probability that there exist a bucket full of malformed triples is negligible. Formally:

Corollary 4.1 (Corollary 6.4 in [21]). *Let $N = \tau_1 + \tau_1(\tau_2)^2 \cdot T$ be the number of constructed triples where $s \leq \log_2 \left(\frac{(N \cdot \tau_1 + \tau_1)!}{N \cdot \tau_1! \cdot (N \cdot \tau_1)!} \right)$, then, by opening τ_1 triples it holds that every bucket contains at least one correct triple with overwhelming probability.*

Hence, it is sufficient to open (and discard) τ_1 triples out of the triples from the Construction step and hand the remaining to the Sacrificing step below.

Sacrificing. In the following we describe how to produce $(\tau_2)^2 \cdot T$ correct triples out of $\tau_1 \cdot (\tau_2)^2 \cdot T$ that were remained from the cut-and-choose step, and analyze what should T and τ_1 be in order to have all produced $(\tau_2)^2 \cdot T$ triples correct with overwhelming probability. We have the $(\tau_2)^2 \cdot T$ triples be uniformly assigned to buckets where each bucket contains τ_1 triples, denoted $\{t_k\}_{k \in [\tau_1]}$. For simplicity, in the following we assume that $\tau_1 = 3$. For every bucket, the parties apply the procedure `CorrectnessTest` (see Fig. 11) to triples t_1 and t_2 . If the procedure

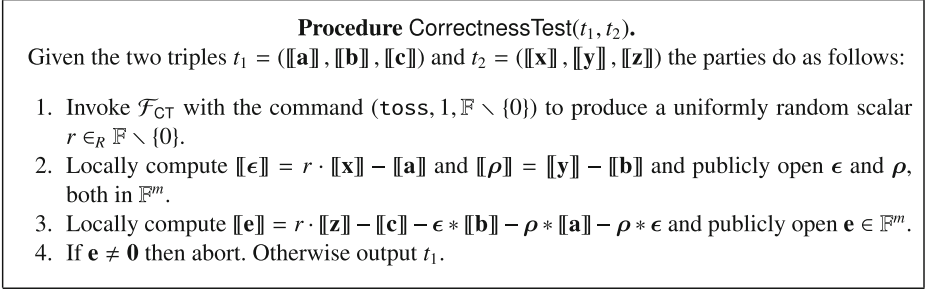


Fig. 11. Procedure CorrectnessTest(t_1, t_2)

returns successfully (i.e. the parties do not abort) they run the procedure again, this time with triples t_1 and t_3 . Finally, if the procedure returns successfully from the second invocation as well then they consider t_1 as a correct triple, otherwise they abort the protocol. We note that this procedure is similar to the one used in [14] and other works.

Security. The correctness and security is explained in [14]. However, for completeness we prove the following lemma in the full version [20], which states that after the sacrificing step all produced triples are correct with overwhelming probability:

Lemma 4.2. *When $2^{-s} \leq \frac{(|\mathbb{F}|-1)^{1-\tau_1} \cdot (\tau_2)^2 \cdot T \cdot (\tau_1 \cdot (\tau_2)^2 \cdot T)! \cdot \tau_1!}{(\tau_1 \cdot (\tau_2)^2 \cdot T + \tau_1)!}$ all the $(\tau_2)^2 \cdot T$ triples that are produced by the sacrificing step are correct except with probability at most 2^{-s} .*

Combining. The goal of this step is to produce T non-leaky triples out of the $(\tau_2)^2 \cdot T$ triples remained from the sacrificing step above. We do this in two sub-steps: First to remove the leakage (with regard to the arithmetic OT) of the sender and then to remove the leakage from the receiver. In each of the sub-steps we map the triples to buckets of size τ_2 and produce a single non-leaky triple out of it. In the following we first show how to produce one triple from each bucket with the apriori knowledge that at least one of the triples in the bucket is non-leaky (but we do not know which one is it) and later we show how to obtain such buckets. Denote the set of τ_2 triples by $\{(\llbracket \mathbf{x}_k \rrbracket, \llbracket \mathbf{y}_k \rrbracket, \llbracket \mathbf{z}_k \rrbracket)\}_{k \in [\tau_2]}$. We produce the triple $(\llbracket \mathbf{x}' \rrbracket, \llbracket \mathbf{y}' \rrbracket, \llbracket \mathbf{z}' \rrbracket)$ out of that set in the following way: The parties compute

$$\llbracket \mathbf{x}' \rrbracket = \llbracket \sum_{k \in [\tau_2]} \mathbf{x}_k \rrbracket \quad \text{and} \quad \llbracket \mathbf{y}' \rrbracket = \llbracket \mathbf{y}_1 \rrbracket \quad \text{and} \quad \llbracket \mathbf{z}' \rrbracket = \llbracket \left(\sum_{k \in [\tau_2]} \mathbf{x}_k \right) * \mathbf{y}_1 \rrbracket$$

which constitute the triple $(\llbracket \mathbf{x}' \rrbracket, \llbracket \mathbf{y}' \rrbracket, \llbracket \mathbf{z}' \rrbracket)$. It is easy to see that $\llbracket \mathbf{x}' \rrbracket$ can be computed locally since it requires additions and constant multiplications only. Furthermore, \mathbf{x}' is completely hidden since at least one of $\mathbf{x}_1, \dots, \mathbf{x}_k$ was not

leaked (and it is guaranteed from the construction step that it is chosen uniformly at random from \mathbb{F}^m). However, notice that $\llbracket \mathbf{z}' \rrbracket$ cannot be computed locally, since it is required to multiply two multiparty commitments $\llbracket \left(\sum_{k \in [\tau_2]} \mathbf{x}_k \right) \rrbracket$ and $\llbracket \mathbf{y}_1 \rrbracket$. Thus, to obtain $\llbracket \mathbf{z}' \rrbracket$ the parties first compute $\llbracket \epsilon_k \rrbracket = \llbracket \mathbf{y}_1 - \mathbf{y}_k \rrbracket$ and open ϵ_k for every $k = 2, \dots, \tau_2$. Then compute $\llbracket \mathbf{z}' \rrbracket = \llbracket \mathbf{z}_1 + \sum_{k=2}^{\tau_2} \epsilon_k * \mathbf{x}_k + \mathbf{z}_k \rrbracket$ by a local computation only.

We prove the following lemma in the full version [20]:

Lemma 4.3. *Having a batch of at least $\tau_2^{-1} \sqrt{\frac{(s \cdot e)^{\tau_2} \cdot 2^s}{\tau_2}}$ triples as input to a combining step, every bucket of τ_2 triples contains at least one non-leaky triple with overwhelming probability in s in the component that has been combined on.*

For instance, when $\mathbb{F} = \text{GF}(2)$ having $s = 40$, $\tau_1 = 3$ $\tau_2 = 4$ it is required to construct $T \approx 8.4 \cdot 10^5$ correct and non-leaky triples in a batch. Instead, having $\tau_2 = 3$ means that $\approx 2.29 \cdot 10^8$ triples are required.

Working over Non-binary Fields. When \mathbb{F} is a field with odd characteristic then there is a gap between the maximal field element and the maximal value that is possible to choose which can fit in the same number of bits. For instance, when working over \mathbb{F}_{11} then the maximal element possible is $10_{10} = 0101_2$ while the maximal value possible to fit in 4 bits is $15_{10} = 1111_2$, i.e. there is a gap of 5 elements. That means that an adversary could input a value that is not in the field and might harm the security.

We observe that the only place where this type of attack matters is in the ArithmeticOT procedure, since in all other steps the values that the adversary inputs percolate to the underlying homomorphic commitment scheme. In the following we analyze this case: To multiply x^i and y^j with $x^i, y^j \in \mathbb{F}_{\mathcal{P}}$ and \mathcal{P} prime the parties P_i and P_j participate in a protocol of $\lceil \log \mathcal{P} \rceil$ steps. In the q -th step, where $q \in \lceil \log \mathcal{P} \rceil$, party P_i inputs x_q^i and P_2 inputs $s_q^0 = r_q$ and $s_q^1 = r_q + y^j$ to the \mathcal{F}_{OT} functionality. The functionality outputs $s^{x_q^i}$ to P_1 which updates the sum of the result. In the end of this process the parties hold shares to the multiplication $z = x^i \cdot y^j$.

We first examine the cases in which either s_q^0 or s_q^1 are not in the prime field, i.e. they belong to the gap $\text{gap} = [2^{\lceil \log \mathcal{P} \rceil}] \setminus \mathbb{F}_{\mathcal{P}}$. We first note that if both of them are in gap then this is certainly detected by P_1 (since P_1 receives one of them as the \mathcal{F}_{OT} 's output). If only one of s_q^0, s_q^1 is in gap then one of two cases occurs:

1. If the value that P_1 received from \mathcal{F}_{OT} is in gap then it is detected immediately as before (since P_1 clearly sees that the value is not in $\mathbb{F}_{\mathcal{P}}$) and can abort. Since this is the preprocessing phase it is independent of any secret input.
2. If the value that P_1 received from \mathcal{F}_{OT} is in $\mathbb{F}_{\mathcal{P}}$ but the other value is not, then it is guaranteed that the value P_1 obtains is a correct share. That the dishonest P_2 obtains a share in the gap is actually the same case as if P_2 adds an incorrect value to the sum s.t. it lands in the gap. This leads to two cases

- (a) If the incorrect value is $s_q^0 \neq r_q$ then this is equivalent to add $s_q^0 \bmod \mathcal{P}$, which leads to an incorrect share of z . This case is detected in the sacrificing step.
- (b) If the incorrect value is $s_q^1 \neq r_q + y^j$ then this is equivalent to add $s_q^1 \bmod \mathcal{P}$. As above, this leads to an incorrect share of z which is being detected in the sacrificing step.

The last case is when either r_q or y^j (or both) are not in $\mathbb{F}_{\mathcal{P}}$ but the sum s_q^1 does. Then this is equivalent to choosing $y^j \in \mathbb{F}_{\mathcal{P}}$ and $r'_q = s_q^1 - y^j \bmod \mathcal{P}$ such that the value that P_2 adds to its sum is incorrect (since it is different than r'_q), and thus, this is being detected in the sacrificing step as before.

Similarly, consider a corrupted receiver who organizes its bits of x^j to represent an element in gap . We observe that this is equivalent to a receiver who inputs an incorrect value (value that is not committed before) for the following reason: The adversary knows nothing about the sender's (honest party) share y^j , let the value that P_i inputs be \tilde{x}^i , thus the **ArithmeticOT** procedure outputs shares to $\tilde{x}^i y^j \bmod \mathcal{P} = (\tilde{x}^i \bmod \mathcal{P})(y^j \bmod \mathcal{P})$. Now, if $\tilde{x}^i \bmod \mathcal{P} = 0$ (i.e. $\tilde{x}^i = \mathcal{P}$) then this is detected by the sacrificing procedure (since $0 \in \mathbb{F}_{\mathcal{P}}$ is not in the field). Otherwise, if $\tilde{x}^i \bmod \mathcal{P} \neq 0$ then the result $\tilde{x}^i y^j \bmod \mathcal{P}$ is a random element in the field $\mathbb{F}_{\mathcal{P}}$ and the same analysis from the proof of Lemma 4.2 follows.

Finally we make the observation that the math still work out in case we use an extension field and not a plain prime-field. Basically using the **ArithmeticOT** procedure we can still multiply with one bit at a time. The parties simply multiply with the appropriate constants in the extension field (and thus do any necessary polynomial reduction), instead of simply a two-power.

We prove the following theorem in the full version [20].

Theorem 4.4. *The method **Mult** in $\Pi_{\text{AHCOT-}\mathbb{F}^m}$ (Fig. 13) UC-securely implements the method **Mult** in functionality $\mathcal{F}_{\text{AHCOT-}\mathbb{F}^m}$ (Fig. 10) in the $\mathcal{F}_{\text{OT-}}$, $\mathcal{F}_{\text{EQ-}}$ and $\mathcal{F}_{\text{CT-}}$ hybrid model against a static and malicious adversary corrupting a majority of the parties.*

4.3 Reorganization of Components of a Commitment

The parties might want to move elements of \mathbb{F} around or duplicate elements of \mathbb{F} within a message. In general we might want to apply a linear function ϕ to a vector in \mathbb{F}^m resulting in another vector in \mathbb{F}^m . To do so, they need to preprocess pairs of the form $([\mathbf{x}], [\phi(\mathbf{x})])$ where \mathbf{x} is random. This is done by first having a pair of random commitments $([\mathbf{x}], [\mathbf{y}])$ (as the output of the **Commit** instruction of $\mathcal{F}_{\text{HCOM-}\mathbb{F}^m}$), then, party P_i broadcasts $\epsilon^i = \phi(\mathbf{x}^i) - \mathbf{y}^i$ (i.e. by first applying ϕ on its own share). Note that from linearity of ϕ it follows that $\sum_{i \in [p]} \phi(\mathbf{x}^i) = \phi(\sum_{i \in [p]} \mathbf{x}^i) = \phi(\mathbf{x})$, thus $\sum_{i \in [p]} \epsilon^i = \sum_{i \in [p]} \phi(\mathbf{x}^i) - \mathbf{y}^i = \phi(\mathbf{x}) - \mathbf{y}$. Then, the parties compute $[\mathbf{y}'] = [\mathbf{y}] + \sum_{i \in [p]} \epsilon^i = [\mathbf{y}] + \phi(\mathbf{x}) - \mathbf{y} = \phi(\mathbf{x})$. For security reasons this is done simultaneously for a batch of $\nu + s$ pairs. Finally, the parties complete s random linear combination tests over the batch by producing

Protocol $\Pi_{\text{AHCOM-}\mathbb{F}^m}$. Describes the implementation of $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ in the $\mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}}, \mathcal{F}_{\text{CT}}$ -hybrid model. The protocol is an interaction between p parties, if $\mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}}$ or \mathcal{F}_{CT} outputs **abort** at any point, so does this protocol. The parties begin the protocol with an empty dictionary **ReOrg**.

Init, Commit, Input, Rand, Linear Combination, Open, Partial Open:

Do exactly as in protocol $\Pi_{\text{HCOM-}\mathbb{F}^m}$ in Fig. 9.

ReOrg: The parties wish to construct reorganization pairs based on the linear function ϕ using the raw commitments with identifiers set K where $|K| = 2\nu + 2s$ for some ν . If $\text{raw}^i[k] \neq \perp$ for each $k \in K$ and $i \in [p]$ then partition K into the sets X, Y, A, B where $|X| = |Y| = \nu$ and $|A| = |B| = s$ and proceed as follows:

1. For each of the ν pairs $\{(x, y)\} \in (X, Y)$ each party i broadcasts the value $\epsilon_{x,y}^i = \phi(\mathbf{x}_x^i) - \mathbf{x}_y^i$.
2. For each of the s pairs $\{(a, b)\} \in (A, B)$ each party i broadcasts the value $\epsilon_{a,b}^i = \phi(\mathbf{x}_a^i) - \mathbf{x}_b^i$.
3. For every pair $(x, y) \in (X, Y)$ and every pair $(a, b) \in (A, B)$ the parties pick freshly new indexes y' and b' and compute $\llbracket \mathbf{x}_{y'} \rrbracket = \llbracket \mathbf{x}_y \rrbracket + \sum_{j \in [p]} \epsilon_{x,y}^j$ and $\llbracket \mathbf{x}_{b'} \rrbracket = \llbracket \mathbf{x}_b \rrbracket + \sum_{j \in [p]} \epsilon_{a,b}^j$. Meaning that $\llbracket \mathbf{x}_{y'} \rrbracket = \llbracket \phi(\mathbf{x}_x) \rrbracket$ and $\llbracket \mathbf{x}_{b'} \rrbracket = \llbracket \phi(\mathbf{x}_a) \rrbracket$. Let Y' be the set of y' and likewise let B' be the set of b' .
4. All parties input (toss, $s \cdot \nu, \mathbb{F}$) to \mathcal{F}_{CT} and thus learn (random, \mathbf{R}) (when viewing the output as a matrix $\mathbf{R} \in \mathbb{F}^{s \times \nu}$).
5. The parties now compute and open the linear combination for each $q \in [s]$, letting $\mathbf{R}_{q,k}$ denote the element in the q 'th row of the k 'th column of \mathbf{R} :

$$\llbracket \mathbf{s}_q \rrbracket = \llbracket \mathbf{x}_{A_q} \rrbracket + \sum_{k \in [s]} \mathbf{R}_{q,k} \cdot \llbracket \mathbf{x}_{X_k} \rrbracket \quad \text{and} \quad \llbracket \bar{\mathbf{s}}_q \rrbracket = \llbracket \mathbf{x}_{B'_q} \rrbracket + \sum_{k \in [s]} \mathbf{R}_{q,k} \cdot \llbracket \mathbf{x}_{Y'_k} \rrbracket$$

6. Each party now verifies that $\phi(\mathbf{s}_q) = \bar{\mathbf{s}}_q$. If not, they abort.
7. The parties set $\text{ReOrg}^i[q] = (X_q, Y'_q)$, $\text{actual}^i[X_q] = \llbracket \mathbf{x}_{X_q} \rrbracket^i$, $\text{actual}^i[Y'_q] = \llbracket \phi(\mathbf{x}_{X_q}) \rrbracket^i$ for every $q \in [s]$ and $\text{raw}^i[k] = \perp$ for every $k \in K$. Output $(\text{reOrg}, (X, Y'))$ to all parties.

Fig. 12. Protocol $\Pi_{\text{AHCOM-}\mathbb{F}^m}$ - Part 1

a uniformly random matrix $\mathbf{R} \in \mathbb{F}^{s \times \nu}$ (using \mathcal{F}_{CT}). Let $\mathbf{R}_{q,k}$ be the element in the q th row and k th column of \mathbf{R} . To perform the test, divide the $\nu + s$ pairs into two sets A, B of ν and s pairs respectively. For each pair $(\llbracket \mathbf{z}_q \rrbracket, \llbracket \mathbf{z}'_q \rrbracket)$ in B for $q \in [s]$ compute and open

$$\llbracket \mathbf{s}_q \rrbracket = \llbracket \mathbf{z}_q \rrbracket + \sum_{k \in [\nu]} \mathbf{R}_{q,k} \cdot \llbracket \mathbf{x}_k \rrbracket \quad \text{and} \quad \llbracket \bar{\mathbf{s}}_q \rrbracket = \llbracket \mathbf{z}'_q \rrbracket + \sum_{k \in [\nu]} \mathbf{R}_{q,k} \cdot \llbracket \mathbf{y}_k \rrbracket$$

Each party now verifies that $\phi(\mathbf{s}_q) = \bar{\mathbf{s}}_q$. If this is so, they accept. Otherwise they abort.

Based on this we state the following theorem, which we prove in the full version [20].

Protocol $\Pi_{\text{AHCOM-F}^m}$. Describes the implementation of $\mathcal{F}_{\text{AHCOM-F}^m}$ in the $\mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}}, \mathcal{F}_{\text{CT}}$ -hybrid model. The protocol is an interaction between p parties, if $\mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}}$ or \mathcal{F}_{CT} output abort at any point, so does this protocol. The parties begin the protocol with an empty dictionary mult.

Mult: Upon receiving a message (mult, K) from all parties where $\text{raw}[k] \neq \perp$ for every $k \in K$, let $|K| = 3(\tau_1 + \tau_1 \cdot (\tau_2)^2 \cdot T)$, assign the commitments into $\tau_1 + \tau_1 \cdot (\tau_2)^2 \cdot T$ triples denoted by $\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket$.

1. **Construction.** For each of the $\tau_1 + \tau_1 \cdot (\tau_2)^2 \cdot T$ triples denoted by $\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket$ do as follows:
 - (a) Party P_i (for every $i \in [p]$) executes the arithmetic OT procedure $\text{ArithmeticOT}(\mathbf{x}^i, \mathbf{y}^j)$ of Fig. 6 together with every party $P_j \neq P_i$ where P_i inputs \mathbf{x}^i and P_j inputs \mathbf{y}^j . Let $\mathbf{s}_{i \leftarrow j}^i$ be the output for P_i and $\mathbf{s}_{j \leftarrow i}^i$ be the output for P_j .
 - (b) Every party P_i computes $\mathbf{s}^i = \mathbf{x}^i * \mathbf{y}^i + \sum_{j \neq i} \mathbf{s}_{i \leftarrow j}^i + \sum_{j \neq i} \mathbf{s}_{j \leftarrow i}^i$ and broadcasts $\mathbf{t}^i = \mathbf{s}^i - \mathbf{z}^i$.
 - (c) All parties compute and store $\llbracket \mathbf{z} \rrbracket = \llbracket \mathbf{z} \rrbracket + \sum_{i \in [p]} \mathbf{t}^i = \llbracket \mathbf{x} * \mathbf{y} \rrbracket$
2. **Cut-and-Choose.** Assign τ_1 randomly picked triples, out of the $\tau_1 + (\tau_2)^2 \cdot T$ triples constructed above, into a bucket using \mathcal{F}_{CT} . For each triple in this bucket, $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$, proceed as follows:
 - (a) The parties publicly open $\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket$ and $\llbracket \mathbf{z} \rrbracket$.
 - (b) Every party locally verifies if $\mathbf{x} * \mathbf{y} = \mathbf{z}$. If this is the case they discard the triple $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$, otherwise they abort.
3. **Sacrificing.** Let $\tau_1 \cdot (\tau_2)^2 \cdot T$ be the number of triples remaining, where each triple is of the form $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$. The parties do as follows:
 - (a) Assign the triples uniformly into τ_1 buckets where each bucket contains exactly τ_1 triples, denoted t_1, \dots, t_{τ_1} (the uniform assignment done via the use of the coin tossing functionality \mathcal{F}_{CT}).
 - (b) Run $\text{CorrectnessTest}(t_1, t_k)$ for $k \in \{2, \dots, \tau_1\}$ (see Fig. 11) where k is the raw-commitment ID of $\llbracket \mathbf{x} \rrbracket$. Note that according to the procedure, if a malformed triple is detected then the parties abort.
 - (c) Consider t_1 as a correct triple.
4. **Combining.** Let $(\tau_2)^2 \cdot T$ be the number of correct triples produced by the above step.
 - (a) **Combine on \mathbf{x} :** The parties assign the triples uniformly into $\tau_2 T$ buckets of τ_2 triples each (as before, this is done using \mathcal{F}_{CT}). For every bucket, denote the triples it contain by $\{(\llbracket \mathbf{x}_k \rrbracket, \llbracket \mathbf{y}_k \rrbracket, \llbracket \mathbf{z}_k \rrbracket)\}_{k \in [\tau_2]}$ the parties do as follows:
 - i. Compute $\llbracket \mathbf{x}' \rrbracket = \llbracket \sum_{k \in [\tau_2]} \mathbf{x}_k \rrbracket$ and $\llbracket \mathbf{y}' \rrbracket = \llbracket \mathbf{y}_1 \rrbracket$
 - ii. Compute $\llbracket \epsilon_k \rrbracket = \llbracket \mathbf{y}_1 - \mathbf{y}_k \rrbracket$ and open ϵ_k for every $k = \{2, \dots, \tau_2\}$.
 - iii. Compute $\llbracket \mathbf{z}' \rrbracket = \llbracket \mathbf{z}_1 + \sum_{k=2}^{\tau_2} \epsilon_k * \mathbf{x}_k + \mathbf{z}_k \rrbracket = \llbracket \mathbf{x}' * \mathbf{y}' \rrbracket$.
 - (b) **Combine on \mathbf{y} :** The parties assign the triples uniformly into T buckets of τ_2 triples each (as before, this is done using \mathcal{F}_{CT}). For every bucket, denote the triples it contain by $\{(\llbracket \mathbf{x}_k \rrbracket, \llbracket \mathbf{y}_k \rrbracket, \llbracket \mathbf{z}_k \rrbracket)\}_{k \in [\tau_2]}$ the parties do as follows:
 - i. Compute $\llbracket \mathbf{y}' \rrbracket = \llbracket \sum_{k \in [\tau_2]} \mathbf{y}_k \rrbracket$ and $\llbracket \mathbf{x}' \rrbracket = \llbracket \mathbf{x}_1 \rrbracket$
 - ii. Compute $\llbracket \epsilon_k \rrbracket = \llbracket \mathbf{x}_1 - \mathbf{x}_k \rrbracket$ and open ϵ_k for every $k = \{2, \dots, \tau_2\}$.
 - iii. Compute $\llbracket \mathbf{z}' \rrbracket = \llbracket \mathbf{z}_1 + \sum_{k=2}^{\tau_2} \epsilon_k * \mathbf{y}_k + \mathbf{z}_k \rrbracket = \llbracket \mathbf{x}' * \mathbf{y}' \rrbracket$.

Fig. 13. Protocol $\Pi_{\text{AHCOM-F}^m}$ - Part 2

Theorem 4.5. *The method **ReOrg** in $\Pi_{\text{AHCOM-}\mathbb{F}^m}$ of Fig. 12 UC-securely implements the method **ReOrg** in functionality $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ of Fig. 10 in the $\mathcal{F}_{\text{OT-}}$, $\mathcal{F}_{\text{EQ-}}$ and $\mathcal{F}_{\text{CT-}}$ -hybrid model against a static and malicious adversary corrupting a majority of the parties.*

5 Protocol for Multiparty Computation

In Fig. 14 we show how to realize a fully fledged arithmetic MPC protocol secure against a static and malicious adversary, with the possibility of corrupting a majority of the parties. This protocol is very similar to the one used in MiniMAC [16] and thus we will not dwell on its details.

Init: The parties invoke (**init**) followed by (**commit**, I) on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ to get a sufficient amount of raw commitments. Next the parties call (**mult**, \cdot) and (**reOrg**, ϕ , \cdot) to get a sufficient amount of multiplication triples, $(\llbracket \mathbf{x} \rrbracket, \llbracket \mathbf{y} \rrbracket, \llbracket \mathbf{z} \rrbracket)$ and reorganization pairs $(\llbracket \mathbf{x} \rrbracket, \llbracket \phi(\mathbf{x}) \rrbracket)$.

Input: To share P_i 's input $\mathbf{y} \in \mathbb{F}^m$, party P_i calls (**Input**, i, k, \mathbf{y}) on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ with k being the identifier of a raw commitment. All other parties P_j call (**Input**, j, k). The parties obtain commitment $\llbracket \mathbf{y}_k \rrbracket$.

Rand: All parties call (**random**, k) on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ with k being an identifier of a raw commitment. The parties obtain commitment $\llbracket \mathbf{x}_k \rrbracket$.

Public Add: To add together a public value \mathbf{y} and a commitment, $\llbracket \mathbf{x} \rrbracket$, the parties simply compute $\mathbf{y} + \llbracket \mathbf{x} \rrbracket = \llbracket \mathbf{y} + \mathbf{x} \rrbracket$ using the **Linear** command on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Add: To add two commitments together, $\llbracket \mathbf{x} \rrbracket$ and $\llbracket \mathbf{y} \rrbracket$ the parties simply compute $\llbracket \mathbf{x} \rrbracket + \llbracket \mathbf{y} \rrbracket = \llbracket \mathbf{x} + \mathbf{y} \rrbracket$ using the **Linear** command on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Public Multiply: To multiply together a public value \mathbf{y} and a commitment, $\llbracket \mathbf{x} \rrbracket$, the parties simply compute $\mathbf{y} * \llbracket \mathbf{x} \rrbracket = \llbracket \mathbf{y} * \mathbf{x} \rrbracket$ using the **Linear** command on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Multiply: To multiply together two commitments, $\llbracket \mathbf{x} \rrbracket$ and $\llbracket \mathbf{y} \rrbracket$, the parties select a preprocessed multiplication triple $(\llbracket \mathbf{a} \rrbracket, \llbracket \mathbf{b} \rrbracket, \llbracket \mathbf{c} \rrbracket)$ and proceed as follows:

1. The parties open $\epsilon = \llbracket \mathbf{x} \rrbracket - \llbracket \mathbf{a} \rrbracket$ and $\rho = \llbracket \mathbf{y} \rrbracket - \llbracket \mathbf{b} \rrbracket$ using the commands **Linear** and **Open** on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.
2. The parties compute $\llbracket \mathbf{z} \rrbracket = \llbracket \mathbf{x} * \mathbf{y} \rrbracket = \llbracket \mathbf{c} \rrbracket + \epsilon * \llbracket \mathbf{b} \rrbracket + \rho * \llbracket \mathbf{a} \rrbracket + \epsilon * \rho$ using the command **Linear** on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Reorganize: To apply a linear operator ϕ to commitment $\llbracket \mathbf{x} \rrbracket$ the parties select a preprocessed reorganization pair $(\llbracket \mathbf{a} \rrbracket, \llbracket \phi(\mathbf{a}) \rrbracket)$. They then proceed as follows:

1. The parties open $\epsilon = \llbracket \mathbf{x} \rrbracket - \llbracket \mathbf{a} \rrbracket$ using the commands **Linear** and **Open** on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.
2. The parties then compute $\llbracket \phi(\mathbf{x}) \rrbracket = \llbracket \phi(\mathbf{a}) \rrbracket + \phi(\epsilon)$ using the commands **Linear** on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Output: The parties open the value $\llbracket \mathbf{x} \rrbracket$ that should be output of the computation using the command **Open** on $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$.

Fig. 14. Protocol UC-realizing $\mathcal{F}_{\text{MPC-}\mathbb{F}^m}$ in the $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ model.

We prove the following theorem in the full version [20]:

Theorem 5.1. *The protocol in Fig. 14 UC-securely implements the functionality $\mathcal{F}_{\text{MPC-}\mathbb{F}^m}$ of Fig. 10 in the $\mathcal{F}_{\text{AHCOM-}\mathbb{F}^m}$ -hybrid model against a static and malicious adversary corrupting a majority of the parties.*

6 Efficiency

Practical Optimizations. Several significant optimizations can be applied to our protocol. We chose to describe the optimizations here rather than earlier for the ease of presentation. In the following we present each of the optimizations and sketch out its security.

Using less storage. As we mentioned before, the two-party homomorphic commitment scheme of [18] can be used as an implementation of functionality $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$. Briefly, in this two party commitment scheme the committer holds a set of $2m$ vectors from \mathbb{F}^γ , namely the vectors $\bar{\mathbf{s}}_1^0, \bar{\mathbf{s}}_1^1, \dots, \bar{\mathbf{s}}_m^0, \bar{\mathbf{s}}_m^1$ whereas the receiver choose a set of m bits b_1, \dots, b_m , denoted as “its choice of watch bits” and obtains the m vectors $\bar{\mathbf{s}}_1^{b_1}, \dots, \bar{\mathbf{s}}_m^{b_m}$, denoted as “the watchbits”.

Recall that in our multiparty homomorphic commitment scheme party P_i participates as a receiver in $p - 1$ instances of two-party commitment scheme with all other parties. This means that P_i needs to remember its choice of watchbits for every other party and this accordingly for every linear operation that is performed over the commitments. For instance, let $[\mathbf{x}], [\mathbf{y}]$ be two multiparty commitments between three parties, then party P_1 stores $[\mathbf{x}]^1 = \{\{[\mathbf{x}^2]^{2,1}, [\mathbf{x}^2]^{3,1}\}, \{(\mathbf{x}^1)^{1,2}, (\mathbf{x}^1)^{1,3}\}\}$. To perform the operation $[\mathbf{x}] + [\mathbf{y}]$ then P_1 end up with

$$[\mathbf{x} + \mathbf{y}]^1 = \{\{[\mathbf{x}^2]^{2,1} + [\mathbf{y}^2]^{2,1}, [\mathbf{x}^2]^{3,1} + [\mathbf{y}^2]^{3,1}\}, \{(\mathbf{x}^1)^{1,2} + (\mathbf{y}^1)^{1,2}, (\mathbf{x}^1)^{1,3} + (\mathbf{y}^1)^{1,3}\}\}$$

To make it more efficient, P_i can choose the bits b_1, \dots, b_m only once and use them in *all* instances of two-party commitments. This makes the process of linear operations over commitments simpler and does not requires from P_1 to store the commitments for $p - 1$ parties. Applying the optimization to the above example, we have that P_1 stores only a single value for the commitment part, that is, now P_1 needs to store

$$[\mathbf{x} + \mathbf{y}]^1 = \{[\mathbf{x}^2]^{2,1} + [\mathbf{y}^2]^{2,1} + [\mathbf{x}^2]^{3,1} + [\mathbf{y}^2]^{3,1}, \{(\mathbf{x}^1)^{1,2} + (\mathbf{y}^1)^{1,2}, (\mathbf{x}^1)^{1,3} + (\mathbf{y}^1)^{1,3}\}\}$$

Security follows from the underlying commitment scheme, since what we now do is simply equivalent to storing a sum of commitments in a single instance of the two-party scheme.

In a bit more detail, we see that since $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$ is UC-secure, it is secure under composition. Furthermore, considering the worst case where only a single party is honest and all other parties are malicious and colluding we then notice that the above optimization is equivalent to executing $p - 1$ instances of the $\mathcal{F}_{2\text{HCOM-}\mathbb{F}^m}$, but where the same watchbits are chosen by the honest party. We see that this is almost the same as calling **Commit** p times. The only exception is that the seeds of the committing party, P_i , of the calls to \mathcal{F}_{OT} are different in our optimized protocol. Thus it is equivalent to the adversary being able to select p potentially different seeds to the calls to **Commit**. However, the output of the PRG calls are indistinguishable from random in both cases and so the distributions in both cases are indistinguishable assuming p is polynomial in the security parameter.

Optimized CorrectnessTest. Recall that in the sacrificing step of protocol $\Pi_{\text{AHCOM-}\mathbb{F}^m}$ (see Fig. 13) the parties perform two openings of commitments for every bucket (the opening is described as part of the **CorrectnessTest** in Fig. 11). That is, beginning the step with $\tau_1 \cdot (\tau_2)^2 \cdot T$ triples (which are assigned to $(\tau_2)^2 \cdot T$ buckets) leads to the opening of $(\tau_1 - 1) \cdot (\tau_2)^2 \cdot T$ triples.

Since we require that the results of all of these openings be $\mathbf{0}$, then any linear combination over these opening would be $\mathbf{0}$ as well if they are correct. On the other hand, if one or more of the openings are not zero the result of a linear combination over the openings might be $\mathbf{0}$ with probability $\frac{1}{|\mathbb{F}|}$. Thus, agreeing on a s random linear combinations over the openings would detect an incorrect triple with overwhelming probability.

Optimized opening. In the online phase of our protocol, for every multiplication gate the parties need to open some random commitments using the **Open** command. The implementation of the **Open** command requires interaction between every pair of parties, thus, the communication complexity is $\Omega(T \cdot p^2)$ where T is the number of multiplication gates in the circuit. Following the same idea as used in SPDZ and MiniMAC, we note that we can reduce the communication complexity for every gate to $O(p)$ in the following way, to perform a “partial opening” of a commitment $[\mathbf{x}]$: First, every party P_i sends its share \mathbf{x}^i to P_1 . Then P_1 computes $\mathbf{x} = \sum_{j \in [p]} \mathbf{x}^j$ and sends back \mathbf{x} to everyone. This incurs a communication complexity of $O(p)$ rather than $O(p^2)$. In the end of the evaluation of the circuit, the parties perform s random linear combinations over the commitment values that were “partially opened” earlier. Then, they open the results of the linear combinations using the **Open** command. If one of the opened results with a wrong value (i.e. that does not conform with the result of the linear combination of the values sent from P_1 in the partial opening) then the parties abort.

Using this optimization leads to a communication complexity of $\Omega(T \cdot p + s \cdot p^2)$. Security follows by the same arguments as used in SPDZ and MiniMAC. Particularly before opening the output nothing gets leaked during the execution of the gates in the protocol and since the adversary does not know the random linear combinations he cannot send manipulated values that pass this check.

Optimizing for large fields. If the field we compute in contains at least 2^s elements, then the construction of multiplication triples becomes much lighter. First see that in this case it is sufficient to only have two triples per bucket for sacrificing. This is because the adversary’s success probability of getting an incorrect triple through the **CorrectnessTest** in Fig. 11 is less than $|\mathbb{F}|^{-1} \leq 2^{-s}$. Next we see that it is possible to eliminate the combining step on the \mathbf{y} components of the triples. This follows since the party inputting x into the **ArithmeticOT** procedure in Fig. 6 can now only succeed in a selective failure attack on the honest party’s input y if he manages to guess y . To see this notice that if the adversary changes the q ’th bit of his input x then the result of the computation will be different from the correct result with a factor $y \cdot 2^{q-1}$. But since y is in a field of at least 2^s elements then $y \cdot 2^{i-1} = 0$ with probability at most 2^{-s} and thus its cheating

attempt will be caught in the `CorrectnessTest` with overwhelming probability. Furthermore the combining on \mathbf{x} is now also overly conservative in the bucket size τ_2 . To see this notice that the adversary only gets to learn at most $s - 1$ bits in total over all triples. This means that it cannot fully learn the value of a component of \mathbf{x} for all triples in the bucket (since it is at least s bits long), which is what our proof, bounding his success probability assumes. Instead we can now bound its success probability by considering a different attack vectors and using the Leftover Hash Lemma to compute the maximum amount of leakage it can learn when combining less than τ_2 triples in a bucket as done in [29]. However, we leave the details of this as future work. To conclude, even when using the very conservative bound on bucket size, we get that it now takes only $6m \log(|\mathbb{F}|)$ OTs, amortized, when constructing 2^{21} triples instead of $27m \log(|\mathbb{F}|)$ when $s = 40$.

Efficiency Comparison. The computationally heavy parts in our protocol are the usage of oblivious transfers and the use of the underlying homomorphic two-party commitments. Both of these are rather efficient in practice having the state-of-the-art constructions of Keller *et al.* ([28] for OT) and of Frederiksen *et al.* ([18], for two-party homomorphic commitments). It should be noted that if one wish to use a binary field, or another small field, then it is necessary to use a code based on algebraic geometry internally if using the commitment scheme of Frederiksen *et al.* [18]. These are however not as efficient to compute as, for example, the BCH code used in the implementation of [18] done in [35].

Table 2. Comparison of the overhead of OTs needed, in the amortized sense. All values should be multiplied with $p(p - 1)$ to get the true number of needed OTs. We differentiate between regular OTs and the more efficient correlated random OT with error (COTe) [29]. We assume that the computational security parameter $\kappa \geq m \log(|\mathbb{F}|)$ some complexities increase. $\mathbb{F} = GF(2)$. For [7, 29] $m = 1$ is possible. We assume at least 2^{21} triples are generated which gives the smallest numbers to the protocols. *) Using optimization 4. in Sect. 6, requiring $|\mathbb{F}| \geq 2^s$.

Scheme	Finite field	Rand, Input COTe	Schur, ReOrg COTe	Mult	
				COTe	OT
[19]	\mathbb{F}_{2^c} for $c \geq 1$	$m \log(\mathbb{F})$	$m \log(\mathbb{F})$	$24m \log(\mathbb{F})$	$12m \log(\mathbb{F}) + 6s$
[29]	\mathbb{F}_{2^c} for $c \geq 2s$	$m \log(\mathbb{F})$	-	$5m \log(\mathbb{F})$	$3m \log(\mathbb{F})$
[7]	\mathbb{F}_2	$m \log(\mathbb{F})$	-	$12m \log(\mathbb{F})$	$3m \log(\mathbb{F})$
This work	Any	0	0	0	$27m \log(\mathbb{F})$
This work*	\mathbb{F}_{2^c} for $c \geq s$	0	0	0	$6m \log(\mathbb{F})$

Notice that the amount of OTs our protocol require is a factor of $O(m \log(|\mathbb{F}|))$ greater than the amount of commitments it require. Therefore, in Table 2 we try to compare our protocol with [7, 19, 29] purely based on the amount of OTs needed. This gives a fair estimation on the efficiency of our protocol compared to the current state-of-the-art protocols for the same settings (static, malicious majority in the secret sharing approach).

Furthermore, we note that both [19, 29] (which is used as the underlying pre-processing phase for MiniMAC) require a factor of between $O(m)$ and $O(m^2)$ more coin tosses than our protocol. The reason for this is that in our protocol it is sufficient to perform the random linear combinations using a random *scalar* from \mathbb{F} (i.e. scalar multiplication) whereas [19, 29] requires a component-wise multiplication using a random *vector* from \mathbb{F}^m . Note that in the comparison in Table 2 we adjusted the complexity of [19] to fit what is needed to securely fix the issue regarding the sacrificing, which we present in the full version [20].

7 Applications

Practically all maliciously secure MPC protocols require some form of commitments. Some, e.g. the LEGO family of protocols [17, 18, 34, 35], also require these commitments to be additively homomorphic. Our MPC protocol works directly on such commitments, we believe it makes it possible to use our protocol as a component in a greater scheme with small overhead, as all private values are already committed to. Below we consider one such specific case; when constructing committed OT from a general MPC protocol.

7.1 Bit Committed OT

The bit-OT two-party functionality $(b, x_0, x_1) \mapsto (x_b, \perp)$ can be realized using a secure evaluation of a circuit containing a single AND gate and two XOR gates: Let b denote the choice bit and x_0, x_1 the bit messages, then $x_b = b \wedge (x_0 \oplus x_1) \oplus x_0$.

We notice that all shares in our protocol are based on two-party commitments. This means that constructing a circuit similar to the description above will compute OT, based on shares which are committed to. Thus we can efficiently realize an OT functionality working on commitments. Basically we use $\mathbb{F} = \text{GF}(2)$ and compute a circuit with one layer of AND gates computing the functionality above. In the end we only open towards the receiver. At any later point in time it is possible for the sender to open the commitments to x_0 and x_1 , no matter what the receiver chose. The sender can also open b towards the receiver. However we notice that we generally need to open m committed OTs at a time (since we have m components in a message). However, if this is not possible in the given application we can use reorganization pairs to open only specific OTs, by simply branching each output message (consisting of m components) into m output messages each of which only opening a single component, and thus only a single actual OT.

Furthermore, since we are in the two-party setting, and because of the specific topology of the circuit we do not need to have each multiparty commitment be the sum of commitments between each pair of parties. Instead the receiving party simply commits to b towards the sending party using a two-party commitment. Similarly the sending party commits to x_0 and x_1 towards the receiving party using a two-party commitment. Now, when they construct a multiplication triple they only need to do one OT per committed OT they construct; the receiver

inputting his b and the receiver inputting $x_0 \oplus x_1$. Since the sender should not learn anything computed by the circuit the parties do not need to complete the arithmetic OT in other direction.

In this setting we have $\mathbb{F} = \text{GF}(2)$ (hence $m \geq s$), $p = 2$ and 1 multiplication gate when constructing a batch of m committed OTs. Plugging these into the equations in Table 1 we see that the amortized cost for a single committed-OT is 36 regular string OTs of κ bits and $108/m \leq 108/s \leq 3$ (for $s = 40$) commitments for batches of m committed-OTs.

It is also possible to achieve committed OT using other MPC protocols, in particular the TinyOT protocols [7, 33] have a notion of committed OT as part of its internal construction. However our construction is quite different.

Acknowledgment. The authors would like to thank Carsten Baum and Yehuda Lindell for useful discussions along Peter Scholl and Marcel Keller for valuable feedback and discussions in relation to their SPDZ and MiniMAC preprocessing papers.

References


1. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: ACM CCS, pp. 535–548 (2013)
2. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_26
3. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
4. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: STOC, pp. 479–488 (1996)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
6. Brandão, L.T.A.N.: Very-efficient simulatable flipping of many coins into a well (and a new universally-composable commitment scheme). In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 297–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_12
7. Burra, S.S., Larraia, E., Nielsen, J.B., Nordholt, P.S., Orlandi, C., Orsini, E., Scholl, P., Smart, N.P.: High performance multi-party computation for binary circuits based on oblivious transfer. IACR Cryptology ePrint Archive, 2015:472 (2015)
8. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
9. Cascudo, I., Damgård, I., David, B., Döttling, N., Nielsen, J.B.: Rate-1, linear time and additively homomorphic UC commitments. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 179–207. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_7

10. Cascudo, I., Damgård, I., David, B., Giacomelli, I., Nielsen, J.B., Trifiletti, R.: Additively homomorphic UC commitments with optimal amortized overhead. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 495–515. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46447-2_22
11. Damgård, I., David, B., Giacomelli, I., Nielsen, J.B.: Compact VSS and efficient homomorphic UC commitments. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 213–232. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_12
12. Damgård, I., Lauritsen, R., Toft, T.: An empirical study and some improvements of the MiniMac protocol for secure computation. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 398–415. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_23
13. Damgård, I., Orlandi, C.: Multiparty computation for dishonest majority: from passive to active security at low cost. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 558–576. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_30
14. 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
15. Damgård, I., Zakarias, R.: Fast oblivious AES a dedicated application of the MiniMac protocol. In: Pointcheval, D., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2016. LNCS, vol. 9646, pp. 245–264. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31517-1_13
16. Damgård, I., Zakarias, S.: Constant-overhead secure computation of Boolean circuits using preprocessing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_35
17. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: MiniLEGO: efficient secure two-party computation from general assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 537–556. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_32
18. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Trifiletti, R.: On the complexity of additively homomorphic UC commitments. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 542–565. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_23
19. Frederiksen, T.K., Keller, M., Orsini, E., Scholl, P.: A unified approach to MPC with preprocessing using OT. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 711–735. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_29
20. Frederiksen, T.K., Pinkas, B., Yanai, A.: Committed MPC - maliciously secure multiparty computation from homomorphic commitments. IACR Cryptology ePrint Archive, 2017:550 (2017)
21. Furukawa, J., Lindell, Y., Nof, A., Weinstein, O.: High-throughput secure three-party computation for malicious adversaries and an honest majority. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 225–255. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_8
22. Garay, J.A., Ishai, Y., Kumaresan, R., Wee, H.: On the complexity of UC commitments. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 677–694. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_37

23. Gilboa, N.: Two party RSA key generation. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 116–129. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_8
24. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
25. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
26. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC, pp. 21–30 (2007)
27. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_18
28. Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 724–741. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_35
29. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: ACM CCS, pp. 830–842 (2016)
30. Larraia, E., Orsini, E., Smart, N.P.: Dishonest majority multi-party computation for binary circuits. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 495–512. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_28
31. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_4
32. Lindell, Y., Pinkas, B., Smart, N.P., Yanai, A.: Efficient constant round multi-party computation combining BMR and SPDZ. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 319–338. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_16
33. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_40
34. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_22
35. Nielsen, J.B., Schneider, T., Trifiletti, R.: Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In: NDSS (2017)
36. Rindal, P., Trifiletti, R.: SplitCommit: implementing and analyzing homomorphic UC commitments. IACR Cryptology ePrint Archive, 2017:407 (2017)
37. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)



Fast Garbling of Circuits over 3-Valued Logic

Yehuda Lindell and Avishay Yanai^(✉) 

Bar-Ilan University, Ramat Gan, Israel
yehuda.lindell@biu.ac.il, ay.yanay@gmail.com

Abstract. In the setting of secure computation, a set of parties wish to compute a joint function of their private inputs without revealing anything but the output. Garbled circuits, first introduced by Yao, are a central tool in the construction of protocols for secure two-party computation (and other tasks like secure outsourced computation), and are the fastest known method for constant-round protocols. In this paper, we initiate a study of garbling multivalent-logic circuits, which are circuits whose wires may carry values from some finite/infinite set of values (rather than only **True** and **False**). In particular, we focus on the three-valued logic system of Kleene, in which the admissible values are **True**, **False**, and **Unknown**. This logic system is used in practice in SQL where some of the values may be missing. Thus, efficient constant-round secure computation of SQL over a distributed database requires the ability to efficiently garble circuits over 3-valued logic. However, as we show, the two natural (naive) methods of garbling 3-valued logic are very expensive.

In this paper, we present a general approach for garbling three-valued logic, which is based on first encoding the 3-value logic into Boolean logic, then using standard garbling techniques, and final decoding back into 3-value logic. Interestingly, we find that the specific encoding chosen can have a significant impact on efficiency. Accordingly, the aim is to find Boolean encodings of 3-value logic that enable efficient Boolean garbling (i.e., minimize the number of AND gates). We also show that Boolean AND gates can be garbled at the same cost of garbling XOR gates in the 3-value logic setting. Thus, it is unlikely that an analogue of free-XOR exists for 3-value logic garbling (since this would imply free-AND in the Boolean setting).

1 Introduction

1.1 Background – Three-Valued Logic

In classical (Boolean) propositional logic, statements are assigned a “truth-value” that can be either **True** or **False**, but not both. Logical operators are

Y. Lindell and A. Yanai—Supported by the European Research Council under the ERC consolidators grant agreement no. 615172 (HIPS) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office.

used to make up a complex statement out of other, one or more, simpler statements such that the truth value of the complex statement is derived from the simpler ones and the logical operators that connects them. For instance, given that the statement A is True and the statement B is True we infer that the statement $C = "A \text{ and } B"$ (denoted by $C = A \wedge B$) is True as well.

Another branch of propositional logic is the *multivalent logic system*. Multivalent logic systems consider more than two truth-values, that is, they may admit anything from three to an infinite number of possible truth-values. Among those, the simplest and most studied sort is the *three-valued logic* (or ternary logic), which is a system that admits three truth-values, e.g., "truth", "falsity" and "indeterminacy". Such a system seems to suit many real life situations, for instance, statements about the future or paradoxical statements like "*this statement is not correct*", which must have an indeterminate truth-value. Note that in different applications, the third truth-value could be interpreted differently, hence, different inference rules are derived¹. The most common three-valued logic system is Kleene's Logic [6], in which statements are assigned with either True, False or Unknown. For clarity, whenever we use the term three-valued logic or 3VL we actually refer to Kleene's Logic. We remark that although other three-valued logic system exist, in this paper we focus only on Kleene's logic since its use in real life is the most prevalent; see the application example in Sect. 1.2.

The admission of Unknown requires one to expand the set of inference rules, to enable the computation of the truth-value of a complex statement from simpler statements, even if one or more of them are Unknown. In Kleene's logic, the inference process complies with the way we usually make conclusions: It yields Unknown whenever at least one statement that is necessary for deciding True or False is assigned with Unknown. For example, the AND of True and Unknown is Unknown since if the Unknown were False then the result would be false. However, the OR of True and Unknown is True since it equals True irrespective of the Unknown variable's value.

The 3VL inference rules are presented in Table 1 in the form of truth tables. In the rest of the paper whenever we refer to the Boolean version of AND, OR,

Table 1. Definition of the functions \wedge_3 (AND), \vee_3 (OR), \oplus_3 (XOR) and \neg_3 (NOT) using truth tables. Note these functions are symmetric, that is, the order of the inputs makes no difference.

\wedge_3	\vee_3	\oplus_3	\neg_3
$T \mid T \mid U \mid F$	$T \mid T \mid U \mid F$	$T \mid T \mid U \mid F$	$T \mid F$
$U \mid U \mid U \mid F$	$U \mid T \mid U \mid U$	$U \mid U \mid U \mid U$	$U \mid U$
$F \mid F \mid F \mid F$	$F \mid T \mid U \mid F$	$F \mid T \mid U \mid F$	$F \mid T$

¹ In fact, even the two traditional truth-values True and False could have other meaning in different three-valued logic systems.

XOR and NOT we use the usual notation $\wedge, \vee, \oplus, \neg$ and when we use their 3VL version we subscript it with the number 3, e.g. $\wedge_3, \vee_3, \oplus_3, \neg_3$. We denote by T, F and U , the 3VL values True, False and Unknown, respectively.

1.2 Applications in SQL

In SQL specifications [5] the NULL marker indicates the absence of a value, or alternatively, that the value is neither True nor False, but Unknown. Because of this, comparisons with NULL never result in either True or False, but always in the third logical value: Unknown. For example, the statement “SELECT 10 = NULL” results in Unknown. However, certain operations on Unknown can return values if the absent value is not relevant to the outcome of the operation. Consider the following example:

```
SELECT * FROM T1 WHERE (age > 30 OR height < 140) AND weight > 110
```

Now, consider an entry where the person’s age is missing. In this case, if the person’s height is 150 then the OR subexpression evaluates to Unknown and so the entire result is Unknown, hence, this entry is not retrieved. In contrast, if the person’s height is 120, then the OR subexpression evaluates to True, and so the result is True if `weight > 110`, and False if `weight ≤ 110`.

We remark that the main SQL implementations [2, 9, 10] (Oracle, Microsoft and MySQL) conform to the Kleene’s three-valued logic described above. As such, if secure computation is to be used to carry out secure SQL on distributed (or shared) databases, then efficient solutions for dealing with three-valued logic need to be developed.

1.3 Naively Garbling a 3VL Gate

We begin by describing the straightforward (naive) approach to garbling a 3VL gate. Let g_3 be a 3VL gate with input wires x, y and output wire z , where each wire takes one of 3 values, denoted T, F and U . The basic garbling scheme of Yao [8, 13] works by associating a random key with each possible value on each wire, and then encrypting each possible output value under all combinations of input values that map to that output value. Specifically, for each wire $\alpha \in \{x, y, z\}$, choose random keys $k_\alpha^T, k_\alpha^F, k_\alpha^U$. Then, for every combination of $\beta_x, \beta_y \in \{T, F, U\}$, encrypt $k_z^{g(\beta_x, \beta_y)}$ using keys $k_x^{\beta_x}, k_y^{\beta_y}$ and define the garbled table to be a random permutation of the ciphertexts. See Fig. 1 for a definition of such a garbled gate.²

² Note that in a two-party protocol like Yao’s, the parties then run 1-out-of-3 oblivious transfers in order for the evaluator to learn the keys that are associated with its input.

This approach yields a garbled gate of 9 entries. Using the standard garbled row reduction technique [11], it is possible to reduce the size of the gate to 8 entries. This means that 8 ciphertexts need to be communicated for each gate in the circuit. However, this garbling scheme requires *four times* more bandwidth for three-valued logic gates than the state-of-the-art for their Boolean \wedge counterparts [12]. Furthermore, using the free-XOR paradigm [7] (as is also utilized in [12]), XOR gates are free in the Boolean case but require significant bandwidth and computation in the three-valued logic case. (We remark that [7, 12] do require non-standard assumptions; however, these techniques do not translate to the 3VL case and so cannot be used, even under these assumptions.)

Before proceeding, we note that another natural way of working is to translate each variable in the 3VL circuit into two Boolean variables: the first variable takes values T, F (true/false), and the second variable takes values K, U (known/unknown). This method fits into our general paradigm for solving the problem and so will be described later; as we will show, this specific method is not very efficient.

1.4 Our Results

The aim of this paper is to find ways of garbling three-valued logic functions that are significantly more efficient than the naive method described in Sect. 1.3. Our methods all involve first encoding a 3VL function as a Boolean function and then utilizing the state-of-the-art garbling schemes for Boolean functions. These schemes have the property that AND gates are garbled using two ciphertexts, and XOR gates are garbled for free [7, 12]. Thus, our aim is to find Boolean encodings of 3VL functions that can be computed using few AND gates (and potentially many XOR gates).

In order to achieve our aim, we begin by formalizing the notion of a 3VL-Boolean encoding which includes a way of encoding 3VL-input into Boolean values, and a way of computing the 3VL function using a Boolean circuit applied to the encoded input. Such an encoding reduces the problem of evaluating 3VL functions to the problem of evaluating Boolean functions. Our formalization is general, and can be used to model other multivalent logic systems, like that used in fuzzy logic.

Next, we construct *efficient* 3VL-Boolean encodings, where by efficient, we mean encodings that can be computed using few Boolean AND gates. Interestingly, we show that the way that 3VL-variables are encoded as Boolean variables has a great influence on the efficiency of the Boolean computation. We describe three different encodings: The first encoding is the natural one, and it works by

1	$E_{k_x^T}$	$E_{k_y^T}$	$k_z^{g(T,T)}$
2	$E_{k_x^T}$	$E_{k_y^F}$	$k_z^{g(T,F)}$
3	$E_{k_x^T}$	$E_{k_y^U}$	$k_z^{g(T,U)}$
4	$E_{k_x^F}$	$E_{k_y^T}$	$k_z^{g(F,T)}$
5	$E_{k_x^F}$	$E_{k_y^F}$	$k_z^{g(F,F)}$
6	$E_{k_x^F}$	$E_{k_y^U}$	$k_z^{g(F,U)}$
7	$E_{k_x^U}$	$E_{k_y^T}$	$k_z^{g(U,T)}$
8	$E_{k_x^U}$	$E_{k_y^F}$	$k_z^{g(U,F)}$
9	$E_{k_x^U}$	$E_{k_y^U}$	$k_z^{g(U,U)}$

Fig. 1. Garbling a 3VL gate directly using 9 rows.

defining two Boolean variables x_T and x_U for every 3VL-variable x such that $x_U = 1$ if and only if $x = U$, and $x_T = 1$ if $x = T$ and $x_T = 0$ if $x = F$. This is “natural” in the sense that one Boolean variable is used to indicate whether the 3VL-value is known or not, and the other variable is used to indicate whether the 3VL-value is true or false in the case that it is known. We show that under this encoding, 3VL-AND gates can be computed at the cost of 6 Boolean AND gates, and 3VL-XOR gates can be computed at the cost of 1 Boolean AND gate. We then proceed to present two alternative encodings; the first achieves a cost of 4 Boolean AND gates for every 3VL-AND gate and 1 Boolean AND gate for every 3VL-XOR gate, whereas the second achieves a cost of 2 Boolean AND gates both for every 3VL-AND gate and every 3VL-XOR gate. These encodings differ in their cost tradeoff, and the choice of which to use depends on the number of AND gates and XOR gates in the 3VL-circuit.

Given these encodings, we show how *any* protocol for securely computing Boolean circuits, for semi-honest or malicious adversaries, can be used to securely compute 3VL circuits, at almost the same cost. Our construction is black-box in the underlying protocol, and is very simple.

Finally, observe that all our encodings have the property that 3VL-XOR gates are computed using at least 1 Boolean AND-gate. This means that none of our encodings enjoy the free-XOR optimization [7] which is extremely important in practice. We show that this is actually somewhat inherent. In particular, we show that it is possible to garble a Boolean AND gate at the same cost of garbling a 3VL XOR gate. Thus, free-3VL-XOR would imply free-Boolean-AND, which would be a breakthrough for Boolean garbling. Formally, we show that free-3VL-XOR is *impossible* in the linear garbling model of [12].

Brute-force search for encodings. It is theoretically possible to search for efficient 3VL-Boolean encodings by simply trying all functions with a small number of AND gates, for every possible encoding. Indeed, for up to *one* AND gate it is possible since the search space is approximately 2^{20} possibilities. However, if up to *two* AND gates are allowed, then the search space already exceeds 2^{50} possibilities. We ran a brute-force search for up to one AND gate, and rediscovered our 3VL-XOR computation that uses a single AND gate (in fact, we found multiple ways of doing this). However, our search showed that there does *not* exist a way of computing 3VL-AND using a single AND gate, for *any* encoding. See Appendix A for more details on the brute-force search algorithm that we used.

2 Encoding 3VL Functions as Boolean Functions

2.1 Notation

We denote by T, V, U the 3VL values True, False and Unknown, respectively, and by 1, 0 the Boolean values True and False. We denote by F_3 the set of all 3VL functions (i.e. all functions of the form $\{T, F, U\}^* \rightarrow \{T, F, U\}^*$) and by F_2 be the set of all Boolean functions (i.e. all functions of the form $\{0, 1\}^* \rightarrow \{0, 1\}^*$).

In addition, we denote by $F_3(\ell, m)$ and $F_2(\ell, m)$ the set of all 3VL and Boolean functions, respectively, that are given ℓ inputs and produce m outputs. We denote by x_i the i th element in x both for $x \in \{T, F, U\}^*$ and $x \in \{0, 1\}^*$.

2.2 3VL-Boolean Encoding

As we have mentioned, in order to utilize the efficiency of modern garbling techniques, we reduce the problem of garbling 3VL circuits to the problem of garbling Boolean circuits, by encoding 3VL functions as Boolean functions. Informally speaking, a 3VL-Boolean encoding is a way of mapping 3VL inputs into Boolean inputs, computing a Boolean function on the mapped inputs, and mapping the Boolean outputs back to a 3VL output. This method is depicted in Fig. 2. The naive approach appears on the left and involves directly garbling a 3VL circuit, as described in Sect. 1.3. Our approach appears on the right and works by applying a transformation $\text{Tr}_{3 \rightarrow 2}$ to map the 3VL input to a Boolean input, then computing an appropriately defined Boolean function, and finally applying a transformation $\text{Tr}_{2 \rightarrow 3}$ to map the output back. The Boolean function is also defined by a transformation, so that a 3VL function f_3 is transformed to a Boolean function f_2 via the transformation Tr_F , that is, $f_2 = \text{Tr}_F(f_3)$, and this is what is computed. As such, as we will see, it suffices to garble the Boolean function f_2 , and if this function has few AND gates then it will be efficient for this purpose.

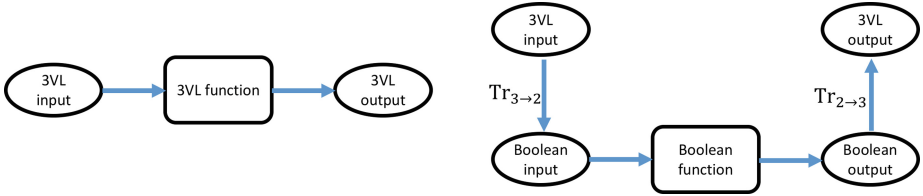


Fig. 2. Naive approach on the left side and our new approach on the right side.

Observe that since we map inputs from three-valued logic to Boolean logic, the set sizes of all possible inputs are different. Thus, we define encodings via *relations* and not via bijective mappings. Of course, the actual transformations $\text{Tr}_{3 \rightarrow 2}$ and $\text{Tr}_{2 \rightarrow 3}$ are functions. However, the mapping between inputs and outputs may be expressed as relations; e.g., when mapping a single 3VL variable to two Boolean variables, it may be the case that one of the 3VL variables can be expressed as two possible Boolean pairs. This enables more generality, and can help in computation, as we will see below.

Although one could define a very general encoding from 3VL to Boolean values, we will specifically consider encodings that map every single 3VL variable to exactly two Boolean variables. We consider this specific case since it simplifies our definitions, and all our encodings have this property.

The formal definition. Let $f : \{T, F, U\}^m \rightarrow \{T, F, U\}^n$ be a 3VL function. We begin by defining the appropriate relations and transformations.

1. A **value encoding** is a relation $R_{3 \rightarrow 2} \subseteq \{T, F, U\} \times \{0, 1\}^2$ that is left-total and injective.³ For $\ell \in \mathbb{N}$, let $R_{3 \rightarrow 2}^\ell \subseteq \{T, F, U\}^\ell \times \{0, 1\}^{2\ell}$ be the relation defined by extending $R_{3 \rightarrow 2}$ per coordinate.⁴
2. A **valid input transformation** is an injective function $\text{Tr}_{3 \rightarrow 2}^m : \{T, F, U\}^m \rightarrow \{0, 1\}^{2m}$ such that $\text{Tr}_{3 \rightarrow 2}^m \subseteq R_{3 \rightarrow 2}^m$. Note that since $R_{3 \rightarrow 2}$ is a relation, there may be multiple different input transformations.
3. A **function transformation** $\text{Tr}_F^{m,n} : F_3(m, n) \rightarrow F_2(2m, 2n)$ is a function that converts 3VL functions to Boolean functions with appropriate input-output lengths.
4. The **output transformation** $\text{Tr}_{2 \rightarrow 3}^n : \{0, 1\}^{2n} \rightarrow \{T, F, U\}^n$ is the inverse of $R_{3 \rightarrow 2}$. That is, $\text{Tr}_{2 \rightarrow 3}^n((b_1, b_2)) = x$ for every $(x, (b_1, b_2)) \in R_{3 \rightarrow 2}$. Note that since $R_{3 \rightarrow 2}$ is injective, this transformation is *unique*.

Observe that $R_{3 \rightarrow 2}$ is required to be injective since otherwise a Boolean value y could represent two possible 3VL values x, z , and so the output cannot be uniquely mapped back from a Boolean value to a 3VL value. Furthermore, note that by requiring $\text{Tr}_{3 \rightarrow 2}^m \subseteq R_{3 \rightarrow 2}^m$, we have that the transformation constitutes a valid encoding according to the relation.

Informally, a 3VL-Boolean encoding is such that the process of transforming the inputs, computing the transformed Boolean function, and transforming the outputs back, correctly computes the 3VL function. Our definition of an encoding includes the value encoding and function transformation only, and we require that it works correctly for *all* input transformations; we discuss why this is the case below.

Definition 2.1. *Let $m, n \in \mathbb{N}$; let $R_{3 \rightarrow 2}$ be a value encoding, and let $\text{Tr}_F^{m,n}$ be a function transformation. Then, the pair $(R_{3 \rightarrow 2}^m, \text{Tr}_F^{m,n})$ is a 3VL-Boolean Encoding of $F_3(m, n)$ if for every $f_3 \in F_3(m, n)$, every valid input transformation $\text{Tr}_{3 \rightarrow 2}^m$, and every $x \in \{T, F, U\}^m$:*

$$\text{Tr}_{2 \rightarrow 3}^n(f_2(\text{Tr}_{3 \rightarrow 2}^m(x))) = f_3(x) \tag{1}$$

where $f_2 = \text{Tr}_F^{m,n}(f_3)$.

The above definition simply states that computing via the transformations yields correct output. However, as we have mentioned, we require that this works for *all* input transformations and not just for a specific one. It may seem more natural to define a 3VL-Boolean encoding in which the input transformation $\text{Tr}_{3 \rightarrow 2}^m$ is fixed, rather than requiring that Eq. (1) holds for *every* valid input

³ A relation R from X to Y is left-total if for all $x \in X$ there exists $y \in Y$ such that $(x, y) \in R$. R is injective if for every $x, z \in X$ and $y \in Y$, if $(x, y) \in R$ and $(z, y) \in R$ then $x = z$.

⁴ That is, $((A_1, \dots, A_\ell), ((b_1, b_2), \dots, (b_{2\ell-1}, b_{2\ell}))) \in R_{3 \rightarrow 2}^\ell$ if and only if for every $1 \leq i \leq \ell$ it holds that $(A_i, (b_{2i-1}, b_{2i})) \in R_{3 \rightarrow 2}$.

transformation. However, in actuality, it is quite natural to require that the transformed function work for every input transformation since this means that it works for every possible mapping of three-valued inputs to their Boolean counterparts. More significantly, this property is *essential* for proving the composition theorem of Sect. 2.3 that enables us to compose different function encodings together. As we will see, this is important since it enables us to define independent encodings for different types of gates, and then compose them together to compute any function.

2.3 Composition of 3VL Functions

In this section, we prove that encodings can be composed together. Specifically, we prove that for any two 3VL functions g_3 and f_3 and any 3VL input x , computing $g \circ f(x)$ yields the same value as when g, f, x are *separately* transformed into g', f', x' using *any valid* 3VL-Boolean encoding, and then the output of $g' \circ f'(x')$ is transformed back to its 3VL representation. As we will see, this is very important since it enables us to define independent encodings on different types of gates, and then compose them together to compute any function. Formally:

Theorem 2.2. *Let m, ℓ, n be natural numbers, and let $R_{3 \rightarrow 2}$ be a value encoding. Let $E_1 = (R_{3 \rightarrow 2}, \text{Tr}_F^{m, \ell})$ and $E_2 = (R_{3 \rightarrow 2}, \text{Tr}_F^{\ell, n})$ be two 3VL-Boolean encodings (with the same relation $R_{3 \rightarrow 2}$). Then, for every $f_3 \in F_3(m, \ell)$, every $g_3 \in F_3(\ell, n)$, every input transformation $\text{Tr}_{3 \rightarrow 2}^m$, and every $x \in \{T, F, U\}^m$:*

$$\text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(f_2 \left(\text{Tr}_{3 \rightarrow 2}^m(x) \right) \right) \right) = g_3(f_3(x)),$$

where $f_2 = \text{Tr}_F^{m, \ell}(f_3)$ and $g_2 = \text{Tr}_F^{\ell, n}(g_3)$. Equivalently, $(R_{3 \rightarrow 2}, \text{Tr}_F^{\ell, n} \circ \text{Tr}_F^{m, \ell})$ is a 3VL-Boolean encoding of $F_3(m, n)$.

Before proving the theorem, we present the following claim which simply express that if the output transformation of an encoding maps a Boolean value \tilde{Y} to some 3VL value y then there must exist a input transformation that maps y to \tilde{Y} . Formally:

Claim 2.1. *Let $R_{3 \rightarrow 2}$ be a valid value encoding and let $\text{Tr}_{3 \rightarrow 2}, \text{Tr}_{2 \rightarrow 3}$ be a valid input and output transformations respectively such that for $\tilde{Y} \in \{0, 1\}^2$ it holds that $\text{Tr}_{2 \rightarrow 3}(\tilde{Y}) = y$ and $\text{Tr}_{3 \rightarrow 2}(y) = Y$. Then there exists a valid input transformation $\tilde{\text{Tr}}_{3 \rightarrow 2}$ (with respect to $R_{3 \rightarrow 2}$) such that $\tilde{\text{Tr}}_{3 \rightarrow 2}(y) = \tilde{Y}$.*

Proof: If $Y = \tilde{Y}$ then there is nothing to prove, i.e. $\tilde{\text{Tr}}_{3 \rightarrow 2} = \text{Tr}_{3 \rightarrow 2}$. Consider the case of $Y \neq \tilde{Y}$: This means that $R_{3 \rightarrow 2}$ maps the 3VL value y to both Boolean pairs Y and \tilde{Y} . Denote the other two 3VL values by y' and y'' and similarly the

remaining Boolean pairs by Y' and Y'' such that $R_{3 \rightarrow 2}(y') = Y'$. It is immediate that the two valid transformations (with respect to $R_{3 \rightarrow 2}$) are

$$\begin{aligned} \text{Tr}_{3 \rightarrow 2} &= \{y' \mapsto Y', y'' \mapsto Y'', y \mapsto Y\} \quad \text{and} \\ \tilde{\text{Tr}}_{3 \rightarrow 2} &= \left\{y' \mapsto Y', y'' \mapsto Y'', y \mapsto \tilde{Y}\right\} \end{aligned}$$

■

Proof: [of Theorem 2.2]. By the validity of encodings E_1, E_2 (Definition 2.1) it follows that for value encoding $R_{3 \rightarrow 2}$ and every valid input transformations $\text{Tr}_{3 \rightarrow 2}^\ell, \text{Tr}_{3 \rightarrow 2}^m$, every $f_3 \in F_3(m, \ell)$, $g_3 \in F_3(\ell, n)$ and every $x \in \{T, F, U\}^m$:

$$g_3(f_3(x)) = \text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(\text{Tr}_{3 \rightarrow 2}^\ell \left(\text{Tr}_{2 \rightarrow 3}^\ell \left(f_2 \left(\text{Tr}_{3 \rightarrow 2}^m(x) \right) \right) \right) \right) \right) \tag{2}$$

where $f_2 = \text{Tr}_F^{m, \ell}(f_3)$ and $g_2 = \text{Tr}_F^{\ell, n}(g_3)$. This is true due to the following: Let $y_f = \text{Tr}_{2 \rightarrow 3}^\ell(f_2(\text{Tr}_{3 \rightarrow 2}^m(x)))$. By Definition 2.1 y_f is guaranteed to be equal to $f_3(x)$ and $y_g = \text{Tr}_{2 \rightarrow 3}^n(g_2(\text{Tr}_{3 \rightarrow 2}^\ell(y_f)))$ is guaranteed to be equal to $g_3(y_f)$. Concluding that the right hand-side of Eq. 2 equals $g_3(f_3(x))$. In the following we show that we can remove the two intermediate transformations $\text{Tr}_{3 \rightarrow 2}^\ell, \text{Tr}_{2 \rightarrow 3}^\ell$ from the Eq. (2) and obtain the same result: Let

$$\begin{aligned} Y &= f_2(\text{Tr}_{3 \rightarrow 2}^m(x)) \quad \text{and} \\ \hat{y} &= \text{Tr}_{2 \rightarrow 3}^\ell(Y) \end{aligned}$$

Let $\hat{\text{Tr}}_{3 \rightarrow 2}^\ell$ be a valid input transformation (with respect to $R_{3 \rightarrow 2}$) such that $\hat{\text{Tr}}_{3 \rightarrow 2}^\ell(\hat{y}) = Y$ (there must exist such a transformation from Claim 2.1). We get:

$$\begin{aligned} g_3(f_3(x)) &= \text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(\text{Tr}_{3 \rightarrow 2}^\ell \left(\text{Tr}_{2 \rightarrow 3}^\ell \left(f_2 \left(\text{Tr}_{3 \rightarrow 2}^m(x) \right) \right) \right) \right) \right) \\ &= \text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(\text{Tr}_{3 \rightarrow 2}^\ell \left(\text{Tr}_{2 \rightarrow 3}^\ell(Y) \right) \right) \right) \\ &= \text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(\text{Tr}_{3 \rightarrow 2}^\ell(\hat{y}) \right) \right) \\ &= \text{Tr}_{2 \rightarrow 3}^n \left(g_2 \left(\hat{\text{Tr}}_{3 \rightarrow 2}^\ell(\hat{y}) \right) \right) \\ &= \text{Tr}_{2 \rightarrow 3}^n(g_2(Y)) \\ &= \text{Tr}_{2 \rightarrow 3}^n(g_2(f_2(\text{Tr}_{3 \rightarrow 2}^m(x)))) \end{aligned}$$

as required. The 2nd equation follows from the definition of Y ; the 3rd follows from definition of \hat{y} ; the 4th follows from the fact that E_2 is a valid encoding and must work for every valid input transformation, in particular it must work with $\hat{\text{Tr}}_{3 \rightarrow 2}^\ell$; the 5th follows from the way we chose $\hat{\text{Tr}}_{3 \rightarrow 2}^\ell$, i.e. $\hat{\text{Tr}}_{3 \rightarrow 2}^\ell(\hat{y}) = Y$ and the 6th equation follows from the definition of Y . Concluding the correctness of the theorem. ■

We remark that it is crucial that the two encodings in Theorem 2.2 be over the *same* relation and that the encodings be such that they work for all input transformations (as required in Definition 2.1). In order to see why, consider for a moment what could happen if the definition of an encoding considered a *specific* input transformation and was only guaranteed to work for this transformation. Next, assume that f_2 outputs a Boolean pair that is not in the range of the input transformation specified for E_2 . In this case, g_2 may not work correctly and so the composition may fail. We remark that this exact case occurred in natural constructions that we considered in the process of this research. This explains why correctness is required for all possible input transformations in Definition 2.1.

Using Theorem 2.2. This composition theorem is important since it means that we can construct separate encodings for each gate type and then these can be combined in the natural way. That is, it suffices to separately find (efficient) *function transformations* for the functions \wedge_3 , \neg_3 and \oplus_3 and then every 3VL function can be computed by combining the Boolean transformations of these gates. Note that since \neg_3 is typically free and De Morgan’s law holds for this three-valued logic as well, we do not need to separately transform \vee_3 .

2.4 More Generalized Encodings

In order to simplify notation, we have defined encodings to be of the form that every 3VL value $x \in \{T, F, U\}$ is mapped to a pair of Boolean bits; indeed, all of our encodings in this paper are of this form. However, we stress that our formalization is general enough to allow other approaches as well. In particular, it is possible to generalize our definition to allow more general encodings from $x \in \{T, F, U\}^m$ to $y \in \{0, 1\}^\ell$ that could result in $\ell < 2m$. In addition, it is conceivable that mapping $x \in \{T, F, U\}$ to *more* than 2 bits may yield more efficient function transformations with respect to the number of Boolean gates required to compute them. These and other possible encodings can easily be captured by a straightforward generalization of our definition.

3 A Natural 3VL-Boolean Encoding

In this section we present our first 3VL-Boolean encoding which we call the “natural” encoding. This encoding is natural in the sense that a 3VL value x is simply transformed to a pair of 2 Boolean values (x_U, x_T) , such that x_U signals whether the value is known or unknown, and x_T signals whether the value is true or false (assuming that it is known). Formally, we define

$$R_{3 \rightarrow 2} = \{(T, (0, 1)), (F, (0, 0)), (U, (1, 0)), (U, (1, 1))\},$$

and so U is associated with two possible values $(1, 0)$, $(1, 1)$, and T and F are each associated with a single value. Note that $R_{3 \rightarrow 2}$ is left-total and injective

as required by the definition. We now define the input transformation function $\text{Tr}_{3 \rightarrow 2}$ which is defined by mapping T and F to their unique images, and maps U to one of $(1, 0)$ or $(0, 1)$. For concreteness, we define:

$$(x_U, x_T) = \text{Tr}_{3 \rightarrow 2}(x) = \begin{cases} (0, 1) & x = T \\ (0, 0) & x = F \\ (1, 0) & x = U \end{cases}$$

We proceed to define the function transformation Tr_F . As we have discussed, it is possible to define many such transformations, and our aim is to find an “efficient one” with as few AND gates as possible. Below, we present the most efficient transformations of $\wedge_3, \oplus_3, \neg_3$ that we have found for this encoding. As mentioned in Sect. 2.3, these suffice for computing any function (and \vee_3 can be computed using \wedge_3 and \neg_3 by De Morgan’s law).

Let $x, y \in \{T, F, U\}$ be the input values, and let z be the output value. We denote $\text{Tr}_{3 \rightarrow 2}(x) = (x_U, x_T)$ meaning that (x_U, x_T) is the Boolean encoding of x ; likewise for y and z . We define the transformations below. All of these transformations work by computing z_T as the standard logical operation over the x_T, y_T variables (since these indicate T/F), and compute the z_U based on the reasoning as to when the output is unknown. We have:

1. $\text{Tr}_F(\wedge_3)$ outputs the function $\wedge_2(x_U, x_T, y_U, y_T) = (z_U, z_T)$, defined by

$$z_U = (x_U \wedge y_U) \vee (x_U \wedge y_T) \vee (x_T \wedge y_U) \quad \text{and} \quad z_T = x_T \wedge y_T.$$

As mentioned above, $z_T = x_T \wedge y_T$ which gives the correct result and will determine the value if it is known. Regarding z_U , observe that $z_U = 1$ if both x and y equal U or if one of them is U and the other is T (which are the exact cases that the result is unknown). Furthermore, if either of x or y equals F (and so the result should be known), then $z_U = 0$, as required.

2. $\text{Tr}_F(\oplus_3)$ outputs the function $\oplus_2(x_U, x_T, y_U, y_T) = (z_U, z_T)$, defined by

$$z_U = x_U \vee y_U \quad \text{and} \quad z_T = x_T \oplus y_T.$$

Once again, $z_T = x_T \oplus y_T$ which is correct if the value is known. Regarding z_U , recall that for XOR, if either input is unknown then the result is unknown. Thus, $z_U = x_U \vee y_U$.

3. $\text{Tr}_F(\neg_3)$ outputs the function $\neg_2(x_U, x_T) = (z_U, z_T)$, defined by

$$z_U = x_U \quad \text{and} \quad z_T = \neg x_T,$$

which is correct since z_T is computed as above, and $z_U = x_U$ since the output of a negation gate is unknown if and only if the input is unknown.

Correctness. The formal proof that this is a valid encoding is demonstrated simply via the truth tables of each encoding. This can be found in Appendix C.1.

Efficiency. The transformations above have the following cost: \wedge_3 can be computed at the cost of 6 Boolean \wedge and \vee gates (5 for computing z_U and one for computing z_T), \oplus_3 can be computed at the cost of a single Boolean \vee and a single Boolean \oplus gate, and \neg_3 can be computed at the cost of a single Boolean \neg gate. (We ignore \neg gates from here on since they are free in all known garbling schemes.)

Concretely, when using the garbling scheme of [12] that incorporates free-XOR and requires two ciphertexts for \vee and \wedge , we have that the cost of garbling \vee_3 is 12 ciphertexts, and the cost of garbling \oplus_3 is 2 ciphertexts. In comparison, recall that the naive garbling scheme of Sect. 1.3 required 8 ciphertexts for both \vee_3 and \oplus_3 . In order to see which is better, let C be a 3VL circuit and denote by C_\wedge and C_\oplus the number of \wedge_3 and \oplus_3 gates in C , respectively. Then, the natural 3VL-Boolean encoding is better than the naive approach of Sect. 1.3 if and only if

$$12 \cdot C_\wedge + 2 \cdot C_\oplus < 8 \cdot C_\wedge + 8 \cdot C_\oplus,$$

which holds if and only if $C_\wedge < 1.5 \cdot C_\oplus$. This provides a clear tradeoff between the methods. We now proceed to present encodings that are strictly more efficient than both the natural 3VL Boolean encoding and the naive garbling of Sect. 1.3.

4 A More Efficient Encoding Using a Functional Relation

In this section we present a 3VL-Boolean encoding, in which the relation $R_{3 \rightarrow 2}$ is *functional*.⁵ Since $R_{3 \rightarrow 2}$ is already left-total and injective, this implies that $R_{3 \rightarrow 2}$ is in fact a 1-1 function. We define $R_{3 \rightarrow 2} = \{(T, (1, 1)), (F, (0, 0)), (U, (1, 0))\}$. Since $R_{3 \rightarrow 2}$ is a 1-1 function, there is only one possible input transformation $(x_T, x_F) = \text{Tr}_{3 \rightarrow 2} = R_{3 \rightarrow 2}^{-1}$. The intuition behind this encoding is as follows: The value $x \in \{T, F, U\}$ is mapped to a pair (x_T, x_F) so that if x is true or false then $x_T = x_F$, appropriately (i.e., if $x = T$ then $x_T = x_F = 1$, and if $x = F$ then $x_T = x_F = 0$). In contrast, if x is unknown, then x_T and x_F take different values of 1 and 0, respectively, representing an “unknown” state (both 1 and 0). We denote the Boolean values x_T and x_F because in case that $x = U$ then x_T is assigned with True and x_F is assigned with False.

As we will see, it is possible to compute \wedge_3 , \oplus_3 and \neg_3 gates under this encoding at a cost that is strictly more efficient than the natural encoding of Sect. 3. In order to show this, in Sect. 4.1, we begin by presenting a simple transformation Tr_F for \wedge_3 and \neg_3 gates. These are clearly complete, and furthermore are the most common connectives used in the context of SQL (as above, \neg_3 is “free” and so \vee_3 can be transformed at the same cost as \wedge_3). However, for the general case, an efficient transformation for \oplus_3 gates is also desired since the naive method of computing \oplus from \wedge, \vee, \neg is quite expensive. We therefore show how to also deal with \oplus_3 gates in Sect. 4.2.

⁵ Relation R from X to Y is functional if for all $x \in X$ and $y, z \in Y$ it holds that if $(x, y) \in R$ and $(x, z) \in R$ then $y = z$. Stated differently, R is a function.

4.1 An Efficient Function Transformation for \wedge_3, \neg_3 Gates

We now show how to transform \wedge_3 and \neg_3 gates into Boolean forms at a very low cost: \wedge_3 gates can be transformed at the cost of just two Boolean \wedge gates, and \neg_3 gates can be transformed at the cost of two Boolean \neg gates (which are free in all garbling schemes).

1. $\text{Tr}_F(\wedge_3)$ outputs the function $\wedge_2(x_T, x_F, y_T, y_F) = (z_T, z_F)$, defined by

$$z_T = x_T \wedge y_T \quad \text{and} \quad z_F = x_F \wedge y_F.$$

2. $\text{Tr}_F(\neg_3)$ outputs the function $\neg_2(x_T, x_F) = (z_T, z_F)$, defined by

$$z_T = \neg x_F \quad \text{and} \quad z_F = \neg x_T.$$

We now prove that these transformations are correct. We begin with $\text{Tr}_F(\wedge_3)$:

1. If $x \wedge y = T$ then $x = y = T$ and so $x_T = x_F = y_T = y_F = 1$. Thus, $z_T = z_F = 1$ which means that $z = \text{Tr}_{2 \rightarrow 3}(z_T, z_F) = \text{Tr}_{2 \rightarrow 3}(1, 1) = T$, as required.
2. If $x \wedge y = F$, then either $x = F$ which means that $x_T = x_F = 0$, or $y = F$ which means that $y_T = y_F = 0$, or both. This implies that $z_T = z_F = 0$ and so $z = \text{Tr}_{2 \rightarrow 3}(z_T, z_F) = \text{Tr}_{2 \rightarrow 3}(0, 0) = F$, as required.
3. Finally, if $x \wedge y = U$, then we have three possible cases:
 - (a) *Case 1: $x = y = U$:* In this case, $x_T = y_T = 1$ and $x_F = y_F = 0$, and thus $z_T = 1, z_F = 0$ and $z = \text{Tr}_{2 \rightarrow 3}(z_T, z_F) = \text{Tr}_{2 \rightarrow 3}(1, 0) = U$, as required.
 - (b) *Case 2: $x = T$ and $y = U$:* In this case, $x_T = x_F = y_T = 1$ and $y_F = 0$, and thus $z_T = 1$ and $z_F = 0$, implying that $z = U$, as required.
 - (c) *Case 3: $x = U$ and $y = T$:* This case is symmetric to the previous case and so also results in U , as required.

It remains to prove that $\text{Tr}_F(\neg_3)$ is correct:

1. If $x = T$, then $x_T = x_F = 1$ and so $z_T = z_F = 0$. Thus, $z = \text{Tr}_{2 \rightarrow 3}(0, 0) = F$, as required.
2. If $x = F$, then $x_T = x_F = 0$ and so $z_T = z_F = 1$. Thus, $z = \text{Tr}_{2 \rightarrow 3}(1, 1) = T$, as required.
3. If $x = U$, then $x_T = 1$ and $x_F = 0$ and so $z_T = \neg x_F = 1$ and $z_F = \neg x_T = 0$. Thus, $z = \text{Tr}_{2 \rightarrow 3}(1, 0) = U$, as required.

Efficiency. The transformations above are very efficient and require 2 Boolean AND gates for every 3VL-AND (or 3VL-OR) gate, and 2 Boolean NOT gates for each 3VL-NOT gate. Using the garbling scheme of [12], this means 4 ciphertext for each \wedge_3, \vee_3 gate, and 0 ciphertexts for \neg_3 gates. This is far more efficient than any of the previous encodings. However, as we have mentioned above, we still need to show how to compute \oplus_3 gates.

4.2 An Efficient Function Transformation for \oplus_3 Gates

We now present the transformation for \oplus_3 gates for the above functional relation. We begin by remarking that the method above for \wedge_3 gates does not work for \oplus_3 gates.

For example, if we define $z_T = x_T \oplus y_T$ and $z_F = x_F \oplus y_F$, then the result is correct as long as neither of x or y are unknown: If both are unknown then $x = y$, and thus $z_T = z_F = 0$. The result of transforming $(z_T, z_F) = (0, 0)$ back to a 3VL is F rather than U . If only one is unknown then $x \neq y$,

x	y	$x \oplus_3 y$	x_T	x_F	y_T	y_F	(z_T, z_F)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	U	0	0	1	0	(1, 0)	U
F	T	T	0	0	1	1	(1, 1)	T
U	F	U	1	0	0	0	(1, 0)	U
U	U	U	1	0	1	0	(0, 0)	F
U	T	U	1	0	1	1	(0, 1)	undefined
T	F	T	1	1	0	0	(1, 1)	T
T	U	U	1	1	1	0	(0, 1)	undefined
T	T	F	1	1	1	1	(0, 0)	F

Fig. 3. The result of the transformation of \oplus_3 by $(z_T, z_F) = (x_T \oplus y_T, x_F \oplus y_F)$

and thus $z_T = 0$ and $z_F = 1$). The result of transforming $(z_T, z_F) = (0, 1)$ is undefined since the pair (0, 1) is not in the range of $R_{3 \rightarrow 2}$. In general, the truth table for the transformation $z_T = x_T \oplus y_T$ and $z_F = x_F \oplus y_F$ appears in Fig. 3; the blue lines are where this transformation is incorrect.

Our transformation must therefore “fix” the incorrect rows in Fig. 3. We define $\text{Tr}_F(\oplus_3)$ that outputs the function $\oplus_2(x_T, x_F, y_T, y_F) = (z_T, z_F)$ defined by

$$\begin{aligned}
 z'_T &= (x_T \oplus y_T) \oplus ((x_T \oplus x_F) \wedge (y_T \oplus y_F)) \quad \text{and} \quad z'_F = x_F \oplus y_F \\
 \text{aux} &= \neg z'_T \wedge z'_F \\
 z_T &= z'_T \oplus \text{aux} \quad \text{and} \quad z_F = z'_F \oplus \text{aux}
 \end{aligned}$$

Observe that the value (z'_T, z'_F) is just the transformation in Fig. 3, with the addition that z'_T is adjusted so that it is flipped in the case that both $x = y = U$ (since in that case $x_T \neq x_F$ and $y_T \neq y_F$). This therefore fixes the 5th row in Fig. 3 (i.e., the input case of $x = y = U$). Note that it doesn’t affect any other input cases since $(x_T \oplus x_F) \wedge (y_T \oplus y_F)$ equals 0 in all other cases.

In order to fix the 6th and 8th rows in Fig. 3, it is necessary to adjust the output in the case that (0, 1) is received, and only in this case (note that this is only received in rows 6 and 8). Note that the aux variable is assigned value 1 if and only if $z'_T = 0$ and $z'_F = 1$. Thus, defining $z_T = z'_T \oplus \text{aux}$ and $z_F = z'_F \oplus \text{aux}$ adjusts $(z'_T, z'_F) = (0, 1)$ to $(z_T, z_F) = (1, 0)$ which represents U as required. Furthermore, no other input cases are modified and so the resulting function is correct.

Correctness. The formal proof that this is a valid encoding is demonstrated simply via the truth tables of each encoding. This can be found in Appendix C.2.

Efficiency. The transformation of \oplus_3 incurs a cost of two Boolean \wedge gates and 6 Boolean \oplus gates. Utilizing free-XOR and the garbling scheme of [12], we have that 4 ciphertexts are required for garbling \oplus_3 gates.

Combining this with Sect. 4.1, we have a cost of 4 ciphertexts for \wedge_3 and \oplus_3 gates, and 0 ciphertexts for \neg_3 gates. This is far more efficient than the naive garbling of Sect. 1.3 for all gate types. Next, recall that the natural encoding of Sect. 3 required 12 ciphertexts for \wedge_3 gates and 2 ciphertexts for \oplus_3 gates. Thus, denoting by C_\wedge and C_\oplus the number of \wedge_3 and \oplus_3 gates, respectively, in a 3VL circuit C , we have that the scheme in this section is more efficient if and only if $4 \cdot C_\wedge + 4 \cdot C_\oplus < 12 \cdot C_\wedge + 2 \cdot C_\oplus$, which holds if and only if $C_\oplus < 4 \cdot C_\wedge$. Thus, the natural encoding is only better if the number of \oplus_3 gates is *over four times* the number of \wedge_3 gates in the circuit. In Sect. 5, we present transformations that perform better in some of these cases.

5 Encoding Using a Non-functional Relation

In this section, we present an alternative encoding that is more expensive for \wedge_3 gates but cheaper for \oplus_3 gates, in comparison to the encoding of Sect. 4. The value encoding that we use in this section is the same as in Sect. 4, except that we also include $(0, 1)$ in the range; thus the relation is no longer functional. Since the motivation regarding the relation is the same as in Sect. 4, we proceed directly to define the relation:

$$R_{3 \rightarrow 2} = \{(T, (1, 1)), (F, (0, 0)), (U, (0, 1)), (U, (1, 0))\}.$$

Thus, $R_{3 \rightarrow 2}$ maps the 3VL value U to both Boolean pairs $(0, 1)$ and $(1, 0)$. As such, there are two admissible input transformation functions $\text{Tr}_{3 \rightarrow 2}$. Both of them map T to $(1, 1)$ and map $(0, 0)$ to F ; one of them maps U to $(1, 0)$ the other maps U to $(0, 1)$. Recall that our function transformation needs to work for both, in order for the composition theorem to hold.

We use the same notation of (x_T, x_F) as in Sect. 4 for the Boolean pairs in the range of $R_{3 \rightarrow 2}$. The motivation is the same as before; if $x = T$ or $x = F$ then both values are the same; if $x = U$ then the “true” bit x_T is different from the “false” bit x_F .

The transformation Tr_F for each gate type is given below.

1. $\text{Tr}_F(\wedge_3)$ outputs the function $\wedge_2(x_T, x_F, y_T, y_F) = (z_T, z_F)$, defined by:

$$\begin{aligned} z_T &= x_T \wedge y_T \\ z_F &= (x_F \wedge y_F) \oplus \left((x_T \oplus x_F) \wedge (y_T \oplus y_F) \wedge (\neg(x_F \oplus y_T)) \right) \end{aligned}$$

Recall that in Sect. 4, it sufficed to define $z_T = x_T \wedge y_T$ and $z_F = x_F \wedge y_F$. However, this does not yield a correct result in this encoding in the case that x and y are both unknown, and x is encoded as $(0, 1)$ and y is encoded as $(1, 0)$. Specifically, in this case, z is computed as F instead of as U . We fix this case by changing the second bit of z (i.e., z_F) when the encodings are of this form. Observe that the expression $(x_T \oplus x_F) \wedge (y_T \oplus y_F) \wedge (\neg(x_F \oplus y_T))$ evaluates to 1 if and only if $x_T \neq x_F$ and $y_T \neq y_F$ and $x_F = y_T$, which is exactly the case that one of the value is encoded as $(1, 0)$ and the other is encoded as $(0, 1)$.

2. $\text{Tr}_F(\oplus_3)$ outputs the function $\oplus_2(x_T, x_F, y_T, y_F) = (z_T, z_F)$, defined by:

$$\begin{aligned} z_T &= (x_T \oplus y_T) \oplus ((x_T \oplus x_F) \wedge (y_T \oplus y_F)) \\ z_F &= x_F \oplus y_F \end{aligned}$$

This is the same transformation of \oplus_3 described in Sect. 4.2 for the functional encoding of Sect. 4, except that here there is no need to switch the left and right bits of the result in the case that they are (0, 1). This is due to the fact that (0, 1) is a valid encoding of U under $R_{3 \rightarrow 2}$ used here.

3. $\text{Tr}_F(\neg_3)$ outputs the function $\vee_2(x_T, x_F) = (z_T, z_F)$, defined by:

$$z_T = \neg x_T \quad \text{and} \quad z_F = \neg x_F$$

This is almost the same as the transformation of \neg_3 in Sect. 4.1, excepts that we do not exchange the order of the bits. Again, this is due to the fact that both (1, 0) and (0, 1) are valid encodings of 0 and so the negation of U by just complementing both bits results in U and is correct.

Correctness. The formal proof that this is a valid encoding is demonstrated simply via the truth tables of each encoding. This can be found in Appendix C.3.

Efficiency. The Boolean function $\text{Tr}_F(\wedge_3)$ requires 4 AND gates, which translates to 8 ciphertexts using the garbling of [12]. The Boolean function $\text{Tr}_F(\oplus_3)$ requires only one AND gate, which translates to two ciphertexts using the garbling of [12]. Denote by C_\wedge and C_\oplus the number of \wedge_3 and \oplus_3 gates in the 3VL circuit, then the encoding of this section is better than that of Sect. 4 if and only if $8 \cdot C_\wedge + 2 \cdot C_\oplus < 4 \cdot C_\wedge + 4 \cdot C_\oplus$ which holds if and only if $C_\oplus > 2 \cdot C_\wedge$. Observe also that the encoding in this section is always at least as good as the natural encoding of Sect. 3; in particular, it has the same cost for \oplus_3 gates and is strictly cheaper for \wedge_3 gates.

6 Efficiency Summary of the Different Methods

We have presented a naive garbling method and three different encodings. We summarize the efficiency of these different methods, as a function of the number of ciphertexts needed when garbling, in Table 2.

Table 2. A summary of the garbling efficiency of the different methods

Encoding	Ciphertexts for \wedge_3	Ciphertexts for \oplus_3	Best in range
Section 1.3 – Naive	8	8	None
Section 3 – Natural	12	2	None
Section 4 – Functional	4	4	$C_\oplus < 2 \cdot C_\wedge$
Section 5 – Non-functional	8	2	$C_\oplus > 2 \cdot C_\wedge$

7 A Black-Box Protocol for Computing 3VL Circuits

In this section, we show how to securely compute 3VL circuits. Of course, one could design a protocol from scratch using a garbled 3VL circuit. However, our goal is to be able to use *any* protocol that can be used to securely evaluate a Boolean circuit, and to directly inherit its security properties. This approach is simpler, and allows us to leverage existing protocol optimizations for the Boolean case.

Before proceeding, we explain why there is an issue here. Seemingly, one could compile any 3VL-circuit into a Boolean circuit using our method above, and then run the secure computation protocol on the Boolean circuit to obtain the output. As we will see, this is actually not secure. Fortunately, however, it is very easy to fix. We now explain why this is not secure:

1. *Output leakage*: The first problem that arises is due to the fact that Definition 2.1 allows $R_{3 \rightarrow 2}$ to be a non-functional relation. This implies that a value $x \in \{T, F, U\}$ might be mapped to two or more Boolean representations. Now, if a secure protocol is run on the Boolean circuit, this implies that a single 3VL output could be represented in more than one way. This could potentially leak information that is not revealed by the function itself. In Appendix B, we show a concrete scenario where this does actually reveal more information than allowed. We stress that this leakage can occur even if the parties are *semi-honest*.

This leakage can be avoided by simply transforming y to a *unique*, predetermined Boolean value y^* at the end of the circuit computation and before outputs are revealed. This is done by incorporating an “output translation” gadget into the circuit for every output wire.

2. *Insecurity due to malicious inputs*. Recall that the relation $R_{3 \rightarrow 2}$ does not have to be defined over the entire range of $\{0, 1\} \times \{0, 1\}$, and this is indeed the case for the relation that we use in Sect. 4. In such a case, if the malicious party inputs a Boolean input that is not legal (i.e., is not in the range of $R_{3 \rightarrow 2}$), then this can result in an incorrect result (or worse).

This cheating can be prevented by incorporating an “input translation” gadget for every input wire of the circuit that deterministically translates all possible Boolean inputs (even invalid ones) into valid inputs that appear in the range of $R_{3 \rightarrow 2}$. This prevents a malicious adversary from inputting incorrect values (to be more exact, it can input incorrect values but they will anyway be translated into valid ones).

The key observation from above is that the solutions to both problems involve modifications to the circuit only. Thus, *any* protocols that is secure for arbitrary Boolean circuits can be used to securely compute 3VL circuits. Furthermore, these input and output gadgets are very small (at least for all of our encodings) and thus do not add any significant overhead.

We have the following theorem⁶:

Theorem 7.1. *Let π be a protocol for securely computing any Boolean circuit, let f_3 be a 3VL function with an associated 3VL circuit C , and let C' be a Boolean circuit that is derived from C via a valid 3VL-Boolean encoding. Then, Denote by C'_1 the circuit obtained by adding output-translation gadgets to C' , and denote by C'_2 the circuit obtained by adding input-translation and output-translation gadget to C' .*

1. *If π is secure in the presence of semi-honest adversaries, then protocol π with circuit C'_1 securely computes the 3VL function f_3 in the presence of semi-honest adversaries.*
2. *If π is secure in the presence of malicious (resp., covert) adversaries, then protocol π with circuit C'_2 securely computes the 3VL function f_3 in the presence of malicious (resp., covert) adversaries.*

Secure computation. The above theorem holds for *any* protocol for secure computation. This includes protocols based on Yao and garbled circuits [8, 13], as well as other protocols like that of [4].

8 Lower Bounds

One of the most important optimizations of the past decade for garbled circuits is that of free-XOR [7]. Observe that none of the 3VL-Boolean encodings that we have presented have free-XOR, and the cheapest transformation of \oplus_3 requires 2 ciphertexts. In this section, we ask the following question:

Can free-XOR garbling be achieved for 3VL functions?

We prove a negative answer for a *linear garbling scheme*, which is defined in the Linicrypt model of [1]. Our proof is based on a reduction from any garbling scheme for 3VL circuit to a garbling scheme for Boolean circuits. Specifically, we show that any garbling scheme for 3VL-XOR can be used to garble Boolean-AND gates at the exact same cost. Now, [12] proved that at least 2 ciphertexts are required for garbling AND gates using any linear garbling method. By reducing to this result, we will show that 3VL-XOR cannot be garbled with less than two ciphertexts using any linear garbling method. Thus, a significant breakthrough in garbling would be required to achieve free-XOR in the 3VL setting, or even to reduce the cost of 3VL-XOR to below two ciphertexts.

Reducing Boolean AND to 3VL XOR. It is actually very easy to compute a Boolean AND gate given a 3VL XOR gate. This is due to the fact that 3VL XOR actually contains an *embedded AND*; this is demonstrated in Fig. 4.

⁶ The proof of the theorem is straightforward and is thus omitted.

This can be utilized in the following way. Let \tilde{g} be a garbled 3VL-XOR gate with input wires x, y and output wire z . By definition, given keys k_x^α and k_y^β on the input wires with $\alpha, \beta \in \{T, F, U\}$, the garbled gate can be used to compute the key k_z^γ on the output wire where $\gamma = \alpha \oplus_3 \beta$. Thus, in order to compute a Boolean AND gate, the following can be carried out. First, associate the 3VL-value F with the Boolean value 1 (True), and associate the 3VL-value U with the Boolean value 0 (False). Then, given any two of k_x^U, k_x^F and k_y^U, k_y^F the output of the garbled gate will be k_z^F if and only if $x = y = F$, which is exactly a Boolean AND gate. (This is depicted in the shaded square in Fig. 4.) Observe that the 3VL-value T is not used in this computation and so is ignored. The fact that this method is a *secure* garbling of an AND gate follows directly from the security of the 3VL garbling scheme.

\oplus_3	T	F	U
T	F	T	U
F	T	F	U
U	U	U	U

Fig. 4. Shows that \oplus_3 embeds the truth table of both \wedge, \vee and \oplus .

It follows that a (single) Boolean AND gate can be garbled at the same cost of a 3VL XOR gate. Thus, free-3VL-XOR would imply free-Boolean-AND, and even 3VL XOR with just a single ciphertext would imply a construction for garbling a Boolean AND gate at the cost of just one ciphertext. Both of these would be surprising results. We now formalize this more rigorously using the framework of linear garbling.

Impossibility for linear garbling. The notion of linear garbling was introduced by [12], who also showed that all known garbling schemes are linear. In their model, the garbling and evaluation algorithms use only linear operations, apart from queries to a random oracle (which may be instantiated by the garbling scheme) and choosing which linear operation to apply based on some select bits for a given wire. They prove that for every ideally secure linear garbling scheme (as defined in [12]), at least two ciphertexts must be communicated for every Boolean AND gate in the circuit. Combining [12, Theorem 3] with what we have shown above, we obtain the following theorem with regards to garbling schemes for 3VL circuits in the same model.

Theorem 8.1. *Every ideally secure garbling scheme for 3VL-XOR gates, that is linear in the sense defined in [12], has the property that the garbled gate consists of at least $2n$ bits, where n is the security parameter.*

This explains why we do not achieve free-XOR in our constructions in the three-valued logic setting.

A Exhaustive Search for Expressions with One Boolean AND

We present a simplified version of the technique that we used in order to search for a Boolean expression with only one AND gate (and an unlimited number of XOR gates) that implements the functionality of a 3VL AND and 3VL XOR.

We actually used several simple optimizations to this technique to make it run faster, but these are not of significance to the discussion and so are omitted.

In this paper we focus on a specific set of possible 3VL-to-Boolean encodings, specifically, we focus on encodings that map each 3VL value x (i.e. T, F, U) to a pair of Boolean values (x_L, x_R) (L, R for left and right). Note that this means that either the encoding is *functional*, which means that each 3VL value is mapped to exactly one Boolean pair and hence there remains one invalid Boolean pair, or the encoding is *non-functional* which means that one 3VL value is mapped to two Boolean pairs while the other two 3VL values are mapped to a single Boolean pair. The total number of possible encodings (functional and non-functional) is 60, as can be seen by a simple combinatorial computation.

Let Enc be some 3VL-to-Boolean encoding. A Boolean implementation of 3VL-AND (resp. 3VL-XOR) using Enc is given two pairs of Boolean values, (x_L, x_R) and (y_L, y_R) , and outputs a single pair of Boolean values (z_L, z_R) , such that when given encodings of the 3VL values x and y it outputs an encoding of $x \wedge_3 y$ (resp. $x \oplus_3 y$) where \wedge_3 is a 3VL-AND (resp. \oplus_3 is a 3VL-XOR). When Enc is non-functional, this should hold for every possible encoding of x and y . This means that for a functional encoding we test the correctness of 9 possibilities, and for a non-functional encoding we test the correctness of 16 possibilities of (x_L, x_R) and (y_L, y_R) .

Since we are interested in an implementation with a single Boolean AND gate, the values of z_L and z_R are basically a Boolean expression over the four literals x_L, x_R, y_L, y_R and the constant 1 (the constant 0 can be obtained by simply XORing the literal with itself) with a single Boolean AND and unlimited number of Boolean XOR gates. Our goal is to find a way to enumerate over all these expressions and test if they form a correct implementation of the 3VL-AND and 3VL-XOR.

The exhaustive search process is depicted in Fig. 5. We set $E_0 = \{x_L, x_R, y_L, y_R, 1\}$. This is the set of initial values, from which the expressions for z_L, z_R are formed. Before we apply a Boolean AND to the above values we first want to obtain a set of all possible expressions using Boolean XOR gates only, we denote this set by E_0^+ . Note that E_0^+ can be obtained by taking the XOR of each non-empty subset of values from E_0 , which means that $|E_0^+| = |E_0| + \sum_{i=1}^5 \binom{5}{i} = 36$. Then, we can choose 2 expressions $e_1, e_2 \in E_0^+$ and apply $e_3 = e_1 \wedge e_2$. We denote the set of possible expressions of this form by E_1 and by counting we get that $E_1 = \binom{|E_0^+|}{2} = \binom{36}{2} = 630$. Notice that E_1 does not contain *all* possible expressions with exactly one Boolean AND, since XOR operations are only computed before the AND. Thus, for example, $x_L \oplus (x_R \wedge y_L) \notin E_1$. In order to obtain the

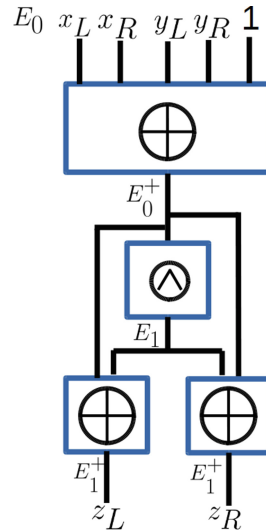


Fig. 5. The exhaustive search process.

set of *all* possible expressions, denoted E_1^+ , we need to add another layer of Boolean XORs after the AND. However, since we want to test *pairs* of Boolean expressions for z_L, z_R using up to one Boolean AND gate, we may use the *same* expression from E_1 and apply XOR after it with two different expressions from E_0^+ . Therefore, we have that $E_1^+ = E_1 \times (E_0^+ \times E_0^+)$. Note that E_1^+ also contains expressions with no Boolean AND at all. For example, for any expression e without Boolean AND gates, E_1^+ also contains $e = (1 \wedge 1) \oplus (\neg e)$. We therefore conclude that $|E_1^+| = |E_1| \cdot |E_0^+|^2 = 630 \cdot 36^2 = 816480$.

Putting it all together, we have 60 possible encodings. For each encoding, we have 816,480 possible pairs of expressions for z_L, z_R with up to one AND gate and for each possible pair we need to test its correctness over 9 or 16 possible inputs. The total number of tests is therefore $60 \cdot 816480 \cdot 9 = 440,899,200 \approx 2^{28.7}$ or $60 \cdot 816480 \cdot 16 = 783,820,800 \approx 2^{29.5}$.

B Insecurity of the Naive Protocol for Evaluating 3VL Functions

In this section we provide a concrete attack on a protocol that uses a valid Boolean encoding, without adding the input/output gadgets described in Sect. 7. Consider the 3VL function $f_3 : \{T, F, U\}^2 \rightarrow \{T, F, U\}^2$ defined by $f_3(a, b) = a \oplus_3 b$; denote the output by c . Now, consider a 2-party protocol for evaluating this function, where P_1 inputs both a and b , and P_2 does not input anything. (Needless to say, this is a silly example since such a function can be singlehandedly computed by P_1 and the result can be sent to P_2 . However, this illustrates the problem, and of course applies to more “interesting” cases as well.)

Now, assume that the output of the function is U . In this case, a secure evaluation of f does not reveal anything to P_2 except the fact that (a, b) is either (U, U) , (T, U) , (U, T) , (F, U) or (U, F) . Furthermore, consider the case that P_1 ’s inputs are random. In this case, each of these possible inputs occurs with probability $\frac{1}{5}$ (assuming P_2 has no auxiliary information). Consider now what happens if a secure two-party protocol is run to compute this function on the encoding, without applying an output transformation gadget as described in Sect. 7. For the sake of concreteness, consider the non-functional relation encoding of Sect. 5. For this encoding U can be mapped to $(1, 0)$ or to $(0, 1)$. Assume that $\text{Tr}_{3 \rightarrow 2}(U) = (0, 1)$. Then, the possible outputs of the function (since it is just a single XOR) are given in Table 3; the shaded rows are associated with output U :

Observe that if P_2 receives $(z_T, z_F) = (0, 1)$ for output, then it *knows* that P_1 ’s input was either (F, U) or (U, F) . In contrast, if P_2 receives $(z_T, z_F) = (1, 0)$ for output, then it *knows* that P_1 ’s input was either (U, U) or (U, T) or (T, U) . This is clearly information that P_2 should not learn (observe also that if P_2 receives $(0, 1)$ then it know with full certainty that either $a = F$ or $b = F$). This is therefore not a secure protocol.

Table 3. $\text{Tr}_F(\oplus_3)$

x	y	z	x_T	x_F	y_T	y_F	(z_T, z_F)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	U	0	0	0	1	(0, 1)	U
F	T	T	0	0	1	1	(1, 1)	T
U	F	U	0	1	0	0	(0, 1)	U
U	U	U	0	1	0	1	(1, 0)	U
U	T	U	0	1	1	1	(1, 0)	U
T	F	T	1	1	0	0	(1, 1)	T
T	U	U	1	1	0	1	(1, 0)	U
T	T	F	1	1	1	1	(0, 0)	F

C Formal Proofs of Encodings via Truth Tables

In this appendix, we provide the truth tables for each of our encoding methods. These truth tables constitute a formal proof of correctness, since they show that the mapping from input to output is correct for all possible inputs.

C.1 Correctness of the Natural Encoding

See Tables 4, 5 and 6.

Table 4. The Boolean encoding of 3VL-AND in Sect. 3

Table 5. The Boolean encoding of 3VL-XOR in Sect. 3

Table 6. The Boolean encoding of 3VL-NOT in Sect. 3

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	F	0	0	0	1	(0, 0)	F
F	T	F	0	0	1	0	(0, 0)	F
F	U	F	0	0	1	1	(0, 0)	F
U	F	F	0	1	0	0	(0, 0)	F
U	U	F	0	1	0	1	(0, 1)	U
U	T	F	0	1	1	0	(0, 1)	U
U	U	F	0	1	1	1	(0, 1)	U
T	F	F	1	0	0	0	(0, 0)	F
T	U	F	1	0	0	1	(0, 1)	U
T	T	F	1	0	1	0	(1, 0)	T
T	U	F	1	0	1	1	(1, 1)	U
T	T	F	1	0	1	0	(1, 0)	T
T	U	F	1	0	1	1	(1, 1)	U
U	F	F	1	1	0	0	(0, 0)	F
U	U	F	1	1	0	1	(0, 1)	U
U	T	F	1	1	1	0	(1, 1)	U
U	U	F	1	1	1	1	(1, 1)	U

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	F	0	0	0	1	(0, 1)	U
F	T	F	0	0	1	0	(1, 0)	T
F	U	F	0	0	1	1	(1, 1)	U
U	F	F	0	1	0	0	(0, 1)	U
U	U	F	0	1	0	1	(0, 1)	U
U	T	F	0	1	1	0	(1, 1)	U
U	U	F	0	1	1	1	(1, 1)	U
T	F	F	1	0	0	0	(0, 0)	F
T	U	F	1	0	0	1	(0, 1)	U
T	T	F	1	0	1	0	(1, 0)	T
T	U	F	1	0	1	1	(1, 1)	U
U	F	F	1	1	0	0	(0, 0)	F
U	U	F	1	1	0	1	(0, 1)	U
U	T	F	1	1	1	0	(1, 1)	U
U	U	F	1	1	1	1	(1, 1)	U

x	$\neg_3(x)$	x_T	x_U	(z_T, z_U)	z
F	T	0	0	(1, 0)	T
U	U	0	1	(1, 1)	U
T	F	1	0	(0, 0)	F
U	U	1	1	(0, 1)	U

C.2 Correctness of the Encoding Using a Functional Relation

See Tables 7, 8 and 9.

Table 7. The Boolean encoding of 3VL-AND in Sect. 4

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	F	0	0	1	0	(0, 0)	F
F	T	F	0	0	1	1	(0, 0)	F
U	F	F	1	0	0	0	(0, 0)	F
U	U	U	1	0	1	0	(1, 0)	U
U	T	U	1	0	1	1	(1, 0)	U
T	F	F	1	1	0	0	(0, 0)	F
T	U	U	1	1	1	0	(1, 0)	U
T	T	T	1	1	1	1	(1, 1)	T

Table 8. The Boolean encoding of 3VL-XOR in Sect. 4

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	F	U	0	0	1	0	(1, 0)	U
F	F	T	0	0	1	1	(1, 1)	T
F	U	F	1	0	0	0	(1, 0)	U
F	U	U	1	0	1	0	(1, 0)	U
F	T	U	1	0	1	1	(1, 0)	U
T	F	F	1	1	0	0	(1, 1)	T
T	U	U	1	1	1	0	(1, 0)	U
T	T	F	1	1	1	1	(0, 0)	F

Table 9. The Boolean encoding of 3VL-NOT in Sect. 4

x	$\neg_3(x)$	x_T	x_U	(z_T, z_U)	z
F	T	0	0	(1, 1)	T
U	U	1	0	(1, 0)	U
T	F	1	1	(0, 0)	F

C.3 Correctness of the Encoding Using a Non-functional Relation

See Tables 10, 11 and 12.

Table 10. The Boolean encoding of 3VL-AND in Sect. 5

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	F	0	0	0	1	(0, 0)	F
F	U	F	0	0	1	0	(0, 0)	F
F	T	F	0	0	1	1	(0, 0)	F
U	F	F	0	1	0	0	(0, 0)	F
U	U	U	0	1	0	1	(0, 1)	U
U	U	U	0	1	1	0	(0, 1)	U
U	T	U	0	1	1	1	(0, 1)	U
U	F	F	1	0	0	0	(0, 0)	F
U	U	U	1	0	0	1	(0, 1)	U
U	U	U	1	0	1	0	(1, 0)	U
U	T	U	1	0	1	1	(1, 0)	U
T	F	F	1	1	0	0	(0, 0)	F
T	U	U	1	1	0	1	(0, 1)	U
T	U	U	1	1	1	0	(1, 0)	U
T	T	T	1	1	1	1	(1, 1)	T

Table 11. The Boolean encoding of 3VL-XOR in Sect. 5

x	y	z	x_T	x_U	y_T	y_U	(z_T, z_U)	z
F	F	F	0	0	0	0	(0, 0)	F
F	U	U	0	0	0	1	(0, 1)	U
F	U	F	0	0	1	0	(1, 0)	U
F	T	T	0	0	1	1	(1, 1)	T
U	F	U	0	1	0	0	(0, 1)	U
U	U	U	0	1	0	1	(1, 0)	U
U	U	U	0	1	1	0	(0, 1)	U
U	T	U	0	1	1	1	(1, 0)	U
U	F	F	1	0	0	0	(1, 0)	U
U	U	U	1	0	0	1	(0, 1)	U
U	U	U	1	0	1	0	(1, 0)	U
U	T	U	1	0	1	1	(0, 1)	U
T	F	F	1	1	0	0	(1, 1)	T
T	U	U	1	1	0	1	(1, 0)	U
T	U	U	1	1	1	0	(0, 1)	U
T	T	F	1	1	1	1	(0, 0)	F

Table 12. The Boolean encoding of 3VL-NOT in Sect. 5

x	$\neg_3(x)$	x_T	x_U	(z_T, z_U)	z
F	T	0	0	(1, 1)	T
U	U	0	1	(0, 1)	U
U	U	1	0	(1, 0)	U
T	F	1	1	(0, 0)	F

References

1. Carmer, B., Rosulek, M.: Lincrypt: a model for practical cryptography. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 416–445. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_15
2. de Haan, L., Gennick, J.: Nulls: Nothing to Worry About. Oracle Magazine (2005). <http://www.oracle.com/technetwork/issue-archive/2005/05-jul/045sql-097727.html/>

3. Goldreich, O.: Foundations of Cryptography: Volume 2 - Basic Applications. Cambridge University Press, Cambridge (2004)
4. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game - a completeness theorem for protocols with honest majority. In: 19th STOC, pp. 218–229 (1987). For details see [3]
5. ISO/IEC. ISO/IEC 9075-2:2016 (2016). http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=63556/
6. Kleene, S.C.: Introduction to Metamathematics. Bibliotheca Mathematica (1952)
7. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
8. Lindell, Y., Pinkas, B.: A proof of Yao’s protocol for secure two-party computation. *J. Cryptol.* **22**(2), 161–188 (2009)
9. Microsoft. Null and Unknown (Transact-SQL). <https://msdn.microsoft.com/en-us/library/mt204037.aspx/>
10. MySQL. MySQL 5.7 Reference Manual 13.3.3: Logical Operators. <http://dev.mysql.com/doc/refman/5.7/en/logical-operators.html/>
11. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_15
12. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole: reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8
13. Yao, A.: How to generate and exchange secrets. In: 27th FOCS, pp. 162–167 (1986)



Efficient Covert Two-Party Computation

Stanislaw Jarecki^(✉)

University of California, Irvine, Irvine, USA
sjarecki@uci.edu

Abstract. Covert computation strengthens secure computation by hiding not only participants' inputs (up to what the protocol outputs reveal), but also the fact of computation taking place (up to the same constraint). Existing maliciously-secure covert computation protocols are orders of magnitude more costly than non-covert secure computation, and they are either non-constant round [5] or they use non-black-box simulation [10]. Moreover, constant-round covert computation with black-box simulation is impossible in the plain model [10].

We show that constant-round *Covert Two-Party Computation* (2PC) of general functions secure against malicious adversaries is possible with black-box simulation under DDH in the Common Reference String (CRS) model, where the impossibility result of [10] does not apply. Moreover, our protocol, a covert variant of a “cut-and-choose over garbled circuits” approach to constant-round 2PC, is in the same efficiency ballpark as standard, i.e. non-covert, 2PC protocols of this type. In addition, the proposed protocol is covert under concurrent self-composition.

An essential tool we use is a *covert* simulation-sound Conditional KEM (CKEM) for arithmetic languages in prime-order groups, which we realize in CRS or ROM at costs which are either the same (in ROM) or very close (in CRS) to known HVZK's for such languages.

1 Introduction

Covert computation addresses a security concern which is unusual for cryptography, namely how to hide the very fact of (secure) protocol execution. Such hiding of a protocol instance is possible if the communicating parties are connected by *steganographic* channels, which are implied by any channels with sufficient entropy [11]. Consider the simplest example of a steganographic channel, the *random channel*, a.k.a. a *random beacon*. In practice such channel can be implemented e.g. using protocol nonces, padding bits, time stamps, and various other communication (and cryptographic!) mechanisms which exhibit inherent (pseudo)entropy. (Works on steganographic communication, e.g. [11], show that random messages can be embedded into any channel with sufficient entropy.) Two parties sharing a random channel can use it to send protocol messages, and their presence cannot be distinguished from an a priori channel behavior if protocol messages are indistinguishable from random bitstrings. The participants must agree on which bits to interpret as protocol messages, but this can be public information since the decoding is indistinguishable from random.

Covert computation was formalized for the two-party honest-but-curious setting by Von Ahn et al. [28], and then generalized (and re-formulated) to the multi-party and malicious adversary setting by Chandran et al. [5], as a protocol that lets the participants securely compute the desired functionality on their joint inputs, with the additional property that each participant cannot distinguish the others from random beacons, i.e. entities that send out random bitstrings of fixed length instead of prescribed protocol messages, unless the function output implies participants' presence. Technically, in covert computation the computed function outputs an additional *reveal* bit: If this bit is 0 then each participant remains indistinguishable from a random beacon to the others, and if the reveal bit is 1 then the participants learn the function output, and in particular learn that a computation took place, i.e. that they were interacting not with random beacons but with counterparties executing the same protocol.

Q & A on Covert Computation. *Motivation:* Who wants to compute a function while hiding this very fact from (some) potential protocol participants? A generic example is authentication whose participants want to remain undetectable except to counter-parties whose inputs (certificates, secrets, passwords, gathered observations) match their authentication policy: If two spies search for one another in a foreign country, they want to do so while preventing anyone from detecting their authentication attempts. If the spies authenticated each other using covert computation, the only way their presence can be detected is by an active attacker whose inputs are those which the spies search for.

Random Channels: If protocol parties were not communicating by default, it would always be possible to detect a protocol party by just observing that it sends out messages, and observing their number and size should normally suffice to conclude what protocol this party follows. This is why covert protocol participants must have access to channels with some inherent entropy. A network entity cannot hide the fact that it sends out messages, but if the normal communication they emit exhibits some entropy (e.g. in protocol nonces, timing, padding, audio/video signals) then this entropy can be used to create a steganographic channel, and such channel can carry covert MPC protocol messages.

Covert MPC vs. Steganography: Covert MPC does not trivially follow by using steganography [11] to establish covert communication channels between potential protocol participants and running standard MPC over them. First, covert channels require prior key distribution which is not always possible, e.g. in the general authentication application above. Second, even if potential participants did have pre-shared keys, they might still want to hide whether or not they engage in a given protocol instance based on their on-line inputs.

Covert MPC vs. Secure MPC: Secure computation is believed to conceptualize every security task: Whatever security property we want to achieve, we can abstract it as a secure computation of an idealized functionality, and we can

achieve it by MPC for this functionality. However, secure computation does leak extra bit of information, because it does not hide whether some entity engages in the protocol, and in some applications, this is essential information. *Covert computation* strengthens secure computation to hide this remaining bit, and ensures undetectability of protocol participation even to active participants except (and this “escape clause” seems unavoidable) if the computation determines that its outputs, and hence also the fact of protocol participation, should be revealed. We show that, assuming CRS, this strengthening of secure computation to covertness can be achieved in the two-party case in constant rounds with black-box simulation under standard assumptions. Moreover, we achieve it at the cost which is in the same ballpark as the cost of known standard, i.e. non-covert, constant-round 2PC based on Yao’s garbled circuits [29], so covertness in a sense comes “for free”. As a side-benefit, the tools we use for covert enforcement of honest protocol behavior can be re-used in other covert protocols, e.g. for specific functions of interest.

Covert Computation as a Tool: Covert computation can also be a protocol tool with surprising applications, although we are aware of only one such case so far. Cho et al. [6] generalized a construction of Manulis et al. [20] to compile secure computation protocol for so-called “single instance” functionality, e.g. a pair-wise authentication by parties holding certificates issued by the same authority, into its “multiple instance” version, e.g. where each party holds a set of certificates from multiple authorities. The compiler has only linear cost, and it works by encoding messages of n instances of the single-instance protocol as points on an n -degree polynomial. Crucially, no one should distinguish which points encode valid messages until the single-instance functionality reveals its output, and [6] show that the compiler works for general functionalities *if* the single-instance protocol is *covert*.

Prior Work on Covert Computation. Von Ahn et al. [28] proposed the first covert two-party computation (2PC) protocol. Their protocol performed $O(\tau)$ repetitions, for τ a security parameter, of Yao’s garbled circuit evaluation (with the circuit extended to compute an additional hash function), but this protocol guaranteed only secrecy against malicious participants, and not output correctness. Chandran et al. [5] reformulated covert computation to guarantee output correctness and generalized it to multi-party computation, but their protocol was also non-constant-round, and its efficiency was *several orders of magnitude over known non-covert MPC protocols*: Each party was covertly proving that it followed a GMW MPC by casting it as an instance of a Hamiltonian Cycle problem, and that proof internally used Yao’s garbled circuits for checking correctness of committed values. Goyal and Jain [10] showed that non-constant-round protocols are necessary to achieve covert computation with black-box simulation against malicious adversaries, at least in the plain MPC model, i.e., without access to some trusted parameters. [10] also showed a constant-round covert MPC with non-black-box simulation (and bounded concurrency), but their protocol re-uses the above-mentioned components of [5], and is therefore just as costly.

Whereas the covert MPC protocols of [5, 10] assumed the plain computation model, recently Cho et al. [6] exhibited practical constant-round covert computation protocols secure against active adversaries, for two specific two-party functionalities, namely string equality and set intersection, in the Random Oracle Model (ROM). (Moreover, [6] strengthened the definition of covert computation of [5] to unbounded concurrent self-composition, which we adopt here.) However, their constructions are custom-made and it is not clear how they can be extended to computation of general functions. In other related work, Jarecki [13] showed a constant-round covert Authenticated Key Exchange (AKE) with $O(1)$ public key operations, but this protocol satisfied a game-based AKE definition, and it was not a covert secure computation of any function.

Main Contribution: Efficient Constant-Round Covert 2PC in CRS.

This leaves a natural open question whether assuming some relaxation in the trust model (necessary in view of the negative result of [10]) general two-party functions can be computed covertly by a constant-round protocol with black-box simulation, or even, better, by a protocol whose assumptions, the security guarantees, and efficiency, are all comparable to those of the currently known constant-round standard, i.e. non-covert, secure 2PC protocols. We answer these questions affirmatively assuming the Common Reference String (CRS) model and the Decisional Diffie-Hellman (DDH) assumption.¹ In this setting we show a covert 2PC protocol which follows the well-known paradigm for standard, i.e. non-covert, constant-round secure 2PC, initiated by [18, 21] and followed in numerous works. Namely, we use the cut-and-choose technique over $O(\tau)$ copies of Yao’s garbled circuit, but we do so using *efficient* covert equivalents of standard protocol tools for enforcing honest protocol behavior, like extractable commitments and simulation-sound arguments. Moreover, the protocol is secure under concurrent composition. Remarkably, our covert 2PC protocol is roughly in the same efficiency ballpark as the non-covert secure 2PC protocols of this type. For example, for a *one-sided output* function with n -bit inputs with a Boolean circuit with c gates, our protocol requires 5 rounds, $O(n\tau)$ exponentiations, and a transfer of $O(n\tau)$ group elements and $O(c\tau)$ symmetric ciphertexts.

Challenge of Malicious Security for Covert Computation. Assuming random channels, covert *communication* is essentially as easy as secure communication: Under standard assumptions symmetric encryption modes have ciphertexts which are indistinguishable from random bitstrings. Several known public-key encryption schemes, e.g. Cramer-Shoup encryption [8], also have ciphertexts that are indistinguishable from a tuple of random group elements (under DDH), and random group elements in a prime-order subgroup of modular residues are easy to encode as random bitstrings. Von Ahn et al. [28] show that General covert *computation in the honest-but-curious setting* is also not more difficult

¹ We note that a work in progress by Couteau [7] aims to show a corresponding result for covert MPC protocols built using the non-constant-round GMW paradigm.

than standard secure computation, because (1) Yao’s garbled circuit construction can be adjusted so that a garbled circuit for c -gates looks like $4c$ random ciphertexts even to the evaluator (except for whatever is revealed by the output, but that can be set to a random string if the “reveal bit” in the output evaluates to 0), and (2) because a DDH-based OT of Naor-Pinkas Oblivious Transfer (OT) [22] is covert under the same DDH assumption.

However, it is harder to achieve covert 2PC/MPC protocols secure against *malicious* adversaries because of the lack of efficient covert counterparts to standard mechanisms for enforcing honest protocol behavior. For example, the tools often used to enforce honest behavior in Yao’s garbled circuit protocol are (1) Zero-Knowledge (ZK) proofs, e.g. to show that consistent inputs are input into multiple garbled circuit instances and into the OT, and (2) opening a commitment to show that the committed value is correctly formed. Either tool is publicly verifiable and thus violates covertness.

Second Contribution: Efficient Covert Simulation-Sound CKEM’s.

Our tool for enforcing honest protocol behavior is an efficient covert *Conditional Key Encapsulation Mechanism* (CKEM) for a wide class of discrete-log-based languages, including statements that a Cramer-Shoup ciphertext [8] computed correctly, that an encrypted value is a bit, or that a commitment decommits to a given plaintext. A CKEM is a variant of Conditional OT [9], and an interactive counterpart of Smooth Projective Hash Functions (SPHF): A CKEM for language L is a protocol which allows a sender S with input x to transmit a random key K to receiver R with input (x, w) if and only if w is a witness for x in L . A *covert* CKEM [5, 10, 13] assures that an interaction with either S or R is indistinguishable from a random beacon. In particular, even given the witness w for x the receiver cannot distinguish $S(x)$ from random: It can compute key K of this CKEM instance, but this key is random and should not be sender distinguishable from a randomness source. Hence, covert CKEMs can provide a covert counterpart to Zero-Knowledge Proofs: Instead of A proving to B that its protocol messages are correct, B and A run a covert CKEM for the same language as resp. S and R , and use key K to (covertly) encrypt subsequent protocol messages: If A ’s messages were malformed, covert CKEM assures that key K is pseudorandom to A and all subsequent messages of B are pseudorandom.

To be most useful as a protocol tool, a covert CKEM should be *concurrent*, *simulatable*, and *simulation sound* (a *proof of knowledge* property can help too, but our covert 2PC protocol does not utilize it). We exhibit constructions of such covert CKEM’s for two classes of languages. The first class are so-called *Linear Map Image* (LMI) languages, i.e. languages whose statements can be represented as pair (C, M) of a vector C and a matrix M of group elements s.t. C belongs to a range of a linear map $f_M(w) = w \cdot M$ defined by M (where scalar multiplication stands for a homomorphic one-way function, e.g. exponentiation). The second class are languages with so-called Σ -protocols, i.e. three-round public-coin HVZK proofs with special soundness and zero-knowledge properties which are typically satisfied by e.g. proofs of arithmetic statements in prime-order groups.

We overview our covert CKEM constructions below, but in the nutshell we show (1) a 2-round CKEM in CRS for LMI languages defined by a full row rank matrix M , whose cost is about 2–4 times that of the underlying HVZK, (2) a 2-round CKEM in ROM for Σ -protocol languages whose cost almost matches the underlying HVZK, and (3) a 4-round CKEM in CRS for Σ -protocol languages with a larger but still additive overhead over the underlying HVZK.

Prior Work on Covert CKEM’s. Covert CKEM was introduced as *Zero-Knowledge Send* (ZKSend) by Chandran et al. [5], and strengthened to proof of knowledge, simulation-soundness, and bounded concurrency in [10], but it is not clear how these constructions can yield *practical* covert CKEM’s for Σ -protocol or LMI languages. The ZKSend of [5] reduces L to a Hamiltonian Cycle (HC) instance, replaces the verification step in Blum’s binary-challenge ZK proof for HC with covert garbled circuit evaluation of this step, and repeats this $O(\tau)$ times for negligible soundness error. The ZKSend of [10] follows this paradigm using the ZK argument for NP by Pass [23]. Both constructions aimed at feasibility of covert CKEM for NP, while we want covert CKEM’s at costs comparable to the underlying HVZK’s, for LMI or Σ -protocol languages. Moreover, much of the complexity in the constant-round covert CKEM of [10] was to assure covertness (and bounded concurrency) with careful usage of rewinding in the simulation. Indeed, the negative result in [10] for constant-round covertness with black-box simulation was due to rewinding necessary in simulation in the plain model. By contrast, if we assume CRS, an assumption without which we do not know how to achieve even *secure* constant-round protocols with unbounded concurrency and/or practical efficiency, we can get concurrency with straight-line simulation using a CRS trapdoor, and the negative result of [10] no longer applies.

Efficient Covert Simulation-Sound CKEM’s: A Closer Look. One starting point for a practical straight-line simulatable covert CKEM for an LMI language can be an efficient SPHF, because an SPHF for an LMI language defined by a full row rank matrix is covert. However, SPHF by itself is not *simulatable*: Note that a simulator who plays the role of an honest party typically does not form its messages as the honest party would, which would make the statement in the SPHF instance (that the honest party’s protocol messages are well-formed) incorrect in the simulation. Hence, by SPHF security (a.k.a. smoothness), the simulator could not recover key K , and would fail in simulation of subsequent protocol rounds. To amend precisely this shortcoming of SPHF’s, Benhamouda et al. [3] upgraded SPHF’s for LMI languages to CKEM’s in CRS with (1) (*concurrent*) *simulatability*, i.e. the ability for the simulator in possession of the CRS trapdoor to derive key K even on the wrong statement $x \notin L$; and (2) *simulation-soundness*, i.e. that a cheating receiver cannot recover S ’s key K for an instance executing on a wrong statement $x \notin L$ even if the adversary concurrently engages with the simulator who recovers keys corresponding to multiple protocol instances running on any *other* wrong statements $x' \notin L$. Both are needed of ZK proofs in a compiler from (concurrent) honest-but-curious MPC

to (concurrent) maliciously-secure MPC, and [3] showed that this compiler works if ZKP's are replaced by CKEM's. However, their goal was to reduce rounds in MPC by replacing ZKP's with CKEM's, and *not*, as in our case, to assure MPC covertness. In particular, their CKEM's are not covert: To assure straight-line extraction [3] modify LMI statement M, C s.t. $C = w \cdot M$ for both the original witness w_R and the simulator's CRS trapdoor w_{td} . This way the receiver and the simulator can both compute hash value $C \cdot hk$ from projection $hp = M \cdot hk$, respectively as $w_R \cdot hp$ and $w_{td} \cdot hp$. However, this holds only if hp is formed correctly, i.e. if $hp^T \in \text{span}(M^T)$, hence [3] run add a secondary SPHF where R sends $tp = M^T \cdot tk$ for random tk to S , who can compute hash value $hp^T \cdot tk$ as $hk^T \cdot tp$ if and only if $hp^T = hk^T \cdot M^T$. Consequently, the CKEM of [3] publicly sends $hp = M \cdot hk$ and $tp = M^T \cdot tk$. Matrix M is typically full row-rank, assuring that $M \cdot hk$ is a random vector in the column space, but it is not full column-rank, which makes $M^T \cdot tk$ not uniformly random. However, we show how to modify matrix M s.t. the secondary projection $M^T \cdot tk$ is *pseudo-random* during simulation.

A different route towards Covert CKEM's is to take as a starting point an efficient compiler from Σ -protocol to covert CKEM of [13]. The CKEM notion of [13] is covert and proof-of-knowledge, but not simulatable and simulation-sound. The covert CKEM of [13] bears some similarity to the ZKSend of [5]: Both start from an HVZK proof (resp. Σ -protocol and Blum's HVZK for HC), identify some proof messages which are already pseudorandom, replace the offending non-covert message with its commitment, and replace the verification equation with some covert gadget: [5] commit to the final response in Blum's proof and use garbled circuit for the verification step on the committed plaintext, and [13] commit to the prover's first message, and uses the "special simulation" property of a Σ -protocol, i.e. that the first message can be computed from the other two, to replace the verification step with a covert SPHF that checks that the committed first message equals to this single verification-passing value. The benefit of the compiler of [13], in contrast to the above approach, is that it adds only a (small) additive overhead to the underlying Σ -protocol. We show that the 2-round instantiation of this compiler is simulatable and simulation-sound assuming ROM, and that the 4-round version of this compiler can be modified so that the result is covert simulatable and simulation-sound in CRS.

Organization. In Sect. 2 we introduce covertness-related notation. In Sect. 3 we define concurrent covert 2PC for arbitrary functions. In Sect. 4 we define covert counterparts to standard protocol building blocks, including covert CCA PKE, commitment, OT, HbC-secure circuit garbling, and SPHF's. In Sect. 5 we define covert CKEM's. In Sect. 6 we discuss our covert CKEM's constructions. Finally, in Sect. 7 we present our concurrent covert 2PC protocol.

2 Preliminaries

Notation. If a, b are bitstrings then $|a|$ is the length of a , $a|b$ is the concatenation of strings a and b , and $a[i]$ is the i -th bit of a . If n is an integer then $[n] = \{1, \dots, n\}$. We write $y \leftarrow P(x)$ when y is an output of a (randomized) procedure P on input x , and $y \leftarrow S$ when y is sampled from uniform distribution over set S . We write $y \in P(x)$ if there is randomness r s.t. $P(x; r)$ outputs y . We say $(a, b) \leftarrow [A(x), B(y)]$ if a, b are the local outputs of algorithms resp. A, B interacting on local inputs resp. x, y . If L is a language in NP then $\mathcal{R}[L]$ is a relation s.t. $(x, w) \in \mathcal{R}[L]$ if w is an efficiently verifiable witness for $x \in L$. If $\mathbf{ks} = \{(k^{i,0}, k^{i,1})\}_{i \in [n]}$, i.e. \mathbf{ks} is a sequence of n pairs of bitstrings, and $x \in \{0, 1\}^n$, then $\mathbf{ks}[x]$ denotes a *selection* of bitstrings from the n pairs in \mathbf{ks} according to the n bits of x , namely $\mathbf{ks}[x] = \{k^{i,x[i]}\}_{i \in [n]}$.

We call two-party protocol (A, B) *regular* if the number of rounds and length of all messages is a function of the security parameter, and not the parties' inputs. If P is an interactive algorithm in a regular two-party protocol then $P^{\mathfrak{s}(\tau)}$ denotes a random beacon corresponding to P , which sends random bitstrings of the same length as P 's messages in every protocol round. If P is an interactive algorithm then $P_{\&Out}(x)$ is a wrapper which runs $P(x)$ and includes P 's final local output in its last message. For any algorithm Setup and oracles P_0, P_1 we say that $\{\mathcal{A}^{P_0(x_0)}(z)\} \approx \{\mathcal{A}^{P_1(x_1)}(z)\}$ for $(x_0, x_1, z) \leftarrow \text{Setup}(1^\tau)$ if for every efficient \mathcal{A} quantity $|p_{\mathcal{A}}^0 - p_{\mathcal{A}}^1|$ is negligible where $p_{\mathcal{A}}^b = \Pr[1 \leftarrow \mathcal{A}^{P_b(x_b)}(z) \mid (x_0, x_1, z) \leftarrow \text{Setup}(1^\tau)]$, where the probability goes over the coins of Setup , \mathcal{A} , and P_b .

Covert Encodings. In our protocols all communicated values are either random fixed-size bitstrings, or random integers from some range \mathbb{Z}_n , or random elements of a prime-order group G . In the latter two cases what is sent on the wire are not the values themselves but their covert encodings. A covert encoding of domain D is a randomized function $\text{EC} : D \rightarrow \{0, 1\}^{p(\tau)}$ defined for some polynomial p , s.t. a random variable $\{\text{EC}(a; r)\}$, induced by random a in D and random r , is statistically close to a random bitstring of length $p(\tau)$. Moreover, there must exist a decoding procedure DC s.t. $\text{DC}(\text{EC}(a; r)) = a$ for all $a \in D$ and all r . For example, if domain D is an integer range \mathbb{Z}_n then $\text{EC}(a)$ can pick $r \leftarrow \mathbb{Z}_R$ for $R = \lceil 2^{|n|+\tau}/n \rceil$ and output $a + n \cdot r$ (over integers), while $\text{DC}(v)$ outputs $v \bmod n$. If the domain D is a subgroup G of order p in a multiplicative group \mathbb{Z}_q^* of residues modulo q for $q = p \cdot t + 1$ s.t. $\text{gcd}(p, t) = 1$, then $\text{EC}(a)$ can pick $b \leftarrow \mathbb{Z}_q$, compute $v = (a \cdot (b)^p) \bmod q$, and then apply the encoding for integer range \mathbb{Z}_q to v . The corresponding decoding first decodes v and then outputs $w^s \bmod q$ for $w = v^t \bmod q$ and $s = t^{-1} \bmod p$.

3 Defining Concurrent Covert Two-Party Computation

We provide the definition of concurrent covert computation of two-party functions, which is a close variant of the definition which appeared recently in [6].

Intuitively, the differences between the covert computation of a two-party functionality F and the secure computation for F is that (1) F 's inputs and outputs are extended to include a special sign \perp designating non-participation; (2) F is restricted to output a non-participation symbol \perp to each party if the input of either party is \perp ; and (3) the real-world protocol of either party on the non-participation input \perp is fixed as a “random beacon”, i.e. a protocol which sends out random bitstrings of fixed length independently of the messages it receives.

The definition of concurrent covert computation of [6], which we recall (and refine) below, follows the definition of stand-alone (i.e. “single-shot”) covert computation given by Chandran et al. [5], here restricted to the two-party case. The definition casts this notion in the framework of universal composability (UC) by Canetti [4], but the composability guarantee it implies is restricted to concurrent self-composition because it guarantees only self-composability of covert computation for *functions*, and not for general reactive functionalities as in the case of UC definition [4]. The reason for this restriction is two-fold: First, concurrent covert computation for arbitrary efficiently computable functions already provides a significant upgrade over the “single-shot” covert computation notion of [5], and achieving it efficiently presents sufficient technical challenges that justify focusing on this restricted notion. Secondly, composing functionally distinct covert protocols poses conceptual challenges: Consider a protocol Π implemented by a protocol Π_1 which runs Π_2 as a subroutine, and note that the outputs of subroutine Π_2 can reveal the participation of an honest party in Π before Π completes. Here we focus on concurrent composition of covert computation of two-party function, and leave development of a framework for fully composable covert computation for future work.

Ideal and Real Models. The definition of the *ideal model* is the UC analogue of the ideal model of Chandran et al. [5], except that composability guarantees are restricted to self-composition. Covert computation is defined by functionality $F_{C(f,g)}$ shown in Fig. 1, where f, g are functions defined on pairs of bitstrings. As in [5] function g is an admission function, i.e. if $g(x, y) = 0$ then functionality $F_{C(f,g)}$ returns \perp to both parties, and f is the “real function” i.e. if $g(x, y) \neq 0$ then functionality $F_{C(f,g)}$ prepares A 's output as z and B 's output as v where $(z, v) = f(x, y)$. We note that f and g can be randomized functions, in which case functionality $F_{C(f,g)}$ picks the randomness which is appended to input (x, y) before computing g and f . The ideal process involves functionality $F_{C(f,g)}$, an ideal process adversary \mathcal{A}^* , an environment \mathcal{Z} with some auxiliary input z , and a set of dummy parties, any number of which can be (statically) corrupted. Each party can specify its input to some instance of $F_{C(f,g)}$, which is either a bitstring or a special symbol \perp indicating that there is no party which will participate in a given role, e.g. a requester or responder in this protocol instance. The *real model* is exactly as in the standard UC security model, except that the protocol of each real-world uncorrupted party which runs on input \perp is a-priori specified as a random beacon protocol, i.e. such party sends out random bitstrings of lengths appropriate for a given protocol round.

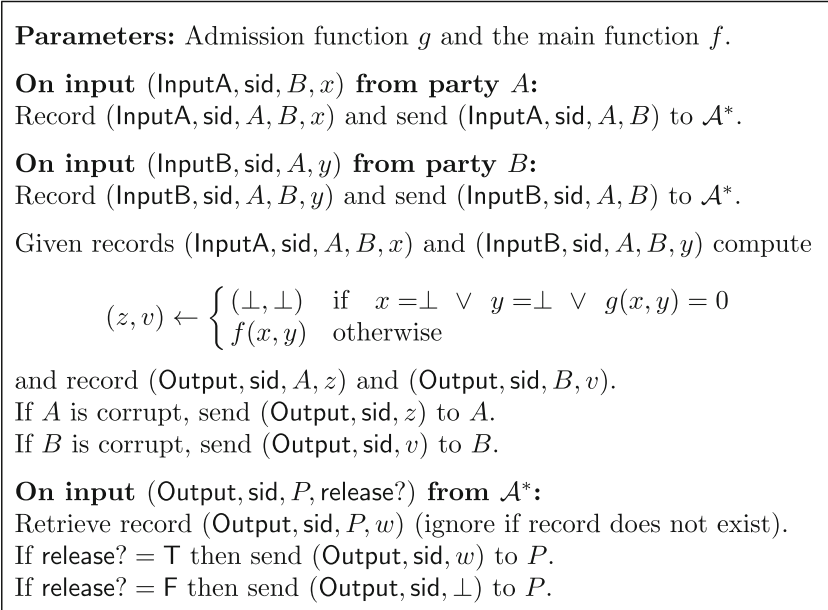


Fig. 1. Covert 2-party function computation functionality $F_{C(f,g)}$

Let $\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux}, r)$ denote the output of environment \mathcal{Z} after interacting in the ideal world with adversary \mathcal{A}^* and functionality $F = F_{C(f,g)}$, on security parameter τ , auxiliary input aux , and random input $r = (r_{\mathcal{Z}}, r_{\mathcal{A}^*}, r_F)$, as described above. Let $\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux})$ be the random variable $\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux}; r)$ when r is uniformly chosen. We denote the random variable $\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux})$ as $\{\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux})\}_{\tau \in \mathbb{N}; \text{aux} \in \{0,1\}^*}$. Correspondingly we let $\text{Real}_{\Pi, \text{Adv}, \mathcal{Z}}(\tau, \text{aux}; r)$ be the output of \mathcal{Z} after interacting with a real-world adversary Adv and parties running protocol Π on security parameter τ , input aux , and random tapes $r = (r_{\mathcal{Z}}, r_{\text{Adv}}, r_A, r_B)$. In parallel to the ideal model, we define the corresponding random variable $\{\text{Real}_{\Pi, \text{Adv}, F}(\tau, \text{aux})\}_{\tau \in \mathbb{N}; \text{aux} \in \{0,1\}^*}$.

Definition 1. Protocol Π realizes the concurrent two-party covert computation functionality $F = F_{C(f,g)}$ if for any efficient adversary Adv there exists an efficient ideal-world adversary \mathcal{A}^* such that for any efficient environment \mathcal{Z} ,

$$\{\text{Ideal}_{F, \mathcal{A}^*, \mathcal{Z}}(\tau, \text{aux})\}_{\tau \in \mathbb{N}; \text{aux} \in \{0,1\}^*} \stackrel{c}{\approx} \{\text{Real}_{\Pi, \text{Adv}, F}(\tau, \text{aux})\}_{\tau \in \mathbb{N}; \text{aux} \in \{0,1\}^*}$$

Notes on Functionality $F_{C(f,g)}$. Functionality $F_{C(f,g)}$ in Fig. 1 is realizable only assuming secure channels. Without secure channels the adversary could hijack a protocol session an honest player wants to execute with some intended counterparty. However, the secure channel assumption does not substantially change the complexity of the protocol problem because the intended counterparty can itself be corrupted and follow an adversarial protocol. The second

point we want to make is that functionality $F_{C(f,g)}$ always delivers the output first to a corrupted party, whether it is party A or B , and if this output is not a non-participation symbol \perp then in both cases the corrupted party can decide if the correct computation output should also be delivered to its (honest) counterparty or the honest counterparty's output will be modified to \perp . (Note that if an output of a corrupt party, say A , is \perp then B 's output is also \perp , hence it does not matter in this case whether the adversary sends $(\text{Output}, T, \text{sid})$ or $(\text{Output}, F, \text{sid})$.) Any constant-round protocol without a trusted party *must be unfair* in the sense that the party which speaks last gets its output but can prevent the delivery of an output to its counterparty. However, functionality $F_{C(f,g)}$ affords this unfair advantage to both the corrupt requester and the corrupt responder. Indeed, a concrete protocol Π_{COMP} presented in Sect. 7 which realizes this functionality allows the corrupt party A to learn its output z and stop B from learning anything about its output v (simply by aborting before sending its last message to B). However, this protocol also allows the corrupt party B to prevent party A from being able to decide if its output z (learned in step A2 in Fig. 3) is an output of $f(x, y)$ or a random value induced from an interaction with a random beacon: Only B 's final message can confirm which is the case for A , but a corrupt B can send this message incorrectly, in which case an honest A will dismiss the tentative value z it computed and output \perp instead. We leave achieving $O(1)$ -round covert protocols with one-sided fairness, or two-side fairness, e.g. using an off-line escrow authority, to future work.

4 Covert Protocol Building Blocks

CCA-Covert Public Key Encryption. Covertness of a public key encryption scheme in a Chosen-Ciphertext Attack, or *CCA covertness* for short, is a generalization of CCA security: Instead of requiring that ciphertexts of two challenge messages are indistinguishable from each other, we require that a ciphertext on any (single) challenge message is indistinguishable from a random bitstring, even in the presence of a decryption oracle. For technical reasons it suffices if interaction with the real PKE scheme is indistinguishable from an interaction with a simulator who not only replaces a challenge ciphertext with a random string but also might follow an alternative key generation and decryption strategy.

Formally, we call a (labeled) PKE scheme $(\text{Kg}, \text{E}, \text{D})$ *CCA covert* if there exist polynomial n s.t. for any efficient algorithm \mathcal{A} , quantity $\text{Adv}_{\mathcal{A}}(\tau) = |p_{\mathcal{A}}^0(\tau) - p_{\mathcal{A}}^1(\tau)|$ is negligible, where $p_{\mathcal{A}}^b(\tau)$ is the probability that $b' = 1$ in the following game: Generate $(\text{pk}, \text{sk}) \leftarrow \text{Kg}(1^\tau)$, and let $\mathcal{A}^{\text{D}(\text{sk}, \cdot, \cdot)}(\text{pk})$ output an encryption challenge (m^*, ℓ^*) . If $b = 1$ then set $\text{ct}^* \leftarrow \text{E}(\text{pk}, m^*, \ell^*)$, and if $b = 0$ then pick ct^* as a random string of length $n(\tau)$. In either case set $b' \leftarrow \mathcal{A}^{\text{D}(\text{sk}, \cdot, \cdot)}$, where oracle $\text{D}(\text{sk}, \cdot, \cdot)$ returns $\text{D}(\text{sk}, \text{ct}, \ell)$ on any ciphertext, label pair s.t. $(\text{ct}, \ell) \neq (\text{ct}^*, \ell^*)$.

Notice that by transitivity of indistinguishability if PKE is CCA-covert then it is also CCA-secure. The other direction does not hold in general, but many known CCA-secure PKE's are nevertheless also CCA-covert, including RSA OAEP and Cramer-Shoup PKE [8]. We will use here the latter scheme because

its arithmetic structure can be utilized for efficient covert OT (see below) and efficient covert CKEM's on associated languages (e.g. that a ciphertext encrypts a given plaintext). In the full version [14] we show that the proof of CCA security of Cramer-Shoup PKE under the DDH assumption [8] can be extended to imply its CCA covertness. For notational convenience we assume that the key generation Kg picks the group setting (g, G, p) as a *deterministic* function of security parameter τ , and we restrict the message space to group G , since this is how we use this PKE in our covert 2PC protocol, but it can be extended to general message space using covert symmetric encryption.

Cramer-Shoup PKE (for message space G) works as follows: $\text{Kg}(1^\tau)$ chooses generator g of group G of prime order p of appropriate length, sets a collision-resistant hash function H , picks $(x_1, x_2, y_1, y_2, z) \leftarrow (\mathbb{Z}_p^*)^5$, $(g_1, g_2) \leftarrow (G \setminus 1)^2$, sets $(c, d, h) \leftarrow (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^z)$, and outputs $\text{sk} = ((g, G, p, H), x_1, x_2, y_1, y_2, z)$ and $\text{pk} = ((g, G, p, H), g_1, g_2, c, d, h)$. Encryption $E_{\text{pk}}(m, \ell)$, for $m \in G$, picks $r \leftarrow \mathbb{Z}_p$, sets $(u_1, u_2, e) \leftarrow (g_1^r, g_2^r, m \cdot h^r)$, $\xi \leftarrow H(\ell, u_1, u_2, e)$, $v \leftarrow (cd^\xi)^r$, and outputs $\text{ct} = (u_1, u_2, e, v)$. Decryption $D_{\text{sk}}((u_1, u_2, e, v), \ell)$ re-computes ξ , and outputs $m = e \cdot u_1^{-1}$ if $v = u_1^{x_1 + \xi \cdot y_1} u_2^{x_2 + \xi \cdot y_2}$ and \perp otherwise.

Covert Non-malleable Commitments. It is well-known that CCA-secure PKE implements non-malleable commitment. However, to stress that sometimes no one (including the simulator) needs to decrypt, we define commitment $\text{Com}_{\text{pk}}(m)$ as a syntactic sugar for $E_{\text{pk}}(H(m))$ where H is a collision-resistant hash onto G , but we will pass on defining a notion of covert commitment, relying instead directly on the fact that $\text{Com}_{\text{pk}}(m)$ stands for $E_{\text{pk}}(H(m))$.

Covert Oblivious Transfer. Von Ahn et al. [28] used a covert version of Naor-Pinkas OT [22] for their covert 2PC secure against honest-but-curious adversaries. Here we will use a covert version of the OT of Aiello et al. [2] instead because it is compatible with CCA-covert Cramer-Shoup encryption and covert CKEM's of Sect. 6. Let E be the Cramer-Shoup encryption and let $\text{pk} = ((g, G, p, H), g_1, g_2, c, d, h)$. Define a 2-message OT scheme $(E, \text{OTrsp}, \text{OTfin})$ on Rec 's input b , Snd 's input $m_0, m_1 \in G$, and a public label ℓ as follows:

- (1) Rec 's first message to Snd is $\text{ct} = (u_1, u_2, e, v) = E_{\text{pk}}(g^b, \ell; r)$ for $r \leftarrow \mathbb{Z}_p$.
- (2) Snd 's response computation, denoted $\text{OTrsp}_{\text{pk}}(\text{ct}, m_0, m_1; r')$, outputs $\text{otr} = \{s_i, t_i\}_{i=0,1}$ for $(s_i, t_i) = (g_1^{\alpha_i} h^{\beta_i}, u_1^{\alpha_i} (e/g^i)^{\beta_i} m_i)$ and $r' = \{\alpha_i, \beta_i\}_{i=0,1} \leftarrow \mathbb{Z}_p^4$.
- (3) Rec 's output computation, denoted $\text{OTfin}_{\text{pk}}(b, r, \text{otr})$, outputs $m = t_b \cdot (s_b)^{-r}$.

The above OT is covert for random payloads in the following sense: First, the Rec 's message is indistinguishable from random even on access to the decryption oracle $D_{\text{sk}}(\cdot, \cdot)$; Secondly, Snd 's message is indistinguishable from random for payloads (m_0, m_1) random in G^2 . (Note that if (m_0, m_1) were non-random then the Rec 's output would suffice to distinguish OTrsp and $\text{OTrsp}^{\mathfrak{s}(\tau)}$.)

Covert Garbled Circuits. Von Ahn et al. [28] shows a covert version of Yao’s garbling $\text{GCgen}(f)$ for any $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Procedure $\text{GCgen}(f)$ outputs (1) a vector of input wire keys $\text{ks} = \{k^{w,b}\}_{w \in [n], b \in \{0,1\}}$ where n is the bitlength of arguments to f , and (2) a vector gc of $4|C|$ covert symmetric encryption ciphertexts, where $|C|$ is the number of gates in a Boolean circuit for f . The corresponding evaluation procedure Eval_f outputs $f(x)$ given gc and $\text{ks}[x] = \{k^{i,x[i]}\}_{i \in [n]}$, for (gc, ks) output by $\text{GCgen}(f)$ and $x \in \{0, 1\}^n$. Let $m' = 4|C|\tau + n\tau$. The notion of a *covert garbling* defined by [28] and satisfied by their variant of Yao’s garbling scheme, is that for any function f , any distribution D over f ’s inputs, and any efficient algorithm \mathcal{A} , there is an efficient algorithm \mathcal{A}^* s.t. $|\text{Adv}_{\mathcal{A}} - \text{Adv}_{\mathcal{A}^*}|$ is negligible, where:

$$\begin{aligned} \text{Adv}_{\mathcal{A}} &= |\Pr[1 \leftarrow \mathcal{A}(\{\text{gc}, \text{ks}[x]\})]_{x \leftarrow D, (\text{gc}, \text{ks}) \leftarrow \text{GCgen}(f)} - \Pr[1 \leftarrow \mathcal{A}(r)]_{r \leftarrow \{0,1\}^{m'}}| \\ \text{Adv}_{\mathcal{A}^*} &= |\Pr[1 \leftarrow \mathcal{A}^*(f(x))]_{x \leftarrow D} - \Pr[1 \leftarrow \mathcal{A}^*(r)]_{r \leftarrow \{0,1\}^m}| \end{aligned}$$

In other words, for any function f and distribution D over its inputs, the garbled circuit for f together with the set of wire keys $\text{ks}[x]$ defined for input x sampled from D , are (in)distinguishable from a random string *to the same degree* as function outputs $f(x)$ for $x \leftarrow D$. In particular, if f and D are such that $\{f(x)\}_{x \leftarrow D}$ is indistinguishable from random, then so is $\{\text{gc}, \text{ks}[x]\}_{(\text{gc}, \text{ks}) \leftarrow \text{GCgen}(f), x \leftarrow D}$.

SPHF’s. We define a Smooth Projective Hash Function (SPHF) for language family L parametrized by π as a tuple $(\text{PG}, \text{KG}, \text{Hash}, \text{PHash})$ s.t. $\text{PG}(1^\tau)$ generates parameters π and a trapdoor td which allows for efficient testing of membership in $L(\pi)$, $\text{KG}(\pi, x)$ generates key hk together with a *key projection* hp (here we use the SPHF notion of [25], for alternative formulation see e.g. [15]) and $\text{Hash}(\pi, x, \text{hk})$ and $\text{PHash}(\pi, x, w, \text{hp})$ generate hash values denoted H and projH , respectively. SPHF *correctness* requires that $\text{Hash}(\pi, x, \text{hk}) = \text{PHash}(\pi, x, w, \text{hp})$ for all τ , all (π, td) output by $\text{PG}(1^\tau)$, all $(x, w) \in \mathcal{R}[L(\pi)]$, and all (hk, hp) output by $\text{KG}(\pi, x)$. In the context of our protocols SPHF values are elements of group G uniquely defined by security parameter τ via the Cramer-Shoup key generation procedure, hence we can define SPHF *smoothness* as that $(\text{hp}, \text{Hash}(\pi, x, \text{hk}))$ is distributed identically to (hp, r) for $r \leftarrow G$ and $(\text{hk}, \text{hp}) \leftarrow \text{KG}(\pi, x)$, for all π and $x \notin L(\pi)$. However, in our applications we need a stronger notion we call *covert smoothness*, namely that for some constant c , for all π and $x \notin L(\pi)$, pair $(\text{hp}, \text{Hash}(\pi, x, \text{hk}))$ for $(\text{hk}, \text{hp}) \leftarrow \text{KG}(\pi, x)$ is uniform in $G^c \times G$.

5 Covert Simulation-Sound Conditional KEM (CKEM)

Conditional Key Encapsulation Mechanism (CKEM). A Conditional KEM (CKEM) [13] is a KEM version of *Conditional Oblivious Transfer* (COT) [9]: A CKEM for language L is a protocol between two parties, a sender S and a receiver R , on S ’s input a statement x_S and R ’s input a (statement,witness) pair (x_R, w_R) . The outputs of S and R are respectively K_S and K_R s.t. K_S is a random string of τ bits, and $K_R = K_S$ if and only if $x_S = x_R$ and $(x_R, w_R) \in \mathcal{R}[L]$. CKEM

is an encryption counterpart of a zero-knowledge proof, where rather than having R use its witness w_R to prove to S that $x_S \in L$, here R establishes a session key K with S if and only if w_R is a witness for x_S in L . Because of this relation to zero-knowledge proofs we can use proof-system terminology to define CKEM security properties. In particular, we will refer to the CKEM security property that if $x \notin L$ then no efficient algorithm can compute K output by $S(x)$ as the *soundness* property.

Benhamouda et al. [3] considered a stronger notion of *Trapdoor CKEM*, which they called *Implicit Zero-Knowledge*. Namely, they extended the CKEM notion by a CRS generation procedure which together with public parameters generates a trapdoor td that allows an efficient simulator algorithm to compute the session key K_S output by a sender $S(x)$ for any x , including $x \notin L$. The existence of such simulator makes CKEM into a more versatile protocol building block. For example, trapdoor CKEM implies a zero-knowledge proof for the same language, if R simply returns the key K_R to S who accepts iff $K_R = K_S$. Indeed, following [3], we refer to the property that the simulator computes the same key as the honest receiver in the case $x \in L$ as the *zero-knowledge* property of a CKEM.

As in the case of zero-knowledge proofs, if multiple parties perform CKEM instances then it is useful to strengthen CKEM security properties to *simulation-soundness*, which requires that all instances executed by the corrupt players remain sound even in the presence of a simulator \mathcal{S} who uses its trapdoor to simulate the instances performed on behalf of the honest players. Simulation-soundness is closely related to non-malleability: If \mathcal{S} simulates a CKEM instance Π on $x \notin L$ then an efficient adversary must be unable to use protocol instance Π executed by \mathcal{S} to successfully complete another instance Π' of CKEM executed by a corrupt party for any $x' \notin L$.

To distinguish between different CKEM sessions the CKEM syntax must also be amended by *labels*, denoted ℓ , which play similar role as labels in CCA encryption. Formally, a CKEM scheme for language family L is a tuple of algorithms $(PG, TPG, Snd, Rec, TRec)$ s.t. parameter generation $PG(1^\tau)$ generates CRS parameter π , trapdoor parameter generation $TPG(1^\tau)$ generates π together with the simulation trapdoor td , and sender Snd , receiver Rec , and trapdoor receiver $TRec$ are interactive algorithms which run on local inputs respectively (π, x, ℓ) , (π, x, ℓ, w) , and (π, x, ℓ, td) , and each of them outputs a session key K as its local output. CKEM correctness requires that for all labels ℓ :

$$\forall (x, w) \in \mathcal{R}[L], [K_S, K_R] \leftarrow [Snd(\pi, x, \ell), Rec(\pi, x, \ell, w)] \Rightarrow K_S = K_R \quad (1)$$

$$\forall x, [K_S, K_R] \leftarrow [Snd(\pi, x, \ell), TRec(\pi, x, \ell, td)] \Rightarrow K_S = K_R \quad (2)$$

where (1) holds for all π generated by $PG(1^\tau)$ and (2) holds for all (π, td) generated by $TPG(1^\tau)$. Crucially, property (2) holds for all x , and not just for $x \in L$.

Covert CKEM. A *covert* CKEM was introduced as *Zero-Knowledge Send* (ZKSend) by Chandran et al. [5], who strengthened simulatable CKEM by adding covertness, i.e. that an interaction with either S or R (but not the keys they compute locally) is indistinguishable from a random beacon. Goyal and Jain

[10] strengthened the covert CKEM of [5] to proof-of-knowledge and simulation-soundness under (bounded) concurrent composition. Our covert CKEM notion is essentially the same as the covert ZKSend of [10] (minus the proof-of-knowledge property), but we adopt it to the CRS setting of a straight-line simulation using a global CRS trapdoor as in [3].

Covert CKEM Zero-Knowledge. We say that a CKEM for language L is *covert zero-knowledge* if the following properties hold:

1. *Setup Indistinguishability:* Parameters π generated by $\text{PG}(1^\tau)$ and $\text{TPG}(1^\tau)$ are computationally indistinguishable.
2. *Zero Knowledge:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\text{Rec\&Out}(\pi, x, \ell, w)}(\text{st})\} \approx \{\mathcal{A}_2^{\text{TRec\&Out}(\pi, x, \ell, \text{td})}(\text{st})\}$$

for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ and $(\text{st}, x, w, \ell) \leftarrow \mathcal{A}_1(\pi, \text{td})$ s.t. $(x, w) \in \mathcal{R}[L]$.²

3. *Trapdoor-Receiver Covertness:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\text{TRec}(\pi, x, \ell, \text{td})}(\text{st})\} \approx \{\mathcal{A}_2^{\text{TRec}^{\mathcal{S}(\tau)}}(\text{st})\}$$

for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ and $(\text{st}, x, \ell) \leftarrow \mathcal{A}_1(\pi, \text{td})$.

4. *Sender Simulation-Covertness:* For every efficient $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have

$$\{\mathcal{A}_2^{\text{Snd}(\pi, x, \ell), \text{TRec}_{\text{Block}(x, \ell)}(\text{td}, \cdot)}(\text{st})\} \approx \{\mathcal{A}_2^{\text{Snd}^{\mathcal{S}(\tau)}, \text{TRec}_{\text{Block}(x, \ell)}(\text{td}, \cdot)}(\text{st})\}$$

for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ and $(\text{st}, x, \ell) \leftarrow \mathcal{A}_1^{\text{TRec}(\text{td}, \cdot)}(\pi)$ s.t. $\text{TRec}(\text{td}, \cdot)$ was not queried on (x, ℓ) .

Note that *Zero-Knowledge* and *Trapdoor-Receiver Covertness* imply a *Receiver Covertness* property, which asks that $\text{Rec}(\pi, x, \ell, w)$ instances are indistinguishable from $\text{Rec}^{\mathcal{S}(\tau)}$ for any $(x, w) \in \mathcal{R}[L]$. This holds because an interaction with $\text{Rec}(\pi, x, \ell, w)$ for $(x, w) \in \mathcal{R}[L]$ is, by *Zero-Knowledge*, indistinguishable from an interaction with $\text{TRec}(\pi, x, \ell, \text{td})$, which by *Trapdoor-Receiver Covertness* is indistinguishable from an interaction with $\text{TRec}^{\mathcal{S}(\tau)}$, which is in turn identical to an interaction with $\text{Rec}^{\mathcal{S}(\tau)}$, because *Zero-Knowledge* implies that Rec and TRec output equal-sized messages.

Discussion. CKEM zero-knowledge [3] says that an interaction with Rec on any $x \in L$ followed by Rec 's local output K_R , can be simulated by TRec without knowledge of the witness for x . Receiver and Trapdoor-Receiver *covertness* mean that, in addition, the adversary \mathcal{A} who interacts with resp. Rec and TRec , but does not see their local outputs, cannot tell them from random beacons. In the case of TRec we ask for this to hold for any x and not only for $x \in L$ because a simulator of a higher-level protocol will typically create incorrect statements

² If \mathcal{A}_1 outputs $(x, w) \notin \mathcal{R}[L]$ we override \mathcal{A}_2 's output by an arbitrary constant.

and then it will simulate the Receiver algorithm on them. Note that we cannot include the output K_R of either Rec or TRec in \mathcal{A} 's view in the (trapdoor) receiver covertness game because \mathcal{A} can compute it by running $\text{Snd}(x)$. Sender covertness means that an interaction with the Snd is indistinguishable from an interaction with a random beacon for any x . Here too we cannot include Snd 's local output K_S in \mathcal{A} 's view because if $(x, w) \in \mathcal{R}[\mathbb{L}]$ then \mathcal{A} who holds w can compute it running $\text{Rec}(x, w)$. Note that \mathcal{A} 's view in the zero-knowledge and trapdoor-receiver covertness properties includes the simulator's trapdoor td , which implies that both properties will hold in the presence of multiple CKEM instances simulated by TRec using td . This is not the case for in sender simulation-covertness, but there the adversary has *oracle access* to simulator TRec who uses td on other CKEM instances, which suffices for the same goal of preserving the CKEM covertness property under concurrent composition.

Covert Soundness and Simulation-Soundness. A CKEM is covert sound if interaction with Snd on $x \notin \mathbb{L}$ followed by Snd 's local output K_S is indistinguishable from interaction with a random beacon. Recall that CKEM soundness [3] requires pseudorandomness of only Snd 's output K_S on $x \notin \mathbb{L}$, while here we require it also of the transcript produced by Snd . Covert simulation-soundness requires that this holds even if the adversary has access to the Trapdoor-Receiver for any (x', ℓ') which differs from the pair (x, ℓ) that defines the soundness challenge. To that end we use notation $P_{\text{Block}(x)}$ for a wrapper over (interactive) algorithm P which outputs \perp on input $x' = x$ and runs $P(x')$ for $x' \neq x$: CKEM is *Covert Sound* if for every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\text{Snd}_{\&\text{Out}}(\pi, x, \ell)}(\text{st})\} \approx \{\mathcal{A}_2^{\text{Snd}_{\&\text{Out}}^{\text{S}(\tau)}}(\text{st})\}$$

for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ and $(\text{st}, x, \ell) \leftarrow \mathcal{A}_1(\pi)$ s.t. $x \notin \mathbb{L}$.

CKEM is *Covert Simulation-Sound* if for every efficient algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\{\mathcal{A}_2^{\text{Snd}_{\&\text{Out}}(\pi, x, \ell), \text{TRec}_{\text{Block}(x, \ell)}(\text{td}, \cdot)}(\text{st})\} \approx \{\mathcal{A}_2^{\text{Snd}_{\&\text{Out}}^{\text{S}(\tau)}, \text{TRec}_{\text{Block}(x, \ell)}(\text{td}, \cdot)}(\text{st})\}$$

for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ and $(\text{st}, x, \ell) \leftarrow \mathcal{A}_1^{\text{TRec}(\text{td}, \cdot)}(\pi)$ s.t. $x \notin \mathbb{L}$ and $\text{TRec}(\text{td}, \cdot)$ was not queried on (x, ℓ) .

Note that sender simulation-covertness together with standard, i.e. non-covert, simulation-soundness, imply covert simulation-soundness of a CKEM:

Lemma 1. *If a CKEM scheme is simulation-sound [3] and sender simulation-covert, then it is also covert simulation-sound.*

Proof. Consider the simulation-soundness game where adversary \mathcal{A} on input π for $(\pi, \text{td}) \leftarrow \text{TPG}(1^\tau)$ interacts with $\text{TRec}(\text{td}, \cdot)$, generates (x, ℓ) s.t. $x \notin \mathbb{L}$, and interacts with oracles $\text{Snd}_{\&\text{Out}}(\pi, x, \ell)$ and $\text{TRec}_{\text{Block}(x, \ell)}(\text{td}, \cdot)$. The standard (i.e. non-covert) simulation soundness of this CKEM [3] implies that this game is indistinguishable from a modification in which key K_S output by $\text{Snd}_{\&\text{Out}}(\pi, x, \ell)$

is chosen at random. Once K_S is independently random, sender simulation-coverttness, which holds for all x , implies that this game is indistinguishable from a modification where the *messages* sent by Snd are replaced by uniformly random strings. Since these two moves together replace oracle $\text{Snd}_{\&\text{Out}}(\pi, x, \ell)$ with $\text{Snd}_{\&\text{Out}}^{\mathcal{S}(\tau)}$, it follows that the CKEM is covert simulation-sound.

6 Covert CKEM’s for LMI and Σ -Protocol Languages

Linear Map Image (LMI) Languages. The Covert 2PC protocol of Sect. 7 relies on covert zero-knowledge and simulation-sound (covert-zk-and-ss) CKEM’s for what we call *Linear Map Image* languages. A linear map image language $\text{LMI}_{n,m}$ for group G of prime order p contains pairs $(C, M) \in G^n \times G^{n \times m}$ s.t. there exists a vector $w \in \mathbb{Z}_p^m$ s.t. $C = w \cdot M$, where the vector dot product denotes component-wise exponentiation, i.e. $[w_1, \dots, w_m] \cdot [g_{i1}, \dots, g_{im}] = \prod_{j=1}^m (g_{ij})^{w_j}$. In other words, $(C, M) \in \text{LMI}_{n,m}$ if C is in the image of a linear map $f_M : \mathbb{Z}_p^m \rightarrow G^n$ defined as $f_M(w) = w \cdot M$. Using an additive notation for operations in group G we can equivalently say that $(C, M) \in \text{LMI}_{n,m}$ if C is in the subspace of G^n spanned by the rows of M , which we denote $\text{span}(M)$.

We extend the notion of a Linear Map Image language to a *class* of languages, denoted LMI, which includes all languages L for which there exist two efficiently computable functions $\phi : U_x \rightarrow (G \times G^{n \times m})$ and $\gamma : U_w \rightarrow \mathbb{Z}_p^m$ for some n, m , where U_x, U_w are the implicit universes of respectively statements in L and their witnesses, s.t. for all $(x, w) \in U_x \times U_w$, w is a witness for $x \in L$ if and only if $\gamma(w)$ is a witness for $\phi(x) \in \text{LMI}_{n,m}$. We will sometimes abuse notation by treating set $\{\phi(x)\}_{x \in L}$, i.e. L mapped onto (some subset of) $\text{LMI}_{n,m}$, replaceably with L itself. Observe that LMI is closed under conjunction, i.e.

$$[(C_1, M_1) \in \text{LMI}_{n_1, m_1} \wedge (C_2, M_2) \in \text{LMI}_{n_2, m_2}] \Leftrightarrow (C, M) \in \text{LMI}_{n_1+n_2, m_1+m_2}$$

for $C = (C_1, C_2)$ and M formed by placing M_1 in the upper-left corner, M_2 in the lower-right corner, and all-one matrices in the remaining quadrants.

Covert CKEM’s for LMI Languages. We show three types of covert CKEM’s for LMI languages: (1) a 2-round CKEM in CRS for LMI languages defined by a full row rank matrix M , whose cost is about 2–4 times that of the underlying HVZK, (2) a 2-round CKEM in ROM for Σ -protocol languages whose cost matches the underlying HVZK, and (3) a 4-round CKEM in CRS for Σ -protocol languages with a small additive overhead over the underlying HVZK. Note that every LMI languages in prime-order groups has a Σ -protocol, so constructions (2) and (3) apply to LMI languages. For lack of space we include below only the last construction and we refer to the full version [14] for the other two.

6.1 2-Round Covert CKEM for Σ -Protocol Languages in ROM

The covert mutual authentication scheme of Jarecki [13] uses a compiler which converts a Σ -protocol into a 2-round CKEM for the same language, assuming

ROM. The resulting CKEM was shown to satisfy the CKEM covertness notion of [13], which included receiver covertness and *strong* sender covertness, a covert counterpart of strong soundness (a.k.a. proof of knowledge), but did not include simulatability or simulation soundness. However, it is not hard to see that this 2-round CKEM does achieve both zero-knowledge and simulation soundness.

We recall this construction in Fig. 2, and we briefly explain how it works. Assume that the instance, witness pairs of language L are mapped into instances $x = (C, M) \in G^n \times G^{n \times m}$ and witnesses $w \in \mathbb{Z}_p^m$ of $LMI_{n,m}$. Recall a Σ -protocol for $LMI_{n,m}$: The prover picks random $w' \leftarrow \mathbb{Z}_p^m$, sends $a = w' \cdot M$ to the verifier, and on verifier's challenge e chosen uniformly in \mathbb{Z}_p , it outputs $z = w' + ew$ (multiplication by a scalar a vector addition in \mathbb{Z}_p^m). The verifier accepts if $a = z \cdot M - eC$, which holds if $C = w \cdot M$. As is well known, this Σ -protocol becomes a NIZK in ROM if the verifier's e is computed via a random oracle.

Consider an ElGmal-based covert commitment: Let g_1, g_2 be two random group elements in the CRS. Let $Com_{g_1, g_2}(m)$ for $m \in \mathbb{Z}_p$ pick $r \leftarrow \mathbb{Z}_p$ and output $cmt = (cmt_1, cmt_2) = ((g_1)^r, (g_2)^r (g_1)^m)$. This is perfectly binding and covert under the DDH assumption. Define language $Lc = \{(cmt, m) \mid cmt = Com_{g_1, g_2}(m)\}$, and note that Lc has a well-known covert SPHF: $KG(g_1, g_2)$ generates $hk \leftarrow \mathbb{Z}_p^2$ and $hp = hk \cdot (g_1, g_2) = (g_1)^{hk_1} (g_2)^{hk_2}$, $Hash((cmt, m), hk) = hk \cdot (cmt / (1, g_1^m)) = (cmt_1)^{hk_1} (cmt_2 / (g_1)^m)^{hk_2}$, and $PHash((cmt, m), r, hp) = r \cdot hp = (hp)^r$.

Let H be a hash onto \mathbb{Z}_p . The 2-round ROM-based covert CKEM of [13] works just like a ROM-based NIZK except that the prover replaces message a with its commitment $cmt = Com(H(a))$, and the non-interactive verification check whether $a = z \cdot M - eC$, the verifier computes $a = z \cdot M - eC$ locally and uses the covert SPHF for Lc to verify if cmt is a commitment to $H(a)$. This protocol is shown in Fig. 2, where $H_i(x)$ stands for $H(i, x)$.

On inputs $(g_1, g_2), (C, M) = \phi(x), \ell$, and on R 's input w s.t. $C = w \cdot M$:

R: Pick $w' \leftarrow \mathbb{Z}_p^m$ and $r \leftarrow \mathbb{Z}_p$, set $a = w' \cdot M$, $cmt \leftarrow Com_{g_1, g_2}(H_2(a); r)$, $e = H_1(x, \ell, cmt)$, $z = w' + ew$, and send (cmt, z) to S .

S: Set $a = z \cdot M - eC$ for $e = H_1(x, \ell, cmt)$, generate $(hk, hp) \leftarrow KG(g_1, g_2)$, send hp to R and output $K_S = Hash((cmt, m), hk)$ for $m = H_2(a)$.

R: Output $K_R = PHash((cmt, m), r, hp)$ for $m = H_2(a)$.

Fig. 2. 2-round covert-zk-and-ss CKEM in ROM for LMI (adopted from [13])

Figure 2 is written specifically for LMI languages but it is easy to see that the same works for any Σ -protocol language. Note that its cost is that of the Σ -protocol for language L plus 2 exponentiations for S and 1 exponentiation for R . We refer to the full version of the paper [14] for the proof of Theorem 1:

Theorem 1. *For any LMI language L , the CKEM scheme for L shown in Fig. 2 is covert zero-knowledge and covert simulation-sound in ROM, assuming DDH.*

7 Covert Computation of General 2-Party Functions

We describe protocol Π_{COMP} , Fig. 3, which realizes the concurrent 2-party covert computation functionality $F_{C(f,g)}$ in the CRS model. Protocol Π_{COMP} is a *covert* variant of the cut-and-choose method over $O(\tau)$ copies of Yao’s garbled circuit [29], which has been the common paradigm for standard 2PC protocols, initiated by [18, 21], followed by many subsequent works, e.g. [12, 19, 26], including several implementation efforts, e.g. [1, 16, 24, 26, 27].

A standard way of implementing a cut-and-choose involves tools which are inherently non-covert: First, the garbling party B sends commitments to n copies of the garbled circuit and then decommits a randomly chosen half of them, so that party A can verify that the opened circuits are formed correctly *and* that they were committed in B ’s first message. Clearly, if B sends a commitment followed by a decommitment, this can be verified publicly, at which point A would distinguish a protocol-participating party B from a random beacon regardless of the inputs which A or B enter into the computation. Secondly, a cut-and-choose protocol can also use secondary zero-knowledge proofs, e.g. to prove that the OT’s are performed correctly, or that the keys opened for different circuit copies correspond to the same inputs, and zero-knowledge proofs are similarly inherently non-covert.

Here we show that (concurrent and simulation-sound) covert CKEM’s can be effectively used in both of the above cases:

First, we use CKEM’s in place of all zero-knowledge proofs, i.e. instead of party P_1 proving statement x to party P_2 , we will have P_2 encrypt its future messages under a key derived by CKEM on statement x . By covert concurrent zero-knowledge, the simulator can derive the CKEM keys and simulate subsequent interaction of each protocol instance even if the statements it makes on behalf of honest players are incorrect (e.g. because the simulator does not know these players’ real inputs). By covert simulation-soundness, the CKEM’s made by corrupted players are still sound, i.e. the CKEM keys created by the simulator on behalf of honest parties are indistinguishable from random unless the statement made by a corrupted player is correct. Moreover, CKEM messages sent by either party are indistinguishable from random strings.

Secondly, we replace a commit/decommit sequence with a covert commitment c , release of the committed plaintext m (which must be pseudorandom), and a covert CKEM performed on a statement that there exists decommitment d (the CKEM receiver’s witness) s.t. d decommits c to m . We use a perfectly binding commitment so that the notion of language membership suffices to define this problem. Specifically, we implement the commitment scheme using covert Cramer-Shoup encryption, which plays two additional roles in the protocol construction: First, it assures non-malleability of each commitment/ciphertext. Secondly, it allows for straight-line extraction of player’s inputs using the decryption keys as a trapdoor for the CRS which contains a Cramer-Shoup encryption public key, which allows for security across concurrently executed protocol instances. Finally, the arithmetic structure of Cramer-Shoup encryption enables an efficient covert OT and efficient CKEM’s on statements on committed/encrypted values.

These are the basic guidelines we follow, but assuring (concurrent) simulatability of each party in a secure two-party computation, doing so efficiently, and doing so in the *covert* setting where the protocol view of each party must look like a random beacon except when the admission function evaluates to true and the functionality reveals computation outputs, requires several adjustments, which we attempt to explain in the technical protocol overview below.

Defining the Garbled Circuit. We first explain how we use the covert garbling procedure GCgen of [28], see Sect. 4, to enable covert computation of functionality $F_{C(f,g)}$ assuming the simplified case where the party that garbles the circuit is *Honest but Curious*. Our basic design follows the standard Yao's two-party computation protocol but instantiates it using covert building blocks, i.e. party B will use *covert garbling* on a circuit that corresponds to functionality $F_{C(f,g)}$ (more on this below), it will send the garbled circuit together with the input wire keys to A , either directly, for wires corresponding to B 's inputs, or via a *covert OT*, for wires corresponding to A 's inputs, and A will evaluate the garbled circuit to compute the output. This will work if the circuit garbled by B is appropriately chosen, as we explain here.

Step 1: Encoding B 's Output. Note that functionality $F_{C(f,g)}$ has two-sided output, so we must include an encoding of B 's output in the outputs of the garbled circuit in such a way that (1) this encoding looks random to A , and (2) A cannot modify this encoding to cause B to output any other value (except \perp). Let h be the two-sided output function at the heart of functionality $F_{C(f,g)}$, namely $h(x, y) = (z, v)$ s.t. $(z, v) = f(x, y)$ if $g(x, y) = 1$ and $(z, v) = (\perp, \perp)$ if $g(x, y) = 0$. Let n_x, n_y, n_z, n_v define resp. the length of input x of party A , input y of party B , output z of A , and output v of B . Let f_z, f_v satisfy $f(x, y) = (f_z(x, y), f_v(x, y))$. We will encode B 's output in the outputs of the garbled circuit evaluated by A using the standard way for converting the garbled circuit technique into secure computation of a two-sided function: If $\text{ts} = \{t_i^0, t_i^1\}_{i \in [n_v]}$ is the set of garbled circuit keys on the wires encoding B 's output v in the garbled circuit for h , then the garbled circuit evaluator A computes $(z, \text{ts}[v])$ where $(z, v) = f(x, y)$ (if $g(x, y) = 1$). Note that $\text{ts}[v]$ is an encoding of v which satisfies the above two conditions, and if A sends it to B , B can decode it to v using set ts . Even though this encoding of B 's output is implicit in the garbled circuit technique, we will add ts to the inputs and $\text{ts}[v]$ to the outputs of the function $f|_g$ we will garble, because this simplifies our notation and lets us use the covert garbling procedure GCgen of [28] as a black-box. In other words, we modify h to h' which on input $(x, (y, \text{ts}))$ outputs $(z, \text{ts}[v])$ for $(z, v) = f(x, y)$ if $g(x, y) = 1$ and (\perp, \perp) if $g(x, y) = 0$.

Step 2: Making \perp Output Random. Next, note that if B garbles the circuit for h' then for any x, y s.t. $g(x, y) = 0$, party A on input x will distinguish between a random beacon and an honest party B which executes the protocol on input y . (This would not be a covert computation of $F_{C(f,g)}$ because $F_{C(f,g)}$ assures that $A(x)$ cannot distinguish $B(y)$ for y s.t. $(y \neq \perp \wedge g(x, y) = 0)$, from a random beacon $B(\perp)$.) This is because in the 2nd case the garbled circuit

evaluates to $h'(x, y, \text{ts}) = (\perp, \perp)$, and in the 1st case A will interpret random strings as a garbled circuit and the input wire keys, and those will evaluate to random outputs. To make the circuit evaluate to random outputs in the case $g(x, y) = 0$, we add $(n_z + n_v\tau)$ -bit strings c and d to respectively A 's and B 's input, we define $h''((x, c), (y, d, \text{ts}))$ as $(z, \text{ts}[v])$ for $(z, v) = f(x, y)$ if $g(x, y) = 1$, and as $c \oplus d$ if $g(x, y) = 0$, and we specify that both A and B set input random c and d strings into the computation. Note that if B is honest then setting the output to d instead of $c \oplus d$ in the $g(x, y) = 0$ case would suffice, but a malicious B would be then able to set A 's output in the $g(x, y) = 0$ case, because A treats the first n_z bits of the circuit output as its local output z .

Step 3: Adding Simulation Trapdoor. Finally, we add a “simulator escape” input bit u to B 's inputs, and the final circuit we garble, function $f|_g$ defined below, is like h'' but with condition $(g(x, y) \wedge u)$, in place of condition $g(x, y)$, for deciding between output $(z, \text{ts}[v])$ for $(z, v) = f(x, y)$ and output $c \oplus d$:

$$f|_g((x, c), (y, d, \text{ts}, u)) = \begin{cases} (f_z(x, y), \text{ts}[v]) & \text{if } g(x, y) = 1 \wedge u = 1 \\ & \text{where } v = f_v(x, y) \text{ and } \text{ts}[v] = [t_1^{v[1]}, \dots, t_{n_v}^{v[n_v]}] \\ c \oplus d & \text{otherwise,} \end{cases}$$

Here is how we will use this “escape bit” in the $g(x, y) = 1$ clause in the simulation: An honest real-world party B will set $u = 1$, in which case circuit $f|_g$ is identical to h'' . However, a simulator \mathcal{A}^* for the case of corrupt party A , will use the $u = 0$ escape clause to aid in its simulation as follows: \mathcal{A}^* will send to A a garbled circuit for $f|_g$ as B would, but before it sends the wire input keys corresponding to its inputs, it needs to *extract* inputs (x, c) which A contributes to the covert OT. (This is why we base the covert OT of Sect. 4 on CCA(-covert) PKE of Cramer-Shoup: The receiver’s first message will be a vector of Cramer-Shoup encryptions of the bits in string $x|c$, which the simulator will straight-line extract using the decryption key as a trapdoor.) Having extracted (x, c) from the covert OT, the simulator \mathcal{A}^* , playing the role of an ideal-world adversary $F_{C(f,g)}$ ’s instance identified by sid , sends x to $F_{C(f,g)}$ and receives $F_{C(f,g)}$ ’s reply z . Note that if \mathcal{A}^* sets $u = 0$ then the only part of its input that matters is d , because $f|_g$ will outputs $c \oplus d$ to A . Simulator \mathcal{A}^* will then prepare d as follows: If $z \neq \perp$, i.e. the input y to the ideal-world party B must be such that $g(x, y) = 1$, simulator \mathcal{A}^* picks t' as a random $n_v\tau$ string and sets $d = c \oplus (z|t')$. In this way the garbled circuit will output $c \oplus d = z|t'$. Since t' is a sequence of n_v random bitstrings of length τ , string $z|t'$ is distributed identically to the circuit output $z|\text{ts}[v]$ which A would see in an interaction with the real-world party $B(y)$. Moreover, \mathcal{A}^* can detect if A tries to cheat the real-world party B by sending a modified encoding of B 's output: If A sends back the same t' which \mathcal{A}^* embedded in the circuit output, then \mathcal{A}^* sends $(\text{Output}, \text{sid}, B, T)$ to $F_{C(f,g)}$, and if A sends any other value, in which case the real-world B would reject, \mathcal{A}^* sends $(\text{Output}, \text{sid}, B, F)$ to $F_{C(f,g)}$.

Notation for Garbled Circuit Wires. We will find it useful to fix a notation for groups of wires in the garbled circuit $f|_g$ depending on the part of the input they encode. Note that $f|_g$ takes input $((x, c), (y, d, \text{ts}, u))$. We will use W to denote all the input wires, and we will use X, C, Y, D, T, U to denote the sets of wires encoding the bits of respectively x, c, y, d, ts, u , where $|X| = n_x, |Y| = n_y, |T| = 2n_v\tau, |C| = |D| = n_z + n_v\tau, |U| = 1$. We denote the set of wires for A 's inputs as $\bar{X} = X \cup C$ and the set of wires for B 's inputs as $\bar{Y} = Y \cup D \cup T \cup U$. If bitstring s is formed as concatenation of any of the circuit inputs x, c, y, d, t, u and $w \in W$ then $s[w]$ denotes the bit of s corresponding to input wire w .

Fully Malicious Case. In a simple usage of the cut-and-choose technique for garbled circuits, party B would use $\text{GCgen}(f|_g)$ to prepare $n = O(\tau)$ garbled circuit instances $(\text{gc}_1, \text{ks}_1), \dots, (\text{gc}_n, \text{ks}_n)$, would send $(\text{gc}_1, \dots, \text{gc}_n)$ to A , who would choose a random subset $S \in [n]$ of $n/2$ elements, send it to B , who would then open the coins it used in preparing gc_i 's for $i \in S$, and proceed with the OT's and sending its input wire keys for all gc_i 's for $i \notin S$. Party A would then check that each gc_i for $i \in S$ is formed correctly, and it would evaluate each gc_i for $i \notin S$. If at least $n/4$ of these returned the same value w , A would interpret this as the correct output $w = (z, \text{ts}[: v])$ of $f|_g$, output z locally and send $\text{ts}[: v]$ to B , who would decode it to its output v . In order to enforce consistency of the inputs which both parties provide to circuit instances $\{\text{gc}_i\}_{i \notin S}$, we would have each party to commit to their inputs to $f|_g$, and then use efficient ZK proofs that the keys B sends and the bits A chooses in the OT's for the evaluated gc_i instances correspond to these committed inputs. Further, B would need to commit to each key in the wire key sets $\{\text{ks}_i\}_{i \in [n]}$, and show in a ZK proof that the keys it sends and enters into the OT's for $i \notin S$ are the committed keys. Our protocol uses each of the elements of this sketch, but with several modifications.

Step 1: ZK \rightarrow CKEM, Com/Decom \rightarrow CKEM. First, we follow the above method using covert commitments, covert circuit garbling, and covert OT. Second, we replace all the ZK proofs with covert simulation-sound CKEM's. Next, note that circuits $\{\text{gc}_i\}_{i \in [n]}$ in themselves are indistinguishable from random by covertness of the garbling procedure, but if gc_i 's were sent in the clear then B could not then open the coins used in the preparation of gc_i 's for $i \in S$, because coin r_i^{gc} together with gc_i s.t. $(\text{gc}_i, \text{ks}_i) = \text{GCgen}(f|_g; r_i^{\text{gc}})$ forms a publicly verifiable (commitment, decommitment) pair. We deal with it roughly the way we deal with general (commitment, decommitment) sequence. In this specific case, we replace gc_i 's in B 's first message with covert commitments to both the circuits, $\text{cgc}_i \leftarrow \text{Com}_{\text{pk}}(\text{gc}_i; r_i^{\text{cgc}})$, and to all the input wire keys, $\text{ck}_i^{w,b} \leftarrow \text{E}_{\text{pk}}(k_i^{w,b}; r_{i,w,b}^{\text{ck}})$. When B sends r_i^{cgc} for $i \in S$, A can derive $(\text{gc}_i, \{k_i^{w,b}\}_{w,b}) \leftarrow \text{GCgen}(f|_g; r_i^{\text{cgc}})$, and now A has a (commitment, message) pair $(c, m) = (\text{cgc}_i, \text{gc}_i)$ and (encryption, message) pairs $(c, m) = (\text{ck}_i^{w,b}, k_i^{w,b})$, while B has the randomness r s.t. $c = \text{Com}_{\text{pk}}(m; r)$ or $c = \text{E}_{\text{pk}}(m; r)$. Since we implement $\text{Com}(m)$ as $\text{E}(H(m))$, both instances can be dealt with a covert CKEM, with sender A and receiver B , for the statement that (c, m) is in the

language of correct (ciphertext, plaintext) pairs for Cramer-Shoup encryption, i.e. $\text{Le}^\ell(\text{pk})$. Finally, to covertly encode the random $n/2$ -element subset S chosen by A , we have A send to B not the set S but the coins r^{SG} which A uses in the subset-generation procedure SG which generates a random $n/2$ -element subset on n -element set.

Let us list the CKEM's which the above procedure includes so far. A has to prove that it inputs into the OT's for all $i \notin S$ the same bits which A (covertly) committed in its first message. Recall that in the covert OT based on the Cramer-Shoup encryption (see Sect. 4) the receiver's first message is the encryption of its bit. We will have A then commit to its bits by encrypting them, and so the proof we need is that the corresponding plaintexts are bits, and for that purpose we will use a CKEM for language $\text{Lbit}^\ell(\text{pk})$ (see language LA below). Party B has more to prove: First, it needs to prove all the $\text{Le}^\ell(\text{pk})$ statements as explained above (these correspond to items #1, #2, and #3 in the specification of LB below). Second, B proves that its committed inputs on wires $\overline{Y} \setminus D$ are bits, using CKEM for $\text{Lbit}^\ell(\text{pk})$, and that for $i \notin S$ it reveals keys consistent with these committed inputs. Both statements will be handled by CKEM for language Ldis , see below, which subsumes language $\text{Lbit}^\ell(\text{pk})$ (see item #4 in the specification of LB). Third, for reasons we explain below, B will not prove the consistency of inputs d it enters into n instances of garbled circuit $f|_g$, hence for $i \notin S$ and $w \in D$ it needs only to prove that the revealed key $k_i^{w,b}$ is committed either in $\text{ck}_i^{w,0}$ or $\text{ck}_i^{w,1}$, which is done via CKEM for Ldis' (see below, and item #5 in the specification of LB). Finally, B proves that it computes the OT responses otr_i^w correctly, for $i \notin S$ and $w \in \overline{X}$, on A 's first OT message ct_i^w using the keys committed in $\text{ck}_i^{w,0}$, $\text{ck}_i^{w,1}$, which is done via CKEM for Lotr (see below, and item #6 in the specification of LB).

Step 2: Input Consistency Across Garbled Circuit Copies. We must ensure that A and B input the same x and y into each instance of the garbled circuit $f|_g$. However, the decision process in our cut-and-choose approach is that A decides whether the outputs w_i of $n/2$ garbled circuits $i \notin S$ it evaluates correspond to $(z, \text{ts}[v])$ for $(z, v) = f(x, y)$ or to random bits, is that it decides on the former if $n/4$ of the w_i 's are the same. Hence, to prevent B from getting honest A into that case if $g(x, y) = 0$ (or $u = 0$), A chooses each c_i at random, so in that case the (correctly formed) circuits in $[n] \setminus S$ output $w_i = c_i \oplus d_i$ values which B cannot control. Consequently, B must also choose each d_i independently at random, which is why B does not have to commit to the inputs on wires in D .

Step 3: Straight-Line Extraction of Inputs. As we sketched before, we get concurrent security by using CCA-covert encryption as, effectively, a non-malleable and straight-line extractable covert commitment. Each player commits to its input bits by encrypting them (except B does not encrypt its inputs on D wires), and the simulator decrypts the bits effectively contributed by a malicious party. However, for the sake of efficient CKEM's we need these bit encryptions to use a "shifted" form, i.e. an encryption of bit b will be $E_{\text{pk}}(g^b)$

and not $E_{pk}(b)$. This is because the Cramer-Shoup encryption E has group G as a message space. Also, if bit b is encrypted in this way then we can cast the language L_{bit} (and languages L_{dis} and L_{dis}') as an LMI language with an efficient covert CKEM.

Step 4: Encoding and Encrypting Wire Keys. To enable efficient covert CKEM's for the languages we need we also modify the garbling procedure $GCgen$ of [28] so it chooses wire keys $k^{w,b}$ corresponding to A 's input wires, i.e. for $w \in \bar{X}$, as random elements in G , but keys $k^{w,b}$ corresponding to B 's input wires, i.e. for $w \in \bar{Y}$, as random elements in \mathbb{Z}_p . Note that either value can be used to derive a standard symmetric key, e.g. using a strong extractor with a seed in the CRS. We use the same encryption E to commit to these wire keys, but we commit them differently depending on whose wires they correspond to, namely as $ck^{w,b} = E_{pk}(k^{w,b})$ for $w \in \bar{X}$, because $k^{w,b} \in G$ for $w \in \bar{X}$, and as $ck^{w,b} = E_{pk}(g^{k^{w,b}})$ for $w \in \bar{Y}$, because $k^{w,b} \in \mathbb{Z}_p$ for $w \in \bar{Y}$. The reason we do this is that values $ck^{w,b}$ for $w \in \bar{X}$ take part in the CKEM for correct OT response language $Lotr$, and since in OT the encrypted messages (which are the two wires keys $k^{w,0}$ and $k^{w,1}$) will be in the base group, hence we need the same keys to be in the base group in commitments $ck^{w,0}, ck^{w,1}$. By contrast, values $ck^{w,b}$ for $w \in \bar{Y}$ take part in the CKEM of language L_{dis} , for proving consistency of key k^w opened by B with B 's commitment ct^w to bit b on wire w . Bit b is in the exponent in ct^w , and using homomorphism of exponentiation, this allows us to cast language L_{dis} as an LMI language provided that k^w is also in the exponent in $ck^{w,0}$ and $ck^{w,1}$.

Step 5: Using CKEM Keys to Encrypt and/or Authenticate. We will run two CKEM's: After A 's first message, containing A 's input commitments, we run a covert CKEM for language LA for correctness of A 's messages, with sender B and receiver A , denoting the keys this CKEM establishes as K_B for B and K'_B for A . Subsequently, B will encrypt its messages under key K_B , using covert encryption (SE, SD) implemented as $SE_K^0(m) = G^{|\mathbf{m}|}(F(K, 0)) \oplus m$ and $SD_K^0(ct) = G^{|\mathbf{ct}|}(F(K, 0)) \oplus ct$, where F is a PRF with τ -bit keys, arguments, and outputs, G^ℓ is a PRG with τ -bit inputs and ℓ -bit outputs. Similarly when B responds as described above given A 's chosen set $S \subset [n]$, we run a covert CKEM for language LB for correctness of B 's messages, with sender A and receiver B , establishing keys K_A for A and K'_A for B , and A will encrypt its subsequent messages using the same covert encryption. In the last two messages we will use values $F(K_B, 1)$, $F(K_B, 2)$, and $F(K_A, 1)$ derived from the same CKEM keys as, resp. a one-time authenticator for A 's message m_A^2 , an encryption key for B 's final message m_B^3 , and a one-time authenticator for that same message.

Covert CKEM's for Linear Map Image Languages. Protocol Π_{COMP} uses CKEM's for two languages: Language LA which contains correctly formed wire-bit ciphertexts sent by A , and language LB which contains correctly formed messages sent by B . Both are formed as conjunctions of LMI languages, hence

both are LMI languages as well. Let (\mathbf{Kg}, E, D) be the CCA-covert Cramer-Shoup PKE. All languages below are implicitly parametrized by the public key \mathbf{pk} output by $\mathbf{Kg}(1^\tau)$ and some label ℓ . (Formally key \mathbf{pk} and label ℓ are a part of each statement in the given language.) Recall that the public key \mathbf{pk} specifies the prime-order group setting (g, G, p) .

We first list all the component languages we need to define LA and LB. We defer to full version [14] for the specification of the mapping between the instances of each language to instance (C, M) of $\text{LMI}_{n,m}$ for some n, m .

Language Le of correct (ciphertext,label,plaintext) tuples for plaintext $\mathbf{m} \in G$:

$$\text{Le}^\ell(\mathbf{pk}) = \{(\text{ct}, \mathbf{m}) \text{ s.t. } \text{ct} \in E_{\mathbf{pk}}^\ell(\mathbf{m})\}$$

Language Lbit of “shifted” encryptions of a bit:

$$\text{Lbit}^\ell(\mathbf{pk}) = \{\text{ct s.t. } \exists b \in \{0, 1\} (\text{ct}, g^b) \in \text{Le}^\ell(\mathbf{pk})\}$$

Language Ldis is used for proving that a key corresponding to some *sender’s wire* in Yao’s garbled circuit is consistent with the two key values the sender committed in ck_0, ck_1 and with the bit the sender committed in ct . To cast this language as a (simple) LMI language we use the “shifted” version of Cramer-Shoup encryption in these statements, i.e. we encrypt $g^m \in G$ instead of $\mathbf{m} \in \mathbb{Z}_p$. In other words, Ldis consists of tuples $(\mathbf{m}, \text{ct}, \text{ck}_0, \text{ck}_1)$ s.t. either $(\text{ct}$ encrypts g^0 and ck_0 encrypts $g^{\mathbf{m}}$) or $(\text{ct}$ encrypts g^1 and ck_1 encrypts $g^{\mathbf{m}}$):

$$\text{Ldis}^{\ell,i}(\mathbf{pk}) = \{(\text{ct}, \mathbf{m}, \text{ck}_0, \text{ck}_1) \text{ s.t. } \exists b \in \{0, 1\} (\text{ct}, g^b) \in \text{Le}^\ell(\mathbf{pk}) \wedge (\text{ck}_b, g^{\mathbf{m}}) \in \text{Le}^{[\ell|ib]}(\mathbf{pk})\}$$

Language Ldis' is a simplification of Ldis which omits checking the constraint imposed by ciphertext ct .

Language Lotr is used for proving correctness of a response in an Oblivious Transfer of Aiello et al. [2], formed using procedure OTrsp (see Sect. 4), which the sender uses in Yao’s protocol to send keys corresponding to *receiver’s wires*:

$$\begin{aligned} \text{Lotr}^\ell(\mathbf{pk}) = & \{(\text{otr}, \text{ct}, \text{ck}_0, \text{ck}_1) \text{ s.t. } \exists k_0, k_1, r \\ & (\text{ck}_0, k_0) \in \text{Le}^{[\ell|0]}(\mathbf{pk}) \wedge (\text{ck}_1, k_1) \in \text{Le}^{[\ell|1]}(\mathbf{pk}) \wedge \text{otr} = \text{OTrsp}_{\mathbf{pk}}(\text{ct}, k_0, k_1; r)\} \end{aligned}$$

We use the above component languages to define languages LA and LB as follows:

$$\begin{aligned} \text{LA}^{\ell_A}(\mathbf{pk}) = & \{(\{\text{ct}^w\}_{w \in X}, \{\text{ct}_i^w\}_{i \in [n], w \in C}) \\ \text{s.t. } & \text{ct}^w \in \text{Lbit}^{[\ell_A|w]}(\mathbf{pk}) \quad \text{for all } w \in X \\ \text{and } & \text{ct}_i^w \in \text{Lbit}^{[\ell_A|w|i]}(\mathbf{pk}) \quad \text{for all } i \in [n], w \in X\} \end{aligned}$$

$$\begin{aligned} \text{LB}^{\ell_B}(\mathbf{pk}) = & \{(\{(cgc_i, H(gc_i))\}_{i \in [n]} \\ & \{(k_i^{w,b}, \text{ck}_i^{w,b})\}_{i \in S, w \in \bar{X}, b \in \{0,1\}} \\ & \{(g^{k_i^{w,b}}, \text{ck}_i^{w,b})\}_{i \in S, w \in \bar{Y}, b \in \{0,1\}}\} \end{aligned}$$

$$\begin{aligned} & \{(k_i^w, \text{ct}^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1})\}_{i \notin S, w \in \overline{Y} \setminus D} \\ & \{(k_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1})\}_{i \notin S, w \in D} \\ & \{(\text{otr}_i^w, \text{ct}_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1})\}_{i \notin S, w \in \overline{X}} \end{aligned}$$

s.t.

- (1) $(\text{cgc}_i, H(\text{gc}_i)) \in \text{Le}^{[\ell_B|i]}(\text{pk})$ for $i \in [n]$
- (2) $(\text{ck}_i^{w,b}, k_i^{w,b}) \in \text{Le}^{[\ell_B|w|i|b]}(\text{pk})$ for $i \in S, w \in \overline{X}, b \in \{0, 1\}$
- (3) $(\text{ck}_i^{w,b}, g^{k_i^{w,b}}) \in \text{Le}^{[\ell_B|w|i|b]}(\text{pk})$ for $i \in S, w \in \overline{Y}, b \in \{0, 1\}$
- (4) $(\text{ct}^w, k_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1}) \in \text{Ldis}^{[\ell_B|w],i}(\text{pk})$ for $i \notin S, w \in \overline{Y} \setminus D$
- (5) $(k_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1}) \in \text{Ldis}^{[\ell_B|w],i}(\text{pk})$ for $i \notin S, w \in D$
- (6) $(\text{otr}_i^w, \text{ct}_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1}) \in \text{Lotr}^{[\ell_B|w]}(\text{pk})$ for $i \notin S, w \in \overline{X}$

Notation in Fig. 3. Procedures (Kg, E, D), (GCgen, GCev), Com, SG, (OTrsp, OTfin), CKEM, (F, G, SE, SD) are as explained above. If P is a randomized algorithm we sometimes explicitly denote its randomness as r^{P} , with the implicit assumption that it is a random string. Expression $\{x_i \leftarrow \mathsf{P}\}_{i \in R}$ denotes *either* a loop “perform $x_i \leftarrow \mathsf{P}$ for each i in R ”, *or* a set of values $\{x_i\}_{i \in R}$ resulting from executing such a loop. Letter b always stands for a bit, and expressions $\{\dots\}_b$ stand for $\{\dots\}_{b \in \{0,1\}}$.

Cost Discussions. Since the Covert CKEM’s of Sect. 6 have the same asymptotic computation and bandwidth costs as the HVZK proofs for the same languages, protocol Π_{COMP} realizes the concurrent *covert* 2PC functionality $\mathsf{F}_{\mathsf{C}(f,g)}$ with $O(1)$ rounds, $O(\tau|C|)$ bandwidth, $O(\tau|C|)$ symmetric cipher operations, and $O(\tau|W|)$ exponentiations, where $|C|$ is the number of gates and $|W|$ is the size of the input in the circuit for function $f|_g$, and τ is the security parameter. This places covert computation in the same efficiency ballpark as existing $O(1)$ -round secure (but *not* covert) “cut-and-choose over garbled circuits” 2PC protocols. Of course there remains plenty of room for further improvements: Protocol Π_{COMP} uses $2.4 \cdot \tau$ garbled circuit copies instead of τ as the 2PC protocols of [12, 17], it does not use an OT extension, and it does not use many other bandwidth and cost-saving techniques that were developed over the last decade to increase the efficiency of standard, i.e. non-covert, 2PC protocols. However, we see no inherent reasons why, using the techniques we employed here, many of the same cost-saving techniques cannot be adopted to covert computation.

Here we single out two particular sources of an “efficiency gap” between our covert 2PC and current secure 2PC protocols that perhaps stand out. First, protocol Π_{COMP} exchanges $O(\tau)$ garbled circuits instead of $O(\kappa)$ where κ is the statistical security parameter. We could do the latter as well, but the result would realize a weaker functionality than $\mathsf{F}_{\mathsf{C}(f,g)}$ defined in Sect. 3. Namely, with probability $2^{-\kappa}$ the functionality would allow the adversary to specify any function on the joint inputs, and this function would be computed by the honest party. Secondly, circuit $f|_g$ which is garbled in protocol Π_{COMP} increases the

$\text{PG}(1^\tau)$ picks $(\text{pk}, \text{sk}) \leftarrow \text{Kg}(1^\tau)$ and sets $\pi \leftarrow \text{pk}$.

B1: on input $(\text{InputB}, A, y, \text{sid})$ and $\ell_B = (B, A, \text{sid})$:

set $\{\{\text{gc}_i, \{k_i^{w,b}\}_{w \in W,b}\} \leftarrow \text{GCgen}(f|_g; r_i^{\text{gc}})\}_{i \in [n]}$ and $\{\text{cgc}_i \leftarrow \text{Com}_{\text{pk}}^{[\ell_B|i]}(\text{gc}_i)\}_{i \in [n]}$;
 set $\{\text{ck}_i^{w,b} \leftarrow \text{E}_{\text{pk}}^{[\ell_B|w|i|b]}(k_i^{w,b})\}_{w \in \bar{X}, i \in [n], b}$ and $\{\text{ck}_i^{w,b} \leftarrow \text{E}_{\text{pk}}^{[\ell_B|w|i|b]}(g^{k_i^{w,b}})\}_{w \in \bar{Y}, i \in [n], b}$;
 send $m_B^1 = (\{\text{cgc}_i\}_{i \in [n]}, \{\text{ck}_i^{w,b}\}_{i \in [n], w \in W, b})$ to A .

A1: on input $(\text{InputA}, B, x, \text{sid})$ and $\ell_A = (A, B, \text{sid})$, and message m_B^1 from B :

sample $S \leftarrow \text{SG}(n; r^{\text{SG}})$, pick $c_i \leftarrow \{0, 1\}^{n_z + n_v \tau}$ for $i \in [n]$;
 set $x_A \leftarrow (\{\text{ct}^w \leftarrow \text{E}_{\text{pk}}^{[\ell_A|w]}(g^{x[w]}; r_w^{\text{ct}})\}_{w \in X}, \{\text{ct}_i^w \leftarrow \text{E}_{\text{pk}}^{[\ell_A|w|i]}(g^{c_i[w]}; r_{w,i}^{\text{ct}})\}_{w \in C, i \in [n]})$;
 set $w_A \leftarrow (x, \{c_i\}_{i \in [n]}, \{r_w^{\text{ct}}\}_{w \in X}, \{r_{w,i}^{\text{ct}}\}_{w \in C, i \in [n]})$, send $m_A^1 = (r^{\text{SG}}, x_A)$ to B .

A, B run $\text{CKEM}_{\text{LA}(\ell_A)(\text{pk})}$ on x_A and A 's input w_A ; let B output K_B and A output K'_B .

B2: on K_B and r^{SG} received in m_A^1 from A :

re-generate $S \leftarrow \text{SG}(n; r^{\text{SG}})$, set $u = 1$, pick $t \leftarrow \{0, 1\}^{2n_v \tau}$ and $\{d_i \leftarrow \{0, 1\}^{n_z + n_v \tau}\}_{i \in [n]}$;
 set $\text{ct}^B \leftarrow \{\text{ct}^w \leftarrow \text{E}_{\text{pk}}^{[\ell_B|w]}(g^{\bar{y}[w]})\}_{w \in \bar{Y} \setminus D}$ where $\bar{y} = y|t|u$;
 set $\{k_i^w \leftarrow k_i^{w, \bar{y}_i[w]}\}_{w \in \bar{Y}, i \notin S}$ where $\bar{y}_i = y|t|u|d_i$, and $\{\text{ks}_i^B \leftarrow \{k_i^w\}_{w \in \bar{Y}}\}_{i \notin S}$;
 set $\{\text{otr}_i^w \leftarrow \text{OTrsp}_{\text{pk}}(\text{ct}_i^w, k_i^{w,0}, k_i^{w,1})\}_{w \in \bar{X}, i \notin S}$, where $\text{ct}_i^w = \text{ct}^w$ for $w \in X$;
 send $m_B^2 = \text{SE}_{K_B}^0[\text{ct}^B, \{r_i^{\text{gc}}\}_{i \in S}, \{\text{gc}_i, \text{ks}_i^B, \{\text{otr}_i^w\}_{w \in \bar{X}}\}_{i \notin S}]$ to A .

A2: on K'_B and m_B^2 :

set $(\text{ct}^B, \{r_i^{\text{gc}}\}_{i \in S}, \{\text{gc}_i, \text{ks}_i^B, \{\text{otr}_i^w\}_{w \in \bar{X}}\}_{i \notin S}) \leftarrow \text{SD}_{K'_B}^0(m_B^2)$;
 set $\{\text{ks}_i^A \leftarrow \{k_i^w \leftarrow \text{OTfin}_{\text{pk}}(\bar{x}_i[w], r_{w,i}^{\text{ct}}, \text{otr}_i^w)\}_{w \in \bar{X}}\}_{i \notin S}$, for $\bar{x}_i = x|c_i$ and $r_{w,i}^{\text{ct}} = r_w^{\text{ct}}$ for $w \in X$;
 set $(\text{gc}_i, \{k_i^{w,b}\}_{w,b}) \leftarrow \text{GCgen}(f|_g; r_i^{\text{gc}})$ for $i \in S$ and $w_i \leftarrow \text{GCev}(\text{gc}_i, (\text{ks}_i^A \cup \text{ks}_i^B))$ for $i \notin S$.

A, B run $\text{CKEM}_{\text{LB}(\ell_B)(\text{pk})}$ on x_B and B 's input w_B for $x_B = (\{\text{ct}^w, k_i^w, \text{ck}_i^{w,0}, \text{ck}_i^{w,1}\}_{i \notin S, w \in \bar{Y} \setminus D}, \{\text{ks}_i^A, \text{ck}_i^{w,0}, \text{ck}_i^{w,1}\}_{i \notin S, w \in D}, \{\text{gc}_i, \text{cgc}_i\}_{i \in [n]}, \{\text{ck}_i^{w,b}, k_i^{w,b}\}_{i \in S, w \in W, b}, \{\text{otr}_i^w, \text{ct}_i^w, \text{ck}_i^{w,b}\}_{i \notin S, w \in \bar{X}, b})$, and w_B containing input y and randomness of B . Let A output K_A and B output K'_A .

A3: If $\exists R \subset [n]$ s.t. $|R| = n/4$ and $\exists w$ s.t. $\forall i \in R$ $w_i = w$ then set $(z|t_1|\dots|t_{n_v}) := w$ and $m_A^2 \leftarrow \text{SE}_{K_A}^0(\text{F}(K'_B, 1), t_1, \dots, t_{n_v})$; Otherwise set $z := \perp$ and $m_A^2 \leftarrow \{0, 1\}^{\tau(n_v+1)}$. Send m_A^2 to B .

B3: Set $(\tau, t_1, \dots, t_{n_v}) \leftarrow \text{SD}_{K'_A}^0(m_A^2)$. Parse t as $[t_1^0|t_1^1|\dots|t_{n_v}^0|t_{n_v}^1]$. If $\tau = \text{F}(K_B, 1)$ and $t_j \in \{t_j^0, t_j^1\}$ for all $j \in [n_v]$ then set $m_B^3 \leftarrow \text{F}(K_B, 2) \oplus \text{F}(K'_A, 1)$ and \forall_j set $v[j] := b$ s.t. $t_j = t_j^b$; Otherwise set $v := \perp$ and $m_B^3 \leftarrow \{0, 1\}^\tau$. Send m_B^3 to A and output (Output, v) .

A4: If $m_B^3 \neq \text{F}(K'_B, 2) \oplus \text{F}(K_A, 1)$ then set $z := \perp$. Output (Output, z) .

Fig. 3. Protocol Π_{COMP} for concurrent 2-party covert function computation $\text{F}_{\text{C}(f,g)}$

number of input wires of the underlying circuit for $\text{F}_{\text{C}(f,g)}$ by $O(n_O \tau)$ where n_O is the bitsize of the *output* of function f . However, since this extension in the input wire count was done for conceptual simplicity (see a note on *Encoding B's Output* on page 19), we hope that it might be avoidable with a more careful

analysis. Moreover, since our covert 2PC is self-composable and both sides commit to their input bits in the same way, two instances of this protocol can be run in parallel for *one-sided output* versions of f , one for A 's output and one for B 's output, on same committed inputs. This multiplies all the other costs by 2 but drops the $O(n_{OT}\tau)$ growth in the circuit size.

Theorem 2. *Protocol Π_{COMP} in Fig. 3 realizes the concurrent 2-party covert computation functionality $F_{C(f,g)}$ in the CRS model, assuming $(\text{Kg}, \text{E}, \text{D})$ is a covert CCA public key encryption, F is a PRF, G is a PRG, $(\text{GCgen}, \text{GCev})$ is a covert garbling scheme, $(\text{OTreq}, \text{OTrsp}, \text{OTfin})$ is a covert OT, and $\text{CKEM}_{\text{LA}(\text{pk})}$ and $\text{CKEM}_{\text{LB}(\text{pk})}$ are covert zero-knowledge and simulation-sound CKEM's for languages resp. LA and LB.*

On behalf of honest B , on trapdoor sk and input $(\text{InputB}, \text{sid}, A, B)$ from $F_{C(f,g)}$:

- (1) compute $\{\text{cgc}_i, \{\text{ck}_i^{w,b}\}_{w,b}\}_{i \in [n]}$ and send to A as B does in step B1;
- (2) on $\mathbf{m}_A^1 = (r^{\text{SG}}, x_A)$ for $x_A = (\{\text{ct}^w\}_{w \in X}, \{\text{ct}_i^w\}_{i,w \in C})$ from A in A1, decrypt all ct 's using sk to obtain x, c_1, \dots, c_n , overwrite $x := \perp$ if any decryption returns \perp or not a bit, and send $(\text{InputA}, B, x, \text{sid})$ to $F_{C(f,g)}$ and receive $(\text{Output}, z, \text{sid})$ from $F_{C(f,g)}$; If $x = \perp$ then in steps (4,6) below pick $\mathbf{m}_B^2, \mathbf{m}_B^3$ at random and in step (5) run $\text{Rec}^{\mathbb{S}(\tau)}$.
- (3) run $\text{Snd}(\pi, (x_A, \ell_A))$ in $\text{CKEM}_{\text{LA}(\ell_A)(\text{pk})}$, let K_B be Snd 's output;
- (4) set $y := 0^{n_y}, u := 0, t \leftarrow \{0, 1\}^{2n_v\tau}$; if $z = \perp$ then $\forall i$ pick $d_i \leftarrow \{0, 1\}^{n_z + n_v\tau}$; o/w set $t' \leftarrow \{0, 1\}^{n_v\tau}$ and $d_i \leftarrow c_i \oplus (z|t')$ $\forall i$; compute ct^B and $\{\text{ks}_i^B, \{\text{otr}_i^w\}_{w}\}_{i \notin S}$ as in step B2, and encrypt them under K_B to A with $\{\text{r}_i^{\text{ec}}\}_{i \in S}$ and $\{\text{gc}_i\}_{i \notin S}$ from step (1);
- (5) run $\text{Rec}(\pi, (x_B, \ell_B), w_B)$ in $\text{CKEM}_{\text{LB}(\ell_B)(\text{pk})}$ on inputs set as in Π_{COMP} , output K'_A ;
- (6) on \mathbf{m}_A^2 from A , decrypt it as $\tau|t_1| \dots |t_{n_v}$ using K'_A ; if $\tau = F(K_B, 1)$ and $t_1| \dots |t_{n_v} = t'$ then send $\mathbf{m}_B^3 = F(K_B, 2) \oplus F(K'_A, 1)$ to A and $(\text{Output}, \text{sid}, B, \text{T})$ to $F_{C(f,g)}$, and otherwise send random \mathbf{m}_B^3 in $\{0, 1\}^\tau$ to A and $(\text{Output}, \text{sid}, B, \text{F})$ to $F_{C(f,g)}$.

Fig. 4. Part 1 of simulator \mathcal{A}^* , for Π_{COMP} sessions with honest party B .

For lack of space we only show the algorithm of an ideal adversary \mathcal{A}^* , i.e. the simulator, divided into two parts depending on whether the ideal adversary simulates an instance of an honest party B , shown in Fig. 4, or an honest party A , shown in Fig. 5. The proof, included in the full version of the paper [14], shows that no efficient environment can distinguish an interaction with \mathcal{A}^* and ideal honest players interacting via functionality $F_{C(f,g)}$ (where \mathcal{A}^* additionally interacts with a local copy of the real-world adversary Adv), from an interaction with Adv and real-world honest players who interact via our protocol Π_{COMP} .

On behalf of honest A , on trapdoor sk and input $(\text{Input}A, \text{sid}, A, B)$ from $F_{C(f,g)}$:

- (1) on $m_B^1 = \{\text{cgc}_i, \{\text{ck}_i^{w,b}\}_{w,b}\}_{i \in [n]}$ from B , set $x := 0^{n_x}$ and $c_i \leftarrow \{0, 1\}^{n_z + n_v \tau} \forall i$, compute $x_A := (\{\text{ct}_i^w\}_{w \in X}, \{\text{ct}_i^w\}_{i, w \in C})$, w_A and message m_A^1 as A does in step A1;
- (2) run $\text{Rec}(\pi, (x_A, w_A, \ell_A))$ in $\text{CKEM}_{\text{LA}(\ell_A)(pk)}$, let K'_B be Rec 's output;
- (3) on m_B^2 from B , decrypt it using K'_B to get ct^B , $\{r_i^{\text{gc}}\}_{i \in S}$, $\{\text{gc}_i, \text{ks}_i^B, \{\text{otr}_i^w\}_{w \in \bar{X}}\}_{i \notin S}$; decrypt each ct^w in ct^B using sk to obtain each bit of y, t, u , overwrite $y := \perp$ if any decryption output is not a bit or $u = 0$; and send $(\text{Input}B, A, y, \text{sid})$ to $F_{C(f,g)}$ and receive $(\text{Output}, v, \text{sid})$ back;
- (4) compute $(\text{gc}_i, \{k_i^{w,b}\}) \leftarrow \text{GCgen}(f|g; r_i^{\text{gc}})$ for $i \in S$ to complete statement x_B , and run $\text{Snd}(\pi, (x_B, \ell_B))$ in $\text{CKEM}_{\text{LB}(\ell_B)(pk)}$ on x_B as in Π_{COMP} , let K_A be Snd 's output;
- (5) if $v = \perp$ then set $\text{release?} := F$ and set m_A^2 as a random string, otherwise set $\text{release?} := T$ and $m_A^2 \leftarrow \text{SE}_{K_A}^0(F(K'_B, 1), t_1, \dots, t_{n_v})$ where t_i 's encode v received from $F_{C(f,g)}$ given t decrypted from ct^B above; send m_A^2 to B .
- (6) given m_B^3 , if $m_B^3 \neq F(K_B, 2) \oplus F(K'_A, 1)$ then (re)set $\text{release?} := F$; send $(\text{Output}, \text{sid}, A, \text{release?})$ to $F_{C(f,g)}$.

Fig. 5. Part 2 of simulator \mathcal{A}^* , for Π_{COMP} sessions with honest party A .

References

1. 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
2. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_8
3. Benhamouda, F., Couteau, G., Pointcheval, D., Wee, H.: Implicit zero-knowledge arguments and applications to the malicious setting. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 107–129. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_6
4. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings of 42nd IEEE Symposium on Foundations of Computer Science, FOCS 2001, pp. 136–145. IEEE Computer Society, Washington, DC (2001)
5. Chandran, N., Goyal, V., Ostrovsky, R., Sahai, A.: Covert multi-party computation. In: FOCS, pp. 238–248 (2007)
6. Cho, C., Dachman-Soled, D., Jarecki, S.: Efficient concurrent covert computation of string equality and set intersection. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 164–179. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_10
7. Couteau, G.: Revisiting covert multiparty computation. Cryptology ePrint Archive, Report 2016/951 (2016). <http://eprint.iacr.org/2016/951>

8. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_4
9. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_6
10. Goyal, V., Jain, A.: On the round complexity of covert computation. In: Proceedings of 42nd ACM Symposium on Theory of Computing, STOC 2010, pp. 191–200. ACM, New York (2010)
11. Hopper, N.J., von Ahn, L., Langford, J.: Provably secure steganography. *IEEE Trans. Comput.* **58**(5), 662–676 (2009)
12. Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 18–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_2
13. Jarecki, S.: Practical covert authentication. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 611–629. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_35
14. Jarecki, S.: Efficient covert two-party computation. Cryptology ePrint Archive, Report 2016/1032 (2016). <http://eprint.iacr.org/2016/1032>
15. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_18
16. Kreuter, B., Shelat, A., Shen, C.-H.: Billion-gate secure computation with malicious adversaries. In: USENIX Security, pp. 14–25 (2012)
17. Lindell, Y.: Fast cut-and-choose-based protocols for malicious and covert adversaries. *J. Cryptol.* **29**(2), 456–490 (2016)
18. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_4
19. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. *J. Cryptol.* **25**(4), 680–722 (2012)
20. Manulis, M., Pinkas, B., Poettering, B.: Privacy-preserving group discovery with linear complexity. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 420–437. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13708-2_25
21. Mohassel, P., Franklin, M.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_30
22. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. *J. Cryptol.* **18**(1), 1–35 (2005)
23. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Proceedings of 36th Annual ACM Symposium on Theory of Computing, 13–16 June 2004, Chicago, IL, USA, pp. 232–241 (2004)

24. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_15
25. Raimondo, M.D., Gennaro, R.: Provably secure threshold password-authenticated key exchange. *J. Comput. Syst. Sci.* **72**(6), 978–1001 (2006)
26. Shelat, A., Shen, C.: Two-output secure computation with malicious adversaries. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 386–405. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_22
27. Shelat, A., Shen, C.-H.: Fast two-party secure computation with minimal assumptions. In: ACM Conference on Computer and Communications Security - CCS, pp. 523–534 (2013)
28. von Ahn, L., Hopper, N., Langford, J.: Covert two-party computation. In: Proceedings of 37th Annual ACM Symposium on Theory of Computing, STOC 2005, pp. 513–522. ACM, New York (2005)
29. Yao, A.: How to generate and exchange secrets. In: 27th FOCS, pp. 162–167 (1986)



Towards Characterizing Securely Computable Two-Party Randomized Functions

Deepesh Data^(✉) and Manoj Prabhakaran

Department of Computer Science and Engineering,
Indian Institute of Technology Bombay, Mumbai, India
deepesh.data@gmail.com

Abstract. A basic question of cryptographic complexity is to combinatorially characterize all randomized functions which have information-theoretic semi-honest secure 2-party computation protocols. The corresponding question for deterministic functions was answered almost three decades back, by Kushilevitz [Kus89]. In this work, we make progress towards understanding securely computable *randomized* functions. We bring tools developed in the study of completeness to bear on this problem. In particular, our characterizations are obtained by considering only symmetric functions with a combinatorial property called *simplicity* [MPR12].

Our main result is a complete combinatorial characterization of randomized functions *with ternary output* kernels, that have information-theoretic semi-honest secure 2-party computation protocols. In particular, we show that there exist simple randomized functions with ternary output that do not have secure computation protocols. (For deterministic functions, the smallest output alphabet size of such a function is 5, due to an example given by Beaver [Bea89].)

Also, we give a complete combinatorial characterization of randomized functions that have *2-round* information-theoretic semi-honest secure 2-party computation protocols.

We also give a counter-example to a natural conjecture for the full characterization, namely, that all securely computable simple functions have secure protocols with a unique transcript for each output value. This conjecture is in fact true for deterministic functions, and – as our results above show – for ternary functions and for functions with 2-round secure protocols.

1 Introduction

Understanding the nature of secure multiparty computation has been a key problem in modern cryptography, ever since the notion was introduced. While

D. Data—This work was done while the author was a Ph.D. student at the Tata Institute of Fundamental Research in Mumbai.

this has been a heavily researched area, some basic problems have remained open. In this work we explore the following fundamental question:

Which randomized functions have information-theoretic semi-honest secure 2-party computation protocols?

The corresponding question for deterministic functions was answered almost three decades back, by Kushilevitz [Kus89] (originally, restricted to symmetric functions, in which both parties get the same output). The dual question of which functions are complete, initiated by Kilian [Kil88], has also been fully resolved, for semi-honest [MPR12] and even active security [KMPS14]. However, the above question itself has seen little progress since 1989.

In this work, we make progress towards understanding securely computable *randomized* functions. (Throughout this paper, security will refer to information-theoretic semi-honest security.) We bring tools developed in the study of completeness to bear on this problem. In particular, our characterizations are obtained by considering only symmetric functions with a combinatorial property called *simplicity* [MPR12]. (As shown in [MPR12], a function is semi-honest securely computable if and only if it is simple, and a related function called its “kernel” – which is always a simple function – is securely computable.)

One may start off by attempting to generalize the result of Kushilevitz [Kus89] so that it applies to randomized functions as well. This characterization showed that any securely computable deterministic function has a secure protocol in which the two parties take turns to progressively reveal more and more information about their respective inputs – by restricting each input to smaller and smaller subsets – until there is exactly enough information to evaluate the function. However, a naïve generalization of this result fails for randomized functions, as it is possible for a securely computable function to have every output value in the support of every input combination; thus the input spaces cannot be shrunk at all during a secure protocol.

A more fruitful approach would be to consider the protocol for deterministic functions as partitioning the *output space* at each step, and choosing one of the parts. This is indeed true when considering deterministic functions which are *simple*. Such a protocol results in a unique transcript for each output value. An *a priori* promising conjecture would be that every securely computable simple function – deterministic or randomized – has such a unique-transcript protocol. Unfortunately, this conjecture turns out to be false.

However, for small output alphabets, we can prove that this conjecture holds. Indeed, we show that the exact threshold on the alphabet size where this conjecture breaks down is 4. When the output alphabet size of a simple function is at most 3, we give a combinatorial characterization of secure computability; our characterization implies that such functions do have unique-transcript secure protocols. We also characterize simple functions which have two-round secure protocols, which again turn out to all have unique-transcript protocols.

We leave the full characterization as an important open problem.

Our Results

- Our main result is a complete combinatorial characterization of randomized functions *with ternary output* kernels, that have information-theoretic semi-honest secure 2-party computation protocols. In particular, we show that there exist simple randomized functions with ternary output that do not have secure computation protocols. (For deterministic functions, the smallest output alphabet size of such a function is 5, due to an example given by Beaver [Bea89].)
- We also give a complete combinatorial characterization of randomized functions that have *2-round* information-theoretic semi-honest secure 2-party computation protocols.
- We also give a counter-example to a natural conjecture for the full characterization, namely, that all securely computable simple functions have secure protocols with a unique transcript for each output value. This conjecture is in fact true for deterministic functions, and – as our results above show – for ternary functions and for functions with 2-round secure protocols.

1.1 Technical Overview

Prior work [MPR12, MPR13] lets us focus on symmetric functions: A randomized function F is securely realizable if and only if it is “isomorphic” – i.e., essentially equivalent, up to sampling additional local outputs – to a symmetric function G called its kernel, and G itself has a secure protocol; see Theorem 3. Further the kernel of F is easy to find and explicitly defined in Definition 2. Hence the problem of characterizing secure computability of general randomized functions reduces to the problem of characterizing secure computability of randomized functions which are kernels. Such functions are symmetric and simple (a symmetric function G is simple, if $\Pr[G(x, y) = z] = \rho(x, z) \cdot \sigma(y, z)$ for some fixed functions $\rho : \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}_+$ and $\sigma : \mathcal{Y} \times \mathcal{Z} \rightarrow \mathbb{R}_+$, where \mathcal{X} and \mathcal{Y} are Alice’s and Bob’s input domains and \mathcal{Z} is their common output domain). As such, we work with symmetric and simple functions.

Characterizing Ternary-Kernel Functions. Our main result could be stated as follows:

Theorem 1. *If a randomized function F has a kernel G with an output alphabet of size at most 3, then F is securely computable if and only if F is simple and there is some ordering of G ’s output alphabet \mathcal{Z} as (z_1, z_2, z_3) and two functions $q : \mathcal{X} \rightarrow [0, 1]$ and $r : \mathcal{Y} \rightarrow [0, 1]$, such that one of the following holds:*

$$\begin{aligned} \Pr[G(x, y) = z_1] &= q(x) & \Pr[G(x, y) = z_1] &= r(y) \\ \Pr[G(x, y) = z_2] &= (1 - q(x)) \cdot r(y) & \text{or } \Pr[G(x, y) = z_2] &= (1 - r(y)) \cdot q(x) \\ \Pr[G(x, y) = z_3] &= (1 - q(x)) \cdot (1 - r(y)) & \Pr[G(x, y) = z_3] &= (1 - r(y)) \cdot (1 - q(x)) \end{aligned}$$

Note that if the first set of conditions holds, there is a secure protocol in which Alice either sends z_1 as the output to Bob (with probability $q(x)$) or asks Bob to pick the output; if Bob is asked to pick the output he sends back either z_2 as the output (with probability $r(y)$) or z_3 otherwise. If the second condition holds there is a symmetric protocol with Bob making the first move.

Surprisingly, these are the only two possibilities for G to have a secure protocol. To prove this, however, takes a careful analysis of the linear-algebraic constraints put on a protocol by the security definition and the fact that the function is simple. We start by observing that a secure protocol for a symmetric simple function must have a simulator that can simulate the transcript of an execution merely from the output (rather than the corrupt party's input and the output).

¹ Then, supposing that the first message in the protocol is a single bit sent by Alice, we identify that there is a quantity independent of either party's input, denoted by $\phi(z)$, that gives the probability of the first message being 0, conditioned on the output being z . Specifying these quantities, $\phi(z)$ at each round fully specifies the protocol. We show that $\sum_{z \in \mathcal{Z}} \Pr[G(x, y) = z] \cdot \phi(z) = q(x)$, a quantity independent of y . By carefully analyzing the constraints arising from these equations, we prove Theorem 1.

Characterizing Functions Having 2-Round Protocols. Our second result is as follows:

Theorem 2. *A function F has a 2-round secure protocol iff its kernel has a 2-round unique-transcript protocol.*

Observe that F has a 2-round secure protocol iff its kernel has one too, as F is isomorphic to its kernel and a secure protocol for a function can be transformed to one for an isomorphic function without changing the communication involved. What needs to be proven is that if the kernel (or any symmetric simple function) has a 2-round secure protocol, then it has a 2-round unique-transcript protocol. We do this by identifying an equivalence relation among the outputs, such that any 2-round protocol with (say) Alice making the first move will have Alice's input only influencing which equivalence class of the output is chosen, and then Bob's input influences which output is chosen, given its equivalence class.

1.2 Related Work

There has been a large body of work regarding the complexity of secure multi-party and 2-party computation. [MPR13] surveys many of the important results in the area. Kushilevitz [Kus89] gave a combinatorial characterization of securely computable two-party *deterministic* functions (with perfect security), along with

¹ This is not true for every symmetric function. For instance, in a semi-honest secure protocol for XOR, the transcript must necessarily reveal both parties' inputs, but this cannot be simulated from the output without knowing one party's input. A function like XOR is not simple, though it is isomorphic to one.

a generic round-optimal secure protocol for functions that satisfy the characterization condition. This was later extended to statistical security and also to security against active corruption [MPR09, KMQR09].

Among randomized functions in which only Bob gets any output, simple functions have a deterministic kernel corresponding to a function of Alice's input alone, and hence are always securely computable. This observation was already made by Kilian [Kil00]. Recently, Data [Dat16] considered the same problem with a probability distribution on the inputs, and gave communication-optimal protocols in different security settings.

2 Preliminaries

A general randomized function is denoted by a conditional probability distribution $p_{Z_A Z_B | XY}$, where X, Y, Z_A, Z_B take values in finite alphabets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}_A, \mathcal{Z}_B$, respectively. In a protocol for computing this function, when Alice and Bob are given inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively, they should output $z_A \in \mathcal{Z}_A$ and $z_B \in \mathcal{Z}_B$ respectively, such that (z_A, z_B) is distributed according to $p_{Z_A Z_B | X=x, Y=y}$.

Notation. We shall consider various discrete random variables (inputs, outputs, protocol messages). We denote the probability mass function of a random variable U by p_U . For random variables (U, V) , we denote the conditional probability mass function of U conditioned on V by $p_{U|V}$. $[n]$ denotes the set $\{1, \dots, n\}$.

Protocols. We consider computationally unbounded two-party protocols, without any setup. Such a protocol Π is fully defined by the input and output domains, the *next message functions* and the *output functions* for Alice and Bob. Let $(\mathcal{X}, \mathcal{Z}_A, \text{next}_{\Pi}^A, \text{out}_{\Pi}^A)$ and $(\mathcal{Y}, \mathcal{Z}_B, \text{next}_{\Pi}^B, \text{out}_{\Pi}^B)$ denote the input domain, output domain, the next message function and the output function, for Alice and Bob, respectively. The functions are all potentially randomized. next_{Π}^A and next_{Π}^B output the next message given the transcript so far and the local input. (Note that in the information-theoretic setting, protocols need not maintain local state.) Similarly, out_{Π}^A and out_{Π}^B map the transcript and local input to a local output in \mathcal{Z}_A and \mathcal{Z}_B , respectively.

In running a protocol, Alice and Bob are first given inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively. Then they take turns sending messages to each other according to their next message functions, and in the end (recognizable by both parties) each party produces an output according to its output function. We assume that the protocol terminates with probability one after a finite number rounds.

Running a protocol induces a distribution over the set of all possible (complete) transcripts, \mathcal{M} . The protocol Π induces a conditional probability distribution $\Pr_{\Pi}[m|x, y]$, for every input $x \in \mathcal{X}, y \in \mathcal{Y}$ and every $m \in \mathcal{M}$. Suppose that $m = (m_1, m_2, \dots)$ is the transcript generated by Π when parties have inputs x and y , where m_i 's, for odd i , are sent by Alice, and m_i 's, for even i , are sent by Bob. Note that during the execution of a protocol, a message sent

by any party is determined by its input and all the messages it has exchanged so far; and conditioned on these two, the message is independent of the other party’s input. Using this we can write $\Pr_{\Pi}[m|x, y] = \alpha(m, x)\beta(m, y)$, where $\alpha(m, x) = \prod_{i:i \text{ is odd}} \Pr_{\Pi}[m_i|m_{<i}, x]$, and $\beta(m, y) = \prod_{i:i \text{ is even}} \Pr_{\Pi}[m_i|m_{<i}, y]$.

Secure Protocols. Throughout this paper security refers to *information-theoretic semi-honest security*. We restrict ourselves to finite functions and perfect security. We remark that all our results can be extended to statistical security, as we shall show in the full version.

Definition 1. A protocol Π for computing a function $p_{Z_A Z_B|XY}$, with transcript space \mathcal{M} , is said to be (perfectly semi-honest) secure iff there exist functions $S_1 : \mathcal{M} \times \mathcal{X} \times \mathcal{Z}_A \rightarrow [0, 1]$ and $S_2 : \mathcal{M} \times \mathcal{Y} \times \mathcal{Z}_B \rightarrow [0, 1]$ such that $\Pr_{\Pi}[m|x, y, z_A, z_B] = S_1(m, x, z_A) = S_2(m, y, z_B)$ for all $m \in \mathcal{M}$ and x, y, z_A, z_B such that $p_{Z_A Z_B|XY}(z_A, z_B|x, y) > 0$.

Kernel and Simple Functions. Maji et al. [MPR12] simplified the secure computation of a general randomized function $p_{Z_A Z_B|XY}$ to secure computation of a symmetric randomized function $p_{Z|XY}$. For that they defined *weighted characteristic bipartite graph* of a randomized function $p_{Z_A Z_B|XY}$ as $\mathcal{G}(p_{Z_A Z_B|XY}) = (V, E, \text{wt})$, where

- $V = (\mathcal{X} \times \mathcal{Z}_A) \cup (\mathcal{Y} \times \mathcal{Z}_B)$,
- $E = \{((x, z_A), (y, z_B)) : p_{Z_A Z_B|XY}(z_A, z_B|x, y) > 0\}$, and
- the weight function $\text{wt}:(\mathcal{X} \times \mathcal{Z}_A) \times (\mathcal{Y} \times \mathcal{Z}_B) \rightarrow [0, 1]$ is defined as

$$\text{wt}((x, z_A), (y, z_B)) := \frac{p_{Z_A Z_B|XY}(z_A, z_B|x, y)}{|\mathcal{X}| \times |\mathcal{Y}|}.$$

Note that if $((x, z_A), (y, z_B)) \notin E$, then $\text{wt}((x, z_A), (y, z_B)) = 0$.

Let k be the number of connected components in the above-defined graph. We say that $\mathcal{G}(p_{Z_A Z_B|XY}) = (V, E, \text{wt})$ is a *product distribution graph*, if there exist probability distributions p over $\mathcal{X} \times \mathcal{Z}_A$, q over $\mathcal{Y} \times \mathcal{Z}_B$, and c over $[k]$, such that for all $(x, z_A) \in \mathcal{X} \times \mathcal{Z}_A$ and $(y, z_B) \in \mathcal{Y} \times \mathcal{Z}_B$, if $((x, z_A), (y, z_B))$ lies in the j^{th} connected component of $\mathcal{G}(p_{Z_A Z_B|XY}) = (V, E, \text{wt})$, then $\text{wt}((x, z_A), (y, z_B)) = p(x, z_A) \cdot q(y, z_B)/c_j$, otherwise $\text{wt}((x, z_A), (y, z_B)) = 0$.

Definition 2 (Kernel – Common-information in a randomized function [MPR12]). The kernel of a randomized function $p_{Z_A Z_B|XY}$ is a symmetric randomized function, which takes x and y from the parties and samples (z_A, z_B) according to $p_{Z_A Z_B|X=x, Y=y}$. Then it outputs to both parties the connected component of $\mathcal{G}(p_{Z_A Z_B|XY})$ which contains the edge $((x, z_A), (y, z_B))$.

Note that the kernel of $p_{Z_A Z_B|XY}$ is a symmetric randomized function. We denote it by $p_{Z|XY}$, where the alphabet of Z is $\mathcal{Z} = \{z_1, z_2, \dots, z_k\}$, where k is the number of connected components in $\mathcal{G}(p_{Z_A Z_B|XY})$. The kernel $p_{Z|XY}$ is defined as follows: for every $j \in [k]$, $p_{Z|XY}(z_j|x, y) :=$

$\sum_{(z_A, z_B)} p_{Z_A Z_B | XY}(z_A, z_B | x, y)$, where summation is taken over all (z_A, z_B) 's such that $((x, z_A), (y, z_B))$ lies in the j^{th} connected component in $\mathcal{G}(p_{Z_A Z_B | XY})$. The following theorem was proved in [MPR12].

Theorem 3 [MPR12, Theorem 3]. *A randomized function $p_{Z_A Z_B | XY}$ is securely computable if and only if $\mathcal{G}(p_{Z_A Z_B | XY})$ is a product distribution graph and the kernel of $p_{Z_A Z_B | XY}$ is securely computable.*

Theorem 3 reduces secure computability of $p_{Z_A Z_B | XY}$ to secure computability of the kernel of $p_{Z_A Z_B | XY}$ and a simple combinatorial check on $p_{Z_A Z_B | XY}$ (which is to check whether the weighted characteristic bipartite graph of $p_{Z_A Z_B | XY}$ is a product distribution graph or not).²

Definition 3. *A symmetric randomized function $p_{Z | XY}$ is said to be simple if there exist two functions $\rho : \mathcal{X} \times \mathcal{Z} \rightarrow \mathbb{R}_+$ and $\sigma : \mathcal{Y} \times \mathcal{Z} \rightarrow \mathbb{R}_+$ such that for all $x \in \mathcal{X}, y \in \mathcal{Y}$, and $z \in \mathcal{Z}$, $p_{Z | XY}(z | x, y) = \rho(x, z) \cdot \sigma(y, z)$.*

Here \mathbb{R}_+ denotes the set of non-negative real numbers. For deterministic functions, instead of \mathbb{R}_+ , one can take $\{0, 1\}$ in the above definition.

Remark 1. The original definition of a simple function given in [MPR12] seems to be different from our definition. There it was defined for a general randomized function $p_{Z_A Z_B | XY}$, which defined simplicity in terms of isomorphism between a function and its kernel, whereas we defined simplicity for symmetric functions only. Since isomorphic functions are essentially equivalent – up to sampling additional local outputs – and the kernel of a general randomized function is a symmetric function, our definition of simplicity is equivalent to the one given in [MPR12].

Note that the Kernel of a securely computable randomized function $p_{Z_A Z_B | XY}$ is a simple function. As shown in [MPR12], a secure protocol for the kernel can be transformed to one for the original function itself, and vice versa, without changing the communication involved. Thus we shall focus on characterizing secure computability for kernel functions, which are all symmetric, simple functions.

The combinatorial definition of simplicity above will be crucially used in our analysis. Indeed, the factorization property clarifies otherwise obscure connections and elusive constraints.

For protocols for simple and symmetric functions, the output can be written as a function merely of the transcript and we can simulate the transcript just based on the (common) output, without needing either party's input. We prove the following lemma in Appendix A.

Lemma 1. *If Π is a perfectly semi-honest secure protocol for a simple symmetric function $p_{Z | XY}$, then there are (deterministic) functions $\text{out}_\Pi : \mathcal{M} \rightarrow \mathcal{Z}$ and $S : \mathcal{M} \rightarrow [0, 1]$ such that for all $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$ and $m \in \mathcal{M}$,*

$$\begin{aligned} \text{out}_\Pi^A(m, x) = \text{out}_\Pi^B(m, y) = \text{out}_\Pi(m) & \quad \text{if } \Pr_\Pi[m | x, y] > 0, \\ \Pr_\Pi[m | x, y, z] = S(m, z) & \quad \text{if } p_{Z | XY}(z | x, y) > 0. \end{aligned}$$

² There are other easier checks; see [MPR12, Lemma 1] for details.

Note that above, if $z \neq \text{out}_\Pi(m)$, $S(m, z) = \Pr_\Pi[m|x, y, z] = 0$. By writing $\mu(m) = S(m, \text{out}_\Pi(m))$ we have the following: for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, $z \in \mathcal{Z}$ s.t. $p_{Z|XY}(z|x, y) > 0$, and all $m \in \mathcal{M}$, we have

$$\Pr_\Pi[m|x, y, z] = \begin{cases} \mu(m) & \text{if } \text{out}_\Pi(m) = z \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Thus, for each $z \in \mathcal{Z}$ (such that for some (x, y) , $p_{Z|XY}(z|x, y) > 0$), μ defines a probability distribution over $\{m \in \mathcal{M} : \text{out}_\Pi(m) = z\}$. Also, since $\Pr_\Pi[m|x, y] = \Pr_\Pi[m, z|x, y]$, for $z = \text{out}_\Pi(m)$, and $\Pr_\Pi[m, z|x, y] = \Pr_\Pi[m|x, y, z] \cdot \Pr_\Pi[z|x, y] = \mu(m) \cdot p_{Z|XY}(z|x, y)$ we have, for all $x \in \mathcal{X}, y \in \mathcal{Y}, m \in \mathcal{M}$,

$$\Pr_\Pi[m|x, y] = \mu(m) \cdot p_{Z|XY}(\text{out}_\Pi(m)|x, y). \tag{2}$$

A Normal Form for $p_{Z|XY}$. For a symmetric randomized functionality $p_{Z|XY}$, we define the relation $x \equiv x'$ for $x, x' \in \mathcal{X}$ to hold, if $\forall y \in \mathcal{Y}, z \in \mathcal{Z}, p(z|x, y) = p(z|x', y)$; similarly we define $y \equiv y'$ for $y, y' \in \mathcal{Y}$. We define $z \equiv z'$ for $z, z' \in \mathcal{Z}$, if there exists a constant $c > 0$ such that $\forall x \in \mathcal{X}, y \in \mathcal{Y}, p(z|x, y) = c \cdot p(z'|x, y)$. We say that $p_{Z|XY}$ is in normal form if $x \equiv x' \Rightarrow x = x', y \equiv y' \Rightarrow y = y',$ and $z \equiv z' \Rightarrow z = z'$.

It is easy to see that any $p_{Z|XY}$ can be transformed into one in normal form $p_{Z^*|X^*Y^*}$ with possibly smaller alphabets, so that $p_{Z|XY}$ is securely computable if and only if $p_{Z^*|X^*Y^*}$ is securely computable. We will assume in this paper that $p_{Z|XY}$ is in normal form.

For the ease of notation, in this paper we often denote a randomized function $p_{Z|XY}$ by an equivalent function f , such that $f(x, y, z) = p_{Z|XY}(z|x, y)$, for every $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$. We may use f and $p_{Z|XY}$ interchangeably.

Unique-Transcript Protocols. A unique transcript protocol Π for a symmetric function is one in which each output $z \in \mathcal{Z}$ has a unique transcript that can result in it: i.e., for every $z \in \mathcal{Z}$, there is at most one $m \in \mathcal{M}$ such that $\text{out}_\Pi(m) = z$. Such a protocol is always a secure protocol for the function it computes (and has a deterministic simulator, which when given an output z assigns probability 1 to the unique transcript m such that $\text{out}_\Pi(m) = z$).

It follows from [Kus89] that every securely computable simple *deterministic* function has a unique-transcript secure protocol. The subset of securely computable simple randomized functions that we characterize also have unique-transcript protocols. However, we shall show an example of a securely computable function (just outside the sets we characterize) which does not have any unique-transcript protocol. Understanding such functions remains the next step in fully characterizing securely computable randomized functions.

3 Characterization of Functions up to Ternary Output Alphabet

As mentioned earlier, we shall represent a randomized function $p_{Z|XY}$ by an equivalent function $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \rightarrow [0, 1]$, such that $f(x, y, z) = p_{Z|XY}(z|x, y)$.

For a simple function f , we shall write $f = (\rho, \sigma)$ where $f(x, y, z) = \rho(x, z) \cdot \sigma(y, z)$. Note that for every $x \in \mathcal{X}, y \in \mathcal{Y}$ we have $\sum_{z \in \mathcal{Z}} f(x, y, z) = 1$.

3.1 Compact Representation of Secure Protocols

We assume, w.l.o.g., that both Alice and Bob send only binary messages to each other in every round, and only one party sends a message in one round. Suppose Π is a protocol that securely computes a simple function $f = (\rho, \sigma)$. In this section we consider protocols of an arbitrary number of rounds, and so, w.l.o.g., we assume that in Π Alice and Bob take turns exchanging single bits and that Alice sends the first message in Π .

Let $q : \mathcal{X} \rightarrow [0, 1]$ be such that, Alice, given an input x chooses 0 as her first message with probability $q(x)$ (and 1 with probability $1 - q(x)$). We define $\Pi^{(0)}$ to be the protocol, with input spaces $\mathcal{X}^{(0)} = \{x \in \mathcal{X} : q(x) > 0\}$ and \mathcal{Y} , in which Alice's first message is redefined to be 0 with probability 1 for every input $x \in \mathcal{X}^{(0)}$; otherwise $\Pi^{(0)}$ has identical next message and output functions as Π . Let $f^{(0)}$ be the function computed by $\Pi^{(0)}$: i.e., $f^{(0)}(x, y, z) = \Pr_{\Pi^{(0)}}[z|x, y]$ for all $x \in \mathcal{X}^{(0)}, y \in \mathcal{Y}, z \in \mathcal{Z}$. $f^{(1)}$ is defined symmetrically.

Claim 1. *There exists a function $\phi : \mathcal{Z} \rightarrow [0, 1]$ such that for all $x \in \mathcal{X}, y \in \mathcal{Y}$,*

$$q(x) = \sum_{z \in \mathcal{Z}} \phi(z) f(x, y, z). \tag{3}$$

Further, $f^{(0)}(x, y, z) = \frac{\phi(z)}{q(x)} \cdot f(x, y, z)$, for all $x \in \mathcal{X}^{(0)}, y \in \mathcal{Y}, z \in \mathcal{Z}$.

Proof. We define

$$\phi(z) := \sum_{\substack{m: m_1=0, \\ \text{out}_{\Pi}(m)=z}} \mu(m), \tag{4}$$

where $\mu(m)$ is as defined in (1). Here m_1 denotes the first bit in the transcript m . Note that we have $\phi(z) \in [0, 1]$ because for each z , μ defines a probability distribution over $\mathcal{M}_z := \{m : \text{out}_{\Pi}(m) = z\}$ and $\phi(z)$ sums up the probabilities for a subset of \mathcal{M}_z .

To see that ϕ satisfies the claim, note that $\sum_{z \in \mathcal{Z}} \phi(z) f(x, y, z) = \sum_{m: m_1=0} \mu(m) \cdot f(x, y, \text{out}_{\Pi}(m))$. Now, from (2), $\mu(m) \cdot f(x, y, \text{out}_{\Pi}(m)) = \Pr_{\Pi}[m|x, y]$. Hence,

$$\sum_{z \in \mathcal{Z}} \phi(z) f(x, y, z) = \sum_{m: m_1=0} \Pr_{\Pi}[m|x, y] = q(x).$$

Also, for all $x \in \mathcal{X}^{(0)}, y \in \mathcal{Y}, z \in \mathcal{Z}$,

$$\begin{aligned} f^{(0)}(x, y, z) &= \sum_{\substack{m: \\ \text{out}_{\Pi}(m)=z}} \Pr_{\Pi^{(0)}}[m|x, y] = \sum_{\substack{m: m_1=0, \\ \text{out}_{\Pi}(m)=z}} \frac{\Pr_{\Pi}[m|x, y]}{q(x)} = \sum_{\substack{m: m_1=0, \\ \text{out}_{\Pi}(m)=z}} \mu(m) \frac{f(x, y, z)}{q(x)} \\ &= \frac{f(x, y, z)}{q(x)} \phi(z). \end{aligned}$$

□

Note that since $\mu(m)$ is the probability of the transcript being m given that the output is $\text{out}_\Pi(m)$, (4) gives that for every $z \in \mathcal{Z}$, $\phi(z) = \Pr[m_1 = 0 | \text{out}_\Pi(m) = z]$, i.e., the probability of the first message being 0, conditioned on the output being z . It follows from Claim 1 that a secure protocol is completely and compactly described by the values of $(\phi(z))_{z \in \mathcal{Z}}$ similarly defined in every round.

Remark 2. If $\phi(z) = c$ for all $z \in \mathcal{Z}$, then $q(x) = c$ for all $x \in \mathcal{X}$ (because $\sum_{z \in \mathcal{Z}} f(x, y, z) = 1$ for all (x, y)). Further, if $c > 0$, $f^{(0)} = f$, and if $c < 1$, $f^{(1)} = f$. This corresponds to a protocol in which Alice sends an inconsequential first message, which neither depends on her input, nor influences the output. Hence, if Π is round-optimal, it cannot be the case that $\phi(z) = c$ for all $z \in \mathcal{Z}$.

Note that (3) holds for every $y \in \mathcal{Y}$. Hence, we obtain

$$\sum_{z \in \mathcal{Z}} (f(x, y, z) - f(x, y', z))\phi(z) = 0 \quad \forall x \in \mathcal{X}, y, y' \in \mathcal{Y}. \tag{5}$$

For each $x \in \mathcal{X}$, this gives a system of $|\mathcal{Y}| - 1$ equations, by choosing a fixed $y \in \mathcal{Y}$ and all $y' \in \mathcal{Y} \setminus \{y\}$ (one may write the equations resulting from other choices of y, y' as linear combinations of these $|\mathcal{Y}| - 1$ equations).

3.2 Binary Output Alphabet

Theorem 4. *If $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \rightarrow [0, 1]$ is a simple function with $|\mathcal{Z}| = 2$, then f is securely computable.*

Proof. In fact, we can prove a stronger statement: if f is as above, then $f(x, y, z)$ is either independent of x or independent of y (or both). In that case, clearly f is securely computable by a protocol in which one party computes the output and sends it to the other party.

Since f is simple, we have $f(x, y, z) = \rho(x, z) \cdot \sigma(y, z)$, for all $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$. Since $|\mathcal{Z}| = 2$, we let $\mathcal{Z} = \{0, 1\}$, and abbreviate $\rho(x, z)$ as $\rho_z(x)$ and $\sigma(y, z)$ as $\sigma_z(y)$ (for $z \in \{0, 1\}$). Note that we have the following system of equations:

$$\rho_0(x)\sigma_0(y) + \rho_1(x)\sigma_1(y) = 1 \quad \forall (x, y) \in \mathcal{X} \times \mathcal{Y} \tag{6}$$

We consider 3 cases:

Case 1: $\exists x, x' \in \mathcal{X}, x \neq x', \rho_0(x)\rho_1(x') \neq \rho_0(x')\rho_1(x)$. In this case, one can solve the system in (6) to get two values s_0, s_1 such that $\sigma_0(y) = s_0$ and $\sigma_1(y) = s_1$ for all $y \in \mathcal{Y}$. (Concretely, $s_0 = \frac{\rho_1(x') - \rho_1(x)}{\Delta}$ and $s_1 = \frac{\rho_0(x) - \rho_0(x')}{\Delta}$, where $\Delta = \rho_0(x)\rho_1(x') - \rho_0(x')\rho_1(x) \neq 0$.) Hence $f(x, y, z) = \rho(x, z) \cdot s_z$, depends only on x .

Case 2: $\forall x, x' \in \mathcal{X}, \rho_0(x)\rho_1(x') = \rho_0(x')\rho_1(x)$, and $\exists x \in \mathcal{X}, \rho_0(x) = 0$. In this case we shall show that, $\forall x' \in \mathcal{X}, \rho_0(x') = 0$. Hence the function is the constant, deterministic function, with $f(x, y, 0) = 0$ for all (x, y) .

Let x be such that $\rho_0(x) = 0$. Since $\rho_0(x)\sigma_0(y) + \rho_1(x)\sigma_1(y) = 1$ (for any $y \in \mathcal{Y}$), we have $\rho_1(x) \neq 0$. Then, since $\forall x' \in \mathcal{X}, \rho_0(x)\rho_1(x') = \rho_0(x')\rho_1(x)$, we have $0 = \rho_0(x')\rho_1(x)$ which implies $\rho_0(x') = 0$, as claimed.

Case 3: $\forall x, x' \in \mathcal{X}, \rho_0(x)\rho_1(x') = \rho_0(x')\rho_1(x)$, and $\forall x \in \mathcal{X}, \rho_0(x) \neq 0$. In this case, $\forall x, x' \in \mathcal{X}, \frac{\rho_1(x)}{\rho_0(x)} = \frac{\rho_1(x')}{\rho_0(x')} = \theta$, say. Then, from (6), we have that $\forall x \in \mathcal{X}, y \in \mathcal{Y}, \rho_0(x) = \frac{1}{\sigma_0(y) + \theta\sigma_1(y)}$ and $\rho_1(x) = \frac{\theta}{\sigma_0(y) + \theta\sigma_1(y)}$. Since the RHS in these two expressions do not depend on x , there are constants r_0, r_1 such that for all $x \in \mathcal{X}, \rho_0(x) = r_0$ and $\rho_1(x) = r_1$. Hence $f(x, y, z) = r_z \cdot \sigma(y, z)$, depends only on y . \square

3.3 Ternary Output Alphabet

To prove Theorem 1, we can focus on kernel functions, or simple symmetric functions. Let $\mathcal{Z} = \{z_1, z_2, z_3\}$. For a given symmetric function f with \mathcal{Z} as its output alphabet, we define two binary functions \hat{f}_i and f_i , for any $i \in [3]$, as follows:

1. Output alphabet of \hat{f}_i is $\{z_i, z_*\}$. For every $x \in \mathcal{X}, y \in \mathcal{Y}$, we define $\hat{f}_i(x, y, z_i) := f(x, y, z_i)$ and $\hat{f}_i(x, y, z_*) := \sum_{j \neq i} f(x, y, z_j)$.
2. Output alphabet of f_i is $\mathcal{Z} \setminus \{z_i\}$. For every $x \in \mathcal{X}, y \in \mathcal{Y}$ for which $f(x, y, z_i) < 1$, we define, $f_i(x, y, z_j) := f(x, y, z_j) / (1 - f(x, y, z_i))$, for $j \in [3] \setminus \{i\}$. If $f(x, y, z_i) = 1$ for some $x \in \mathcal{X}, y \in \mathcal{Y}$, we leave $f_i(x, y, z_i)$ undefined.

Note that if for some $\tilde{x} \in \mathcal{X}, f(\tilde{x}, y, z_i) = 1, \forall y \in \mathcal{Y}$, then we need not define f_i for this particular \tilde{x} , because $f(\tilde{x}, y, z_j) = 0, \forall j \neq i$ and for all $y \in \mathcal{Y}$. Similarly, if for some $\tilde{y} \in \mathcal{Y}, f(x, \tilde{y}, z_i) = 1, \forall x \in \mathcal{X}$, then we need not define f_i for this particular \tilde{y} , because $f(x, \tilde{y}, z_j) = 0, \forall j \neq i$ and for all $x \in \mathcal{X}$. In the following, when we say that f_i is securely computable, it must be defined for all inputs.

Theorem 5. *Suppose f is simple and in normal form. Then, f is securely computable if and only if there exists an $i \in [3]$ such that both \hat{f}_i and f_i are simple.*

Remark 3. Since \hat{f}_i and f_i are binary functions, being simple implies that they are functions of only one party’s input (see proof of Theorem 4). Theorem 1 follows by considering the different possibilities of which party’s input they can each depend on.

Remark 4. In the case of functions with a binary output alphabet, we proved in Subsect. 3.2 that if a function is simple, it is securely computable. However, Theorem 5 lets us show that this is not true in general (see in Sect. 5.1).

Proof of Theorem 5. If $|\mathcal{X}| = 1$ or $|\mathcal{Y}| = 1$, then the theorem is trivially true. So, in the following we assume that $|\mathcal{X}|, |\mathcal{Y}| \geq 2$. First we show the only if (\Rightarrow) part, and then the if (\Leftarrow) part.

\Rightarrow : Suppose f is securely computable. Since f is simple, we can write $f(x, y, z) = \rho(x, z)\sigma(y, z)$. Fix a round-optimal secure protocol Π for f . Below, we assume that Alice sends the first message in Π (the case when Bob sends the

first message being symmetric). Let ϕ be as in Claim 1. Since Π is round-optimal, as discussed in Remark 2,

$$\exists i, j \in [3] \text{ s.t. } \phi(z_i) \neq \phi(z_j). \tag{7}$$

For $i \in [3]$ and $x \in \mathcal{X}, y, y' \in \mathcal{Y}$, we define

$$\nabla f_i^{x,y,y'} := f(x, y, z_i) - f(x, y', z_i). \tag{8}$$

It follows from (5) and (8) that, $\forall x \in \mathcal{X}, y, y' \in \mathcal{Y}$:

$$\nabla f_1^{x,y,y'} \phi(z_1) + \nabla f_2^{x,y,y'} \phi(z_2) + \nabla f_3^{x,y,y'} \phi(z_3) = 0. \tag{9}$$

Since $\sum_{z \in \mathcal{Z}} f(x, y, z) = 1$ holds for every $x \in \mathcal{X}, y \in \mathcal{Y}$, we have that $\sum_{i=1}^3 \nabla f_i^{x,y,y'} = 0$, for every $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$. Using this to replace $\nabla f_3^{x,y,y'}$ in (9), we can write, $\forall x \in \mathcal{X}, y, y' \in \mathcal{Y}$:

$$\nabla f_1^{x,y,y'} (\phi(z_1) - \phi(z_3)) + \nabla f_2^{x,y,y'} (\phi(z_2) - \phi(z_3)) = 0 \tag{10}$$

(and two similar equations, replacing $\nabla f_1^{x,y,y'}$ and $\nabla f_2^{x,y,y'}$, respectively).

We define a function $\text{type}_f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{T}$, which classifies (x, y, y') into one of 5 possible *types* in the set $\mathbb{T} = \{\mathbb{T}_1, \mathbb{T}_{2:i}, \mathbb{T}_{2:2}, \mathbb{T}_{2:3}, \mathbb{T}_3\}$, depending on for which i , $\nabla f_i^{x,y,y'} = 0$:

1. $\text{type}_f(x, y, y') = \mathbb{T}_1$ if $\nabla f_1^{x,y,y'} = \nabla f_2^{x,y,y'} = \nabla f_3^{x,y,y'} = 0$.
2. $\text{type}_f(x, y, y') = \mathbb{T}_{2:i}$ if $\nabla f_i^{x,y,y'} = 0$ and $\forall j \in [3] \setminus \{i\}, \nabla f_j^{x,y,y'} \neq 0$.
3. $\text{type}_f(x, y, y') = \mathbb{T}_3$ if $\forall i \in [3], \nabla f_i^{x,y,y'} \neq 0$.

Note that since $\sum_{i=1}^3 \nabla f_i^{x,y,y'} = 0$, it cannot be the case that exactly one of $\nabla f_i^{x,y,y'} \neq 0$.

We define the type of f itself as the set:

$$T(f) := \{\tau \in \mathbb{T} : \exists (x, y, y') \text{ s.t. } \text{type}_f(x, y, y') = \tau\}. \tag{11}$$

We prove several claims regarding $T(f)$ before showing that \hat{f}_i and f_i are simple. In proving these claims, we shall use the fact that f is simple, is in normal-form, and $|\mathcal{X}|, |\mathcal{Y}| > 1$.

Claim 2. $\mathbb{T}_3 \notin T(f)$.

Proof. For the sake of contradiction, suppose there exists (x, y, y') such that $\text{type}_f(x, y, y') = \mathbb{T}_3$. Consider any $x' \in \mathcal{X} \setminus \{x\}$. From (10), we have

$$\begin{aligned} \nabla f_1^{x,y,y'} (\phi(z_1) - \phi(z_3)) + \nabla f_2^{x,y,y'} (\phi(z_2) - \phi(z_3)) &= 0 \\ \nabla f_1^{x',y,y'} (\phi(z_1) - \phi(z_3)) + \nabla f_2^{x',y,y'} (\phi(z_2) - \phi(z_3)) &= 0 \end{aligned}$$

We shall show that $\nabla f_1^{x,y,y'} \nabla f_2^{x',y,y'} \neq \nabla f_2^{x,y,y'} \nabla f_1^{x',y,y'}$. Then, the above system can be uniquely solved for $\phi(z_1) - \phi(z_3)$ and $\phi(z_2) - \phi(z_3)$, to yield 0 as the solution in each case. That is, $\phi(z_1) = \phi(z_2) = \phi(z_3)$, contradicting (7).

To complete the proof, we argue that $\nabla f_1^{x,y,y'} \nabla f_2^{x',y,y'} \neq \nabla f_2^{x,y,y'} \nabla f_1^{x',y,y'}$. Suppose not. Then, $\frac{\nabla f_1^{x',y,y'}}{\nabla f_1^{x,y,y'}} = \frac{\nabla f_2^{x',y,y'}}{\nabla f_2^{x,y,y'}} = \theta$, say (the denominators being non-zero, since $\text{type}_f(x, y, y') = T_3$). Then, $\frac{\nabla f_3^{x',y,y'}}{\nabla f_3^{x,y,y'}} = \frac{-\nabla f_1^{x',y,y'} - \nabla f_2^{x',y,y'}}{-\nabla f_1^{x,y,y'} - \nabla f_2^{x,y,y'}} = \theta$. Invoking the simplicity of f , we get that $\forall j \in [3], \frac{\rho(x, z_j)}{\rho(x', z_j)} = \theta$. However, since for any y we have $\sum_j \rho(x, z_j)\sigma(y, z_j) = \sum_j \rho(x', z_j)\sigma(y, z_j)$, we get $\theta = 1$. Then, $x \equiv x'$, contradicting the normal form of f . This completes the proof. \square

Claim 3. *There can be at most one $i \in [3]$ such that $T_{2:i} \in T(f)$.*

Proof. For the sake of contradiction, suppose $\text{type}_f(x, y, y') = T_{2:j}$, and $\text{type}_f(\tilde{x}, \tilde{y}, \tilde{y}') = T_{2:k}$, for $j \neq k$. We consider the case $j = 1, k = 2$, as the other cases are symmetric. Now, $\text{type}_f(x, y, y') = T_{2:1}$, implies that $\nabla f_i^{x,y,y'} = 0$ only for $i = 1$ and hence, by (10), we have $\nabla f_2^{x,y,y'}(\phi(z_2) - \phi(z_3)) = 0$, and so $\phi(z_2) = \phi(z_3)$. Similarly, $\text{type}_f(\tilde{x}, \tilde{y}, \tilde{y}') = T_{2:2}$ implies that $\phi(z_1) = \phi(z_3)$. Thus we have $\phi(z_1) = \phi(z_2) = \phi(z_3)$, contradicting (7). \square

Claim 4. *If from some $x \in \mathcal{X}$ and distinct $y, y' \in \mathcal{Y}$, $\text{type}_f(x, y, y') = T_1$, then there exists $x' \in \mathcal{X}$ such that $\text{type}_f(x', y, y') \neq T_1$.*

Proof. Since $\text{type}_f(x, y, y') = T_1$, we have that for all $j \in [3], f(x, y, z_j) = f(x, y', z_j)$. Since f is in normal form, $y \neq y'$, and hence there exists $x' \in \mathcal{X}$ such that for some $j \in [3], f(x', y, z_j) \neq f(x', y', z_j)$. Hence $\text{type}_f(x', y, y') \neq T_1$. \square

Claim 5. *Suppose $T(f) = \{T_1, T_{2:i}\}$ for some $i \in [3]$. Then, if for some $x \in \mathcal{X}$, and distinct $y, y' \in \mathcal{Y}$, $\text{type}_f(x, y, y') = T_1$, then for all $\tilde{y} \in \mathcal{Y}$, $f(x, \tilde{y}, z_i) = 1$.*

Proof. By Claim 4, we have x' such that $\text{type}_f(x', y, y') \neq T_1$; since $T(f) = \{T_1, T_{2:i}\}$, we have $\text{type}_f(x', y, y') = T_{2:i}$. Then, for both values of $j \neq i$ in $[3]$, we have $f(x', y, z_j) \neq f(x', y', z_j)$.

Invoking the simplicity of f , we have that for $j \neq i, \rho(x', z_j)(\sigma(y, z_j) - \sigma(y', z_j)) \neq 0$, but $\rho(x, z_j)(\sigma(y, z_j) - \sigma(y', z_j)) = 0$. Hence $\rho(x, z_j) = 0$ for $j \neq i$. That is, for all $\tilde{y} \in \mathcal{Y}, f(x, \tilde{y}, z_j) = 0$ for $j \neq i$ and hence $f(x, \tilde{y}, z_i) = 1$. \square

From the above claims, we have two possibilities for $T(f)$: either $\{T_{2:i}\}$ or $\{T_1, T_{2:i}\}$ for some $i \in [3]$. Then we shall prove that \hat{f}_i and f_i are simple. Note that both these functions are binary functions. For our case where f has a minimal-round protocol with Alice sending the first bit, and with $|\mathcal{X}|, |\mathcal{Y}| > 1$, we show that this means that \hat{f}_i is a function of only Alice's input, and f_i is a function of only Bob's input.

Claim 6. *If $T(f) \subseteq \{T_1, T_{2:i}\}$, then \hat{f}_i is simple.*

Proof. We have that for all (x, y, y') , $\nabla f_i^{x,y,y'} = 0$, i.e., $\rho(x, z_i)(\sigma(y, z_i) - \sigma(y', z_i)) = 0$. Now, if $\rho(x, z_i) = 0$ for all x , then \hat{f}_i is a constant function with $\hat{f}_i(x, y, z_i) = 0$ and $\hat{f}_i(x, y, z_*) = 1$. Otherwise, there is some $x \in \mathcal{X}$ such that $\rho(x, z_i) \neq 0$. Hence for all y, y' we have $\sigma(y, z_i) = \sigma(y', z_i) = s$, say. Then $\hat{f}_i(x, y, z_i) = \rho(x, z_i) \cdot s$, which is independent of y . Thus, in either case, \hat{f}_i is simple. \square

Claim 7. *If $T(f) \subseteq \{\mathsf{T}_1, \mathsf{T}_{2:i}\}$, then $f_i(x, y, z)$ is simple.*

Proof. Recall that f_i has input spaces \mathcal{X}_i and \mathcal{Y} , where $\mathcal{X}_i = \{x \in \mathcal{X} : \exists y \in \mathcal{Y} \text{ s.t. } f(x, y, z_i) < 1\}$. We shall prove that for all $x, x' \in \mathcal{X}_i$ and $y \in \mathcal{Y}$, $f_i(x, y, z_j) = f_i(x', y, z_j)$ for $j \neq i$.

By Claim 5, for all $x \in \mathcal{X}_i$ and distinct $y, y' \in \mathcal{Y}$, $\text{type}_f(x, y, y') \neq \mathsf{T}_1$, and hence $\text{type}_f(x, y, y') = \mathsf{T}_{2:i}$. Therefore, for all $x \in \mathcal{X}_i$ and $j \neq i$, $\rho(x, z_j) \neq 0$.

Let $\{j, \bar{j}\} = [3] \setminus \{i\}$. Now,

$$f_i(x, y, z_j) = \frac{f(x, y, z_j)}{1 - f(x, y, z_i)} = \frac{\rho(x, z_j)\sigma(y, z_j)}{\rho(x, z_j)\sigma(y, z_j) + \rho(x, z_{\bar{j}})\sigma(y, z_{\bar{j}})} = \frac{\sigma(y, z_j)}{\sigma(y, z_j) + \gamma(x)\sigma(y, z_{\bar{j}})},$$

where $\gamma(x) := \frac{\rho(x, z_{\bar{j}})}{\rho(x, z_j)}$. Here we have used the fact that $\rho(x, z_j) \neq 0$ for all $x \in \mathcal{X}_i$. Thus to prove the claim, it is enough to show that $\gamma(x) = \gamma(x')$ for all $x, x' \in \mathcal{X}_i$.

For any $x \in \mathcal{X}_i$, as mentioned above, for any distinct $y, y' \in \mathcal{Y}$, we have $\text{type}_f(x, y, y') = \mathsf{T}_{2:i}$, which implies that $\nabla f_j^{x,y,y'} + \nabla f_{\bar{j}}^{x,y,y'} = -\nabla f_i^{x,y,y'} = 0$. That is,

$$\rho(x, z_j)(\sigma(y, z_j) - \sigma(y', z_j)) + \rho(x, z_{\bar{j}})(\sigma(y, z_{\bar{j}}) - \sigma(y', z_{\bar{j}})) = 0.$$

Writing the above equation for any $x, x' \in \mathcal{X}_i$, if $\rho(x, z_j)\rho(x', z_{\bar{j}}) \neq \rho(x, z_{\bar{j}})\rho(x', z_j)$, we will be able to solve that $(\sigma(y, z_j) - \sigma(y', z_j)) = (\sigma(y, z_{\bar{j}}) - \sigma(y', z_{\bar{j}})) = 0$. But this implies that $\nabla f_j^{x,y,y'} = \nabla f_{\bar{j}}^{x,y,y'} = 0$ (for any $x \in \mathcal{X}_i$), contradicting the fact that $\text{type}_f(x, y, y') \neq \mathsf{T}_1$ for all $x \in \mathcal{X}_i$ and distinct $y, y' \in \mathcal{Y}$. Thus we should have $\rho(x, z_j)\rho(x', z_{\bar{j}}) = \rho(x, z_{\bar{j}})\rho(x', z_j)$, or (dividing by $\rho(x, z_j) \cdot \rho(x', z_j) \neq 0$), $\gamma(x) = \gamma(x')$. \square

Taken together, the above claims prove the only if (\Rightarrow) part of Theorem 5.

\Leftarrow : Let $i \in [3]$ be such that \hat{f}_i and f_i are simple, and therefore, securely computable. A 2-round secure protocol for f is as follows. If both \hat{f}_i and f_i are independent of x or y (or both), then clearly f is securely computable by a protocol in which one party computes the output and sends it to the other party. The only interesting case is when \hat{f}_i is independent of y and f_i is independent of x (the other case when \hat{f}_i is independent of x and f_i is independent of y being symmetric): Alice picks $j \in \{i, *\}$ with probability $\hat{f}_i(x, y_1, z_j)$ and sends j to Bob. If $j = i$, both Alice and Bob output z_i with probability 1; otherwise, Bob picks $k \in [3] \setminus \{i\}$ with probability $f_i(x_1, y, z_k)$ and sends k to Alice. Now both Alice and Bob output z_k with probability 1. It is clear from the definitions of \hat{f}_i and f_i that the output from this protocol is correctly distributed. It is easy to see that this is a unique-transcript protocol, and therefore, is perfectly private. \square

4 Functions with 2-Round Secure Protocols

We prove the following theorem which, when applied to the kernel of a given function, implies Theorem 2.

Theorem 6. *A simple symmetric function $p_{Z|XY}$ with X, Y, Z over alphabets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, has a two-round protocol with Alice making the first move iff there exists a surjective map $g : \mathcal{Z} \rightarrow \mathcal{W}$ to some set \mathcal{W} and probability distributions $p_{W|X}$ and $p_{Z|WY}$ where W is over the alphabet \mathcal{W} , such that $p_{Z|WY}(z|w, y) = 0$ if $w \neq g(z)$, and for all $x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$,*

$$p_{Z|XY}(z|x, y) = p_{W|X}(g(z)|x) \cdot p_{Z|WY}(z|g(z), y).$$

In that case, $p_{Z|XY}$ has a unique-transcript secure protocol in which Alice sends w sampled according to $p_{W|X= x}$ and Bob sends back z according to $p_{Z|W= w, Y= y}$.

Proof. It is easy to see that if $g, p_{W|X}, p_{Z|WY}$ as in the statement exist, then the protocol described is indeed a unique-transcript protocol for $p_{Z|XY}$: its output is distributed correctly, and since $p_{Z|WY}(z|w, y) = 0$ if $w \neq g(z)$, the only transcript resulting in the output z is of the form $(g(z), z)$. A unique-transcript protocol is always a secure protocol since the transcript can be simulated from the output by a (deterministic) simulator.

We prove the other direction below. Suppose we are given a two-round protocol Π_0 for $p_{Z|XY}$, with the two messages denoted by a and b . Then, we can construct a secure protocol Π in which Bob computes the second message b as before, but sends out $z = \text{out}_{\Pi_0}(a, b)$, and both Alice and Bob output z : clearly Π has the same output as Π_0 and is also secure since the transcript of Π can be simulated from a (simulated) transcript of Π_0 by applying a deterministic function to it.

Π is defined by probability distributions $\Pr_{\Pi}[a|x]$ and $\Pr_{\Pi}[z|a, y]$. For convenience, we define $\alpha(a, x) := \Pr_{\Pi}[a|x]$ and $\beta(a, z, y) := \Pr_{\Pi}[z|a, y]$. Also, since $p_{Z|XY}$ is simple, let us write $p_{Z|XY}(z|x, y) = \rho(x, z) \cdot \sigma(y, z)$.

Before proceeding further, note that for a transcript $m = (a, z)$, from (2), we have

$$\Pr_{\Pi}[m|x, y] = \mu(m)\rho(x, z)\sigma(y, z) = \alpha(a, x)\beta(a, z, y).$$

If $\Pr_{\Pi}[m|x, y] > 0$ for some x, y , then by considering the above equality for (x, y) as well as (x', y) , and dividing the latter by the former (which is non-zero), we get that for all $x' \in \mathcal{X}$,

$$\frac{\rho(x', z)}{\rho(x, z)} = \frac{\alpha(a, x')}{\alpha(a, x)}. \tag{12}$$

We define an equivalence relation \equiv over \mathcal{Z} as follows:

$$z_1 \equiv z_2 \text{ if } \exists c > 0, \forall x \in \mathcal{X}, \rho(x, z_1) = c\rho(x, z_2).$$

We let \mathcal{W} be the set of equivalence classes of \equiv , and define $g : \mathcal{Z} \rightarrow \mathcal{W}$ which maps z to its equivalence class. Thus, $z_1 \equiv z_2$ iff $g(z_1) = g(z_2)$. We also define a

function h that maps the first message a in a transcript to an element in \mathcal{W} , as follows:

$$h(a) = g(z) \text{ if } \exists x, y \text{ s.t. } \Pr_{\Pi}[a, z|x, y] > 0.$$

For h to be well-defined, we need that each a has a unique value $h(a)$ that satisfies the above condition. Suppose $z_1, z_2 \in \mathcal{Z}$ are such that $\Pr_{\Pi}[a, z_1|x_1, y_1] > 0$ and $\Pr_{\Pi}[a, z_2|x_2, y_2] > 0$ (from some $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$). By applying (12) to these, we get that for all $x' \in \mathcal{X}$

$$\frac{\rho(x', z_1)}{\rho(x_1, z_1)} = \frac{\alpha(a, x')}{\alpha(a, x_1)} \quad \text{and} \quad \frac{\rho(x', z_2)}{\rho(x_2, z_2)} = \frac{\alpha(a, x')}{\alpha(a, x_2)}.$$

Hence $\forall x' \in \mathcal{X}, \rho(x', z_2) = c\rho(x', z_1)$, where $c := \frac{\rho(x_2, z_2)\alpha(a, x_1)}{\alpha(a, x_2)\rho(x_1, z_1)}$. All the factors in c are positive (as they appear in $\Pr_{\Pi}[a, z_1|x_1, y_1] \cdot \Pr_{\Pi}[a, z_2|x_2, y_2] > 0$), and also independent of x' . Thus $z_1 \equiv z_2$ and $g(z_1) = g(z_2)$, making $h(a)$ well defined.

$p_{W|X}$ is defined as follows: given x , sample a as in Π , and output $w = h(a)$. That is, $p_{W|X}(w|x) = \sum_{a:h(a)=w} \alpha(a, x)$. Finally, we define $p_{Z|WY}$ as follows: given w , we argue that we can reverse sample a without access to x (so that $w = h(a)$), and then use the protocol Π to sample z from (a, y) . That is, we define a distribution over a given w , by the probability

$$\eta(a, w) := \begin{cases} \frac{\alpha(a, x)}{\sum_{a':h(a')=w} \alpha(a', x)} & \text{if } h(a) = w \\ 0 & \text{otherwise,} \end{cases}$$

where *any* $x \in \mathcal{X}$ such that $\sum_{a':h(a')=w} \alpha(a', x) > 0$ is used. This is well defined because, by (12), switching from x to x' amounts to multiplying both the numerator and the denominator by the same factor (namely, $\frac{\rho(x', z)}{\rho(x, z)}$ for any $z \in g^{-1}(w)$). Then, $p_{Z|WY}(z|w, y) = \sum_a \eta(a, w)\beta(a, z, y)$. We verify that, when $w = g(z)$,

$$\begin{aligned} p_{W|X}(w|x) \cdot p_{Z|WY}(z|w, y) &= \left(\sum_{a:h(a)=w} \alpha(a, x) \right) \left(\sum_a \eta(a, w)\beta(a, z, y) \right) \\ &= \sum_a \alpha(a, x)\beta(a, z, y) = \Pr_{\Pi}[z|x, y] = p_{Z|XY}(z|x, y). \end{aligned}$$

□

5 Complexity of Randomized Functions

We point out a couple of complexity aspects in which randomized functions differ from deterministic functions. This also points to the difficulty in characterization of securely computable functions in the case of randomized functions.

5.1 Smaller Simple Functions Which Are Not Securely Computable

It is well-known that simple functions are not all securely computable, even for deterministic functions, with a first example given by Beaver [Bea89], with an output alphabet of size 5. This turns out to be the smallest output alphabet for a simple deterministic function that is not securely realizable.

But for randomized functions, we see a higher level of complexity arising even with an output size of 3. In Fig. 1 we show an example of a simple function with ternary output alphabet that is not securely computable, i.e., it does not satisfy the characterization given in Theorem 5.

		y_1			y_2		
		z_1	z_2	z_3	z_1	z_2	z_3
x_1	z_1	2/9			5/18		
	z_2	4/9			2/9		
	z_3	1/3			1/2		
x_2	z_1	1/3			5/12		
	z_2	5/12			5/24		
	z_3	1/4			3/8		

Fig. 1. This function is not securely computable as it does not satisfy the condition from

Theorem 5. However, it is simple, with the functions ρ and σ given by

	z_1	z_2	z_3
x_1	1/3	2/3	1
x_2	1/2	5/8	3/4

and $\begin{matrix} & z_1 & z_2 & z_3 \\ y_1 & 2/3 & 2/3 & 1/3 \\ y_2 & 5/6 & 1/3 & 1/2 \end{matrix}$.

5.2 Limits of Unique-Transcript Protocols

It follows from Sect. 3 that all securely computable randomized functions with a ternary output kernel can be computed using unique-transcript protocols. Also, it follows from Sect. 4 that all randomized functions securely computable by two-round protocols are in fact securely computable using two-round unique-transcript protocols. Also, the characterization by Kushilevitz [Kus89] showed that all securely computable *deterministic* functions have unique-transcript secure protocols. Thus one may reasonably suspect that all securely computable functions have unique-transcript secure protocols.

However, we show that in some sense, the above results give the limits of unique-transcript protocols: If we go just beyond the above conditions – namely, ternary output, 2-round computable, deterministic – then we can indeed find securely computable functions that do not have any unique-transcript secure protocol. In Appendix B, we demonstrate a simple randomized function with an output alphabet of size 4, securely computable by a 3-round protocol, such that it has no unique-transcript secure protocols.

Acknowledgments. Deepesh Data’s research was supported in part by a Microsoft Research India Ph.D. Fellowship. Manoj Prabhakaran wishes to thank Hemanta Maji and Amit Sahai for an earlier collaboration on the same problem. The observation in Sect. 5.2 was made (using a different example) during that work as well.

A Security for Simple Symmetric Functions

In this section we prove Lemma 1.

Proof of Lemma 1. Firstly, by perfect correctness, for any x, y , we require that with probability one, $\text{out}_\Pi^A(m, x) = \text{out}_\Pi^B(m, y)$, where m is produced by Π on input (x, y) . Suppose there is an $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$ such that $\Pr_\Pi[m|x_0, y_0] > 0$.

Since we can write $\Pr_\Pi[m|x, y] = \alpha(m, x)\beta(m, y)$, we have $\alpha(m, x_0)\beta(m, y_0) > 0$. Also, for all x such that $\alpha(m, x) > 0$, we have a positive probability of Π producing m on input (x, y_0) and hence we must have $\text{out}_\Pi^A(m, x) = \text{out}_\Pi^B(m, y_0)$ with probability 1 (which requires them to be deterministic). Similarly, for all y such that $\beta(m, y) > 0$, we have $\text{out}_\Pi^B(m, y) = \text{out}_\Pi^A(m, x_0)$ with probability 1. Letting $\text{out}_\Pi(m) := \text{out}_\Pi^A(m, x_0) = \text{out}_\Pi^B(m, y_0)$ (which must be deterministic), we have that $\text{out}_\Pi^A(m, x) = \text{out}_\Pi^B(m, y) = \text{out}_\Pi(m)$ if $\Pr_\Pi[m|x, y] > 0$.

Now we prove the second part. Note that since Π computes $p_{Z|XY}$, we have that for all x, y, z , $\Pr_\Pi[z|x, y] = p_{Z|XY}(z|x, y)$. Consider any x, y, z such that $p_{Z|XY}(z|x, y) > 0$. For all m such that $\text{out}_\Pi(m) \neq z$, we can set $S(m, z) = 0$. So, suppose $\text{out}_\Pi(m) = z$. Then

$$\Pr_\Pi[m|x, y, z] = \frac{\Pr_\Pi[m, z|x, y]}{\Pr_\Pi[z|x, y]} = \frac{\Pr_\Pi[m|x, y]}{p_{Z|XY}(z|x, y)} = \frac{\alpha(m, x)\beta(m, y)}{\rho(z, x)\sigma(z, y)},$$

where we wrote $\Pr_\Pi[m, z|x, y] = \Pr_\Pi[m|x, y]$ (since $z = \text{out}_\Pi(m)$), $\Pr_\Pi[m|x, y] = \alpha(m, x)\beta(m, y)$ (since Π is a protocol) and $p_{Z|XY}(z|x, y) = \rho(z, x)\sigma(z, y)$ (since $p_{Z|XY}$ is a simple function). Using the security guarantee for a symmetric function (Definition 1, with $z_A = z_B = z$), we get

$$S_1(m, x, z) = S_2(m, y, z) = \frac{\alpha(m, x)}{\rho(z, x)} \cdot \frac{\beta(m, y)}{\sigma(z, y)}.$$

Now, fixing (m, z) as above, consider all (x, y) such that $p_{Z|XY}(z|x, y) > 0$. If the above expression is 0 for all choices of (x, y) , then we can simply set $S(m, z) = 0$. Otherwise, there is some x such that $S_1(m, x, z) \neq 0$. Then, considering the above expression for that x , we get that $\frac{\beta(m, y)}{\sigma(z, y)}$ equals a quantity that is independent of y (and hence is a function of (m, z) alone). Similarly, by considering $S_2(m, y, z)$, we get that $\frac{\alpha(m, x)}{\rho(z, x)}$ is a function of (m, z) alone. Hence we have

$$S_1(m, x, z) = S_2(m, y, z) = S(m, z).$$

□

B Example for Sect. 5.2

Theorem 7. *There exists a randomized function that can be securely computed using a 3-round protocol, but cannot be securely computed using a unique-transcript protocol with any number of rounds.*

		y_1				y_2			
		z_1	z_2	z_3	z_4	z_1	z_2	z_3	z_4
x_1	z_1	2/9				5/18			
	z_2	31/108				31/216			
	z_3	17/108				17/216			
	z_4	1/3				1/2			
x_2	z_1	1/3				5/12			
	z_2	31/360				31/720			
	z_3	119/360				119/720			
	z_4	1/4				3/8			

Fig. 2. This simple randomized function $p_{Z|XY}$ is securely computable by a 3 round protocol (given in Fig. 3) that is not unique-transcript, but cannot be securely computed using any unique-transcript protocol (with any number of rounds).

Proof. Consider the simple randomized function $p_{Z|XY}$ given in Fig. 2. This can be securely computed, and a 3-round protocol for that is given in Fig. 3. Note that this protocol is not unique-transcript. First we show that the protocol given in Fig. 3 is secure, i.e., it is correct and perfectly private; and later we show that no unique-transcript protocol can securely compute this function with any number of rounds.

Correctness: It follows from the fact that for every x, y, z , $p_{Z|XY}(z|x, y)$ is equal to the sum of the probabilities on different paths leading to leaves labelled as z , where probability of a path is equal to the product of the probabilities (corresponding to the particular x and y) appearing on the edges along that path.

Privacy: Consider an arbitrary $k \in [4]$. We need to show that for any transcript m , $p(m|x_i, y_j, z_k)$ must be the same for every $i, j \in [2]$, $k \in [4]$. This trivially holds for every $z \in \mathcal{Z} \setminus \{z_2, z_3\}$, because there is a unique path from root to the leaf corresponding to z , which means that output being z itself determines the whole transcript. But for z_2 and z_3 there are two possible transcripts. Fix any $z \in \{z_2, z_3\}$, say, z_2 ; a similar argument holds for $z = z_3$ as well. There are two transcripts (m_{11}, m_{22}, m_{31}) and (m_{12}, m_{23}, m_{33}) for z_2 , implying two distinct paths. In order to show that the protocol is perfectly private, we need to show that $p(m_{11}, m_{22}, m_{31}|x_i, y_j, z_2)$ is the same for all $i, j \in [2]$. It can be easily verified that this is indeed the case, which implies that the protocol described in Fig. 3 is perfectly private.

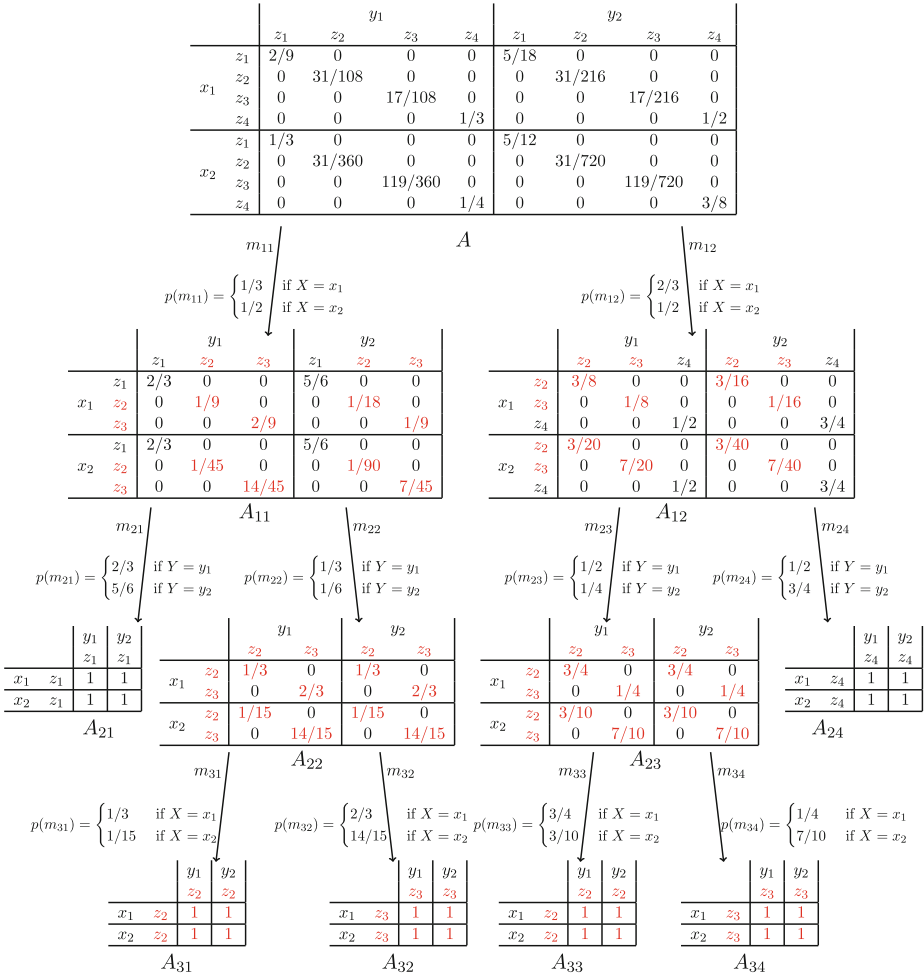


Fig. 3. This describes a 3-round protocol for securely computing $p_{Z|XY}$, denoted by matrix A , where the first message is sent by Alice. This protocol is not unique-transcript: both A_{11} and A_{12} have z_2, z_3 in common, and that is highlighted in red color. The meaning of the probabilities on the edges is as follows: If Alice’s input is x_1 , then she sends m_{11} as the first message with probability $1/3$ and m_{12} as the first message with probability $2/3$. If Alice’s input is x_2 , then she sends m_{11} as the first message with probability $1/2$ and m_{12} as the first message with probability $1/2$. If Alice sends m_{11} , then the problem reduces to securely computing A_{11} , and if Alice sends m_{12} , then the problem reduces to securely computing A_{12} . Suppose Alice reduces the problem to A_{11} . Now it is Bob’s turn to send a message. If Bob’s input is y_1 , he sends m_{21} with probability $2/3$ and m_{22} with probability $1/3$, and so on ... In the end, at the leaf nodes there is only one possible z_i to output, and they output that element with probability 1. (Color figure online)

Now we show that no *unique-transcript* protocol (with any number of rounds) can securely compute $A := p_{Z|XY}$. Note that in a unique-transcript protocol, during any round, the party who is sending a message makes a partition of the output alphabet, and sends the other party the part (i.e., the reduced output alphabet) in which the output should lie. We show below that neither Alice nor Bob can make a partition in the first round itself. This implies that no unique-transcript protocol exists for securely computing A with any number of rounds.

- **Alice cannot partition \mathcal{Z} :** Suppose, to the contrary, that Alice can partition \mathcal{Z} in the first round; and, assume, w.l.o.g., that Alice makes two parts $\mathcal{Z} = \mathcal{Z}_1 \uplus \mathcal{Z}_2$.

Let $m_i, i = 1, 2$ denote the message that Alice sends to Bob in order to restrict the output to $\mathcal{Z}_i, i = 1, 2$. This implies that $p_{M_1|XYZ}(m_1|x_i, y_j, z \in \mathcal{Z}_1) = 1$ and $p_{M_1|XYZ}(m_2|x_i, y_j, z \in \mathcal{Z}_2) = 1$, for every $i, j \in \{1, 2\}$. We show below that if Alice partitions $\mathcal{Z} = \mathcal{Z}_1 \uplus \mathcal{Z}_2$, then the following must hold for every $i \in \{1, 2\}$:

$$\sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_i, y_1) = \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_i, y_2), \tag{13}$$

$$\sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_i, y_1) = \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_i, y_2). \tag{14}$$

It can be easily verified that the matrix A does not satisfy the above two conditions for any non-trivial and disjoint $\mathcal{Z}_1, \mathcal{Z}_2$, which is a contradiction: since $\mathcal{Z} = \mathcal{Z}_1 \uplus \mathcal{Z}_2$ and $|\mathcal{Z}| = 4$, one of the following must hold: (i) either $|\mathcal{Z}_1| = 1$ or $|\mathcal{Z}_2| = 1$, or (ii) either $|\mathcal{Z}_1| = 2$ or $|\mathcal{Z}_2| = 2$. This verification can be done easily even exhaustively. We show (13) and (14) below. In the following, i belongs to $\{1, 2\}$.

$$\begin{aligned} p_{M_1|XY}(m_1|x_i, y_1) &= \sum_{z \in \mathcal{Z}_1} p_{M_1Z|XY}(m_1, z|x_i, y_1) + \sum_{z \in \mathcal{Z}_2} p_{M_1Z|XY}(m_1, z|x_i, y_1) \\ &= \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_i, y_1) \underbrace{p_{M_1|XYZ}(m_1|x_i, y_1, z)}_{= 1} \\ &\quad + \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_i, y_1) \underbrace{p_{M_1|XYZ}(m_1|x_i, y_1, z)}_{= 0} \\ &= \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_i, y_1) \end{aligned} \tag{15}$$

Similarly we can show the following:

$$p_{M_1|XY}(m_1|x_i, y_2) = \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_i, y_2), \tag{16}$$

$$p_{M_1|XY}(m_2|x_i, y_1) = \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_i, y_1), \tag{17}$$

$$p_{M_1|XY}(m_2|x_i, y_2) = \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_i, y_2). \quad (18)$$

Since Alice sends the first message, which means that the Markov chain $M_1 - X - Y$ holds. This implies that $p_{M_1|XY}(m_j|x_i) = p_{M_1|XY}(m_j|x_i, y_1) = p_{M_1|XY}(m_j|x_i, y_2)$ for every $j \in \{1, 2\}$. Now comparing (15) and (16) gives (13), and (17) and (18) gives (14).

- **Bob cannot partition \mathcal{Z} :** Switching the roles of Alice and Bob with each other in the above argument and using the fact that for every partition $\mathcal{Z} = \mathcal{Z}_1 \uplus \mathcal{Z}_2$, the matrix A does not satisfy the following two conditions, we can prove that Bob also cannot partition \mathcal{Z} . In the following, i belongs to $\{1, 2\}$.

$$\begin{aligned} \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_1, y_i) &= \sum_{z \in \mathcal{Z}_1} p_{Z|XY}(z|x_2, y_i), \\ \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_1, y_i) &= \sum_{z \in \mathcal{Z}_2} p_{Z|XY}(z|x_2, y_i). \end{aligned}$$

This completes the proof of Theorem 7. □

References

- [Bea89] Beaver, D.: Perfect privacy for two-party protocols. In: Feigenbaum, J., Merritt, M. (eds.) Proceedings of DIMACS Workshop on Distributed Computing and Cryptography, vol. 2, pp. 65–77. American Mathematical Society (1989)
- [Dat16] Data, D.: Secure computation of randomized functions. In: IEEE International Symposium on Information Theory. ISIT 2016, Barcelona, Spain, 10–15 July 2016, pp. 3053–3057 (2016)
- [Kil88] Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31. ACM (1988)
- [Kil00] Kilian, J.: More general completeness theorems for secure two-party computation. In: Proceedings of 32th STOC, pp. 316–324. ACM (2000)
- [KMPS14] Kraschewski, D., Maji, H.K., Prabhakaran, M., Sahai, A.: A full characterization of completeness for two-party randomized function evaluation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 659–676. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_36
- [KMQR09] Künzler, R., Müller-Quade, J., Raub, D.: Secure computability of functions in the IT setting with dishonest majority and applications to long-term security. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 238–255. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_15
- [Kus89] Kushilevitz, E.: Privacy and communication complexity. In: FOCS, pp. 416–421. IEEE (1989)
- [MPR09] Maji, H.K., Prabhakaran, M., Rosulek, M.: Complexity of multi-party computation problems: the case of 2-party symmetric secure function evaluation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 256–273. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_16

- [MPR12] Maji, H.K., Prabhakaran, M., Rosulek, M.: A unified characterization of completeness and triviality for secure function evaluation. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 40–59. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34931-7_4
- [MPR13] Maji, H.K., Prabhakaran, M., Rosulek, M.: Complexity of multi-party computation functionalities. In: Prabhakaran, M., Sahai, A. (eds.) Secure Multi-Party Computation. Cryptology and Information Security Series, vol. 10, pp. 249–283. IOS Press (2013)
- [Rei09] Reingold, O. (ed.): TCC 2009. LNCS, vol. 5444. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-00457-5>



On the Message Complexity of Secure Multiparty Computation

Yuval Ishai¹(✉), Manika Mittal^{2,3}, and Rafail Ostrovsky³

¹ Technion, Haifa, Israel
yuvali@cs.technion.ac.il

² Yahoo!, Sunnyvale, USA
manikamittal22@gmail.com

³ UCLA, Los Angeles, USA
rafail@cs.ucla.edu

Abstract. We study the minimal number of point-to-point messages required for general secure multiparty computation (MPC) in the setting of computational security against semi-honest, static adversaries who may corrupt an arbitrary number of parties.

We show that for functionalities that take inputs from n parties and deliver outputs to k parties, $2n + k - 3$ messages are necessary and sufficient. The negative result holds even when given access to an arbitrary correlated randomness setup. The positive result can be based on any 2-round MPC protocol (which can in turn be based on 2-message oblivious transfer), or on a one-way function given a correlated randomness setup.

1 Introduction

Since the seminal works from the 1980s that established the feasibility of secure multiparty computation (MPC) [3, 9, 19, 24], there has been a large body of work on different *efficiency* measures of MPC protocols. In particular, a lot of research efforts were aimed at characterizing the minimal communication complexity, round complexity, computational complexity, and randomness complexity of MPC protocols.

In the present work we study the *message complexity* of MPC protocols, namely the number of messages that the parties need to communicate to each other over point-to-point channels. While there have been a few prior works studying the message complexity of MPC in different settings (see Sect. 1.2 below), this complexity measure received relatively little attention. The goal of minimizing the message complexity of protocols is motivated by scenarios in which sending or receiving a message has a high cost, which is not very sensitive to the size of the message. For instance, this is the case when using a traditional postal system for message delivery (say, shipping optical media from one party to another), or when establishing a communication channel between pairs of parties is expensive due to limited connectivity.

The main focus of our work is on the standard model of *computationally* secure MPC in the presence of a *static* (non-adaptive), *semi-honest* (passive) adversary, who may corrupt an *arbitrary subset* of the parties. In this model, we ask the following question:

How many messages are needed for securely computing functions that take inputs from n parties and deliver outputs to k of these parties?

For simplicity, we consider the above question in the setting of fixed, or “oblivious,” interaction patterns, a commonly used assumption in the MPC literature (see, e.g., [11, 20]). In this setting, it is assumed that the protocol specifies a-priori the sender-receiver pairs of the messages sent in each round.¹

1.1 Our Contribution

Our main result is a sharp answer to the above question: we show that in the setting discussed above, $2n + k - 3$ messages are necessary and sufficient.

The negative result holds even when the parties can communicate over *secure* point-to-point channels or, more generally, even when allowing an arbitrary input-independent correlated randomness setup. This result builds (non-trivially) on the general characterization of the power MPC with general interaction patterns from the recent work of Halevi et al. [20].

The positive result can be based on any 2-round MPC protocol, applying a natural greedy message forwarding strategy to emulate the quadratic number of messages of such protocols with an optimal number of messages. Using recent constructions of 2-round MPC protocols, this approach can be instantiated in the plain model, public point-to-point channels, under the (minimal) assumption that a 2-message semi-honest oblivious transfer protocol exists [4, 17]. (Alternative constructions with incomparable efficiency features can be based on the LWE assumption [23] or even the DDH assumption given a PKI setup [6]). Given a general correlated randomness setup, the positive result can be based on any one-way function, or even provide unconditional information theoretic security when considering low-complexity functions such as NC^1 functions.

1.2 Related Work

As mentioned above, Halevi et al. [20] consider the question of MPC with general interaction patterns, giving a full characterization for the “best possible security” of an MPC protocol that uses a given interaction pattern with a general

¹ Message complexity is more subtle when allowing dynamic interaction patterns, since not receiving a message also conveys information; see e.g. [13] for discussion. Our positive results do not require this relaxation. Moreover, our negative result can be extended to capture dynamic interactions, by exploiting the fact that the adversary can “guess” the identity of a party that sends a constant number of messages with high success probability and corrupt all of the other parties.

correlated randomness setup. Our negative result builds on their general characterization, focusing on the case where the “best possible security” coincides with the standard notion of security. The positive results in [20] consider a more general setting that (inevitably) requires the use of indistinguishability obfuscation and a correlated randomness setup. In contrast, our positive results rely on weaker assumptions and apply also to the plain model.

The message complexity of MPC protocols has been explicitly considered in several previous works, but the model of MPC considered in these works is quite different from ours. In particular, the message complexity in the *information-theoretic* setting with a *bounded fraction* of corrupted parties has been studied in [5, 7, 11, 13, 14]. Our focus on computational security (or alternatively, allowing a correlated randomness setup) allows us to circumvent previous lower bounds that apply to the information-theoretic setting. In particular, our positive results circumvent the quadratic message lower bound from [11]. On the other hand, considering an adversary that can corrupt an arbitrary number of parties rules out MPC protocols that achieve sublinear message complexity in the number of parties by assigning the computation to a small random subset of parties (see, e.g., [7, 12, 16]).

Organization. Following some preliminaries (Sect. 2), we present our negative result in Sect. 3 and our positive results in Sect. 4. In Appendix A we include a standard definition of MPC for self-containment.

2 Preliminaries

By default, we consider an MPC protocol Π for an n -party functionality f to provide *computational* security against a *semi-honest* adversary that may *statically* (non-adaptively) corrupt an arbitrary subset of the parties and eavesdrop on all communication channels. That is, the communication takes place over public point-to-point channels.

We also consider MPC with *correlated randomness setup*, where the parties are given access to a trusted source of (input-independent) correlated randomness. Note that correlated randomness setup trivially allows secure point-to-point communication over public communication channels. Thus, since our negative result applies also to this model, it applies in particular for protocols over secure point-to-point channels.

As is typically the case for security against semi-honest adversaries, our results are quite insensitive to the details of the model beyond those mentioned above. We refer to reader to Appendix A or to [18] for a standard formal treatment of MPC in this model.

3 The Lower Bound

In this section, we prove our main lower bound: in any n -party MPC protocol for computing a function with $k \geq 1$ outputs, the number of point-to-point messages

is at least $2n + k - 3$. This lower bound holds even in the setting of security against semi-honest adversaries and even when the parties are given access to an arbitrary trusted source of (input-independent) correlated randomness.

The work of Halevi et al. [20] gives a general characterization for the “best possible security” of an MPC protocol with general correlated randomness setup and a given interaction pattern. The characterization in [20] is mainly intended for the case of limited interactions that warrant a relaxed notion of MPC security, and is only formulated for the case of protocols that deliver output to a single party. Here we give a simple self-contained treatment for the case of standard MPC security with an arbitrary number of outputs.

We start by defining a simplified notion of an interaction pattern, which specifies an ordered sequence of pairs of parties that represent the sender and receiver of each message. Note that we implicitly assume here that the protocol sends only a single message in each round. However, any protocol can be trivially converted into this form by splitting the messages sent in each round into multiple rounds in an arbitrary order.

Definition 3.1 (Interaction pattern). *An n -party interaction pattern is specified a sequence of pairs $M \in ([n] \times [n])^*$. The length of M is the number of pairs in the sequence. We say that an n -party MPC protocol Π complies with an n -party interaction pattern $M = ((a_1, b_1), \dots, (a_m, b_m))$ if for every $1 \leq i \leq m$, the communication in Round i of Π involves only a single message, sent from party P_{a_i} to party P_{b_i} .*

It is convenient to represent an interaction pattern M by a directed (multi-) graph, whose nodes represent parties and whose edges represent messages sent over point-to-point channels. Each edge is labeled by its index in M . A *trail* in the graph is a (non-simple, directed) path that respects the order of edges and can visit the same node more than once. We formalize this below.

Definition 3.2 (Interaction graph). *Let $M = ((a_1, b_1), \dots, (a_m, b_m))$ be an n -party interaction pattern. We let G_M denote the labeled directed multi-graph whose node set is $[n]$ and whose edges form the sequence (e_1, \dots, e_m) where $e_i = (a_i, b_i)$. (Each edge e_i in G_M is labeled by its index i .) A trail from node u to node v in G_M is a sequence of edges $(e_{i_1}, \dots, e_{i_\ell})$ such that e_{i_1} starts at u , e_{i_ℓ} ends at v , the end node of each e_{i_j} is the start node of $e_{i_{j+1}}$, and the index sequence i_1, \dots, i_ℓ is strictly increasing.*

We now identify a combinatorial condition that the interaction graph should satisfy in order to accommodate MPC with a given set O of parties who receive an output.

Definition 3.3 (O -connected graph). *Let G_M be an n -party interaction graph and let $O \subseteq [n]$. We say that G_M is O -connected if for any (not necessarily distinct) pair of nodes $s, o \in [n]$ with $o \in O$, and any node $h \in [n] \setminus \{s, o\}$, there is a trail from s to o passing through h .*

Note, in particular, that the above connectivity requirement implies the existence of a trail from every node to every output node.

We now show that the above connectivity requirement is indeed necessary to realize the standard notion of security against semi-honest adversaries. We prove this for an explicit functionality that can be thought of as a natural multi-party variant of oblivious transfer. Intuitively, this functionality has the property that the adversary only learns partial information about honest parties' inputs by invoking it once, but can learn full information by invoking it twice, on any pair of input-tuples that differ in only one entry.

Definition 3.4 (MOT functionality). For $n \geq 2$ and nonempty $O \subseteq [n]$, let $\text{MOT}_O : X^n \rightarrow Y^n$ be the n -party functionality defined as follows:

- The input domain of each party is $X = \{0, 1\}^3$ and the output domain is $Y = \{0, 1\}^{n+1}$.
- Given input (c_i, x_i^0, x_i^1) from each party P_i , the functionality lets $c = c_1 \oplus \dots \oplus c_n$ and outputs (c, x_1^c, \dots, x_n^c) to all parties $P_j, j \in O$ (the output of party P_j for $j \notin O$ is the fixed string 0^{n+1}).

The proof of the following lemma formalizes an argument made in [20].

Lemma 3.5. Let $n \geq 2$ and $O \subseteq [n]$ where $|O| \geq 1$. Suppose Π securely realizes MOT_O in the presence of a semi-honest, static adversary who may corrupt any number of parties, where Π may use an arbitrary correlated randomness setup. If Π complies with an interaction pattern M , then the interaction graph G_M must be O -connected. Moreover, this holds even in the augmented semi-honest model, where the simulator can change the inputs of corrupted parties.

Proof. The high level idea is that in the ideal model, even if the simulator can arbitrarily choose the inputs of $n - 1$ corrupted parties, it can only learn one out of the last two input bits of the remaining party. We show that in the protocol, a semi-honest adversary can learn both input bits of an uncorrupted party, contradicting security. We formalize this below.

Since G_M is not O -connected, there exist nodes $s, o \in [n]$ with $o \in O$ and $h \in [n] \setminus \{s, o\}$ such that all trails from s to o avoid h . We argue that the latter implies that if all parties except h are corrupted, then by running Π once on inputs $x_i = 000$ for all corrupted parties $P_i, i \neq h$, and an unknown input $x_h = (c_h, x_h^0, x_h^1)$ for party P_h , the adversary can efficiently compute the entire input x_h from its view. Indeed, the adversary can recover x_h from (1) the output MOT_O delivers to party P_o on inputs (x_1, \dots, x_n) , obtained directly from the honest execution; and (2) the output of MOT_O on a slightly modified input, where x_s is replaced by $x'_s = 100$. The latter output can be obtained by running a mental experiment in which the view of party P_o on the modified input is simulated given the messages sent out by party P_h in the original execution.

The simulation will simply compute the exact set of messages received by party P_o on the same local inputs and random inputs, with the only difference that $x_s = 000$ is replaced by $x'_s = 100$. To see that this is possible given the

information available to the adversary, note that every message sent in the protocol can be viewed as a deterministic function of the local inputs and random inputs of the n parties. If some message received by party P_h can depend on the input of party P_s , then this message cannot influence the view of party P_o ; otherwise this would imply a trail from s to o passing through h . The adversary can therefore sequentially compute whatever modified messages are implied by the information it has (namely, inputs and random inputs of corrupted parties and messages sent out by party P_h), which includes all messages received by P_o . \square

Given Lemma 3.5, it suffices to prove a lower bound on the number of edges in an O -connected interaction graph G_M . We start with the case of a single output node $O = \{o\}$ and later extend it to the general case. The proof relies on the following lemma.

Lemma 3.6. *Let $n \geq 2$ and $O = \{o\}$ where $o \in [n]$. Suppose G_M is O -connected and $v \in [n] \setminus O$ has indegree $d \geq 2$ and outdegree 1. Then there is an O -connected $G_{M'}$ with the same number of edges in which v has indegree 1 and outdegree 1.*

Proof. Let e_{i_1}, \dots, e_{i_d} be the edges entering v , where $i_1 < \dots < i_d$. We obtain $G_{M'}$ from G_M by replacing every edge $e_{i_j} = (u_j, v)$, $1 \leq j \leq d - 1$, by the edge $e'_{i_j} = (u_j, u_d)$, where u_d is the source of e_{i_d} . An example of this transformation is given in Fig. 1 below. The transformation does not change the number of edges. It leaves e_{i_d} as the only edge entering v and does not add outgoing edges from v , thus making both the indegree and outdegree of v equal to 1 as required. Finally, since i_d is larger than the indices of all edges e'_{i_j} whose new endpoint is u_d , any trail in G_M can be replaced by a valid trail in $G_{M'}$ with the same source and destination and with a superset of the nodes of the original trail (the new trail may replace a direct edge to v by a 2-edge sub-trail passing through u_d). This implies that $G_{M'}$ is also O -connected, as required. \square

We are now ready to prove a lower bound on the number of edges for the case $|O| = 1$.

Proposition 3.7. *Let $n \geq 2$ and $O = \{o\}$ where $o \in [n]$. Suppose G_M is an O -connected n -party interaction graph. Then G_M has at least $2n - 2$ edges.*

Proof. We prove the proposition by induction on n . For the base case of $n = 2$, note that (without loss of generality) letting $s = o = 1$ and $h = 2$ imposes the existence of a trail from 1 to 1 passing through 2, which requires $m \geq 2 = 2 \cdot 2 - 2$ edges as required.

For the induction step, suppose that the proposition holds for all $k < n$, and let G_M be an O -connected n -party interaction graph with m edges. Assume towards contradiction that $m \leq 2n - 3$. We show that under this assumption, G_M can be converted into an O -connected $(n - 1)$ -party $G_{M'}$ that has $m' = m - 2$ edges. By the induction's hypothesis, this implies that $m' \geq 2(n - 1) - 2$, and so $m = m' + 2 \geq 2n - 2$, leading to the desired contradiction.

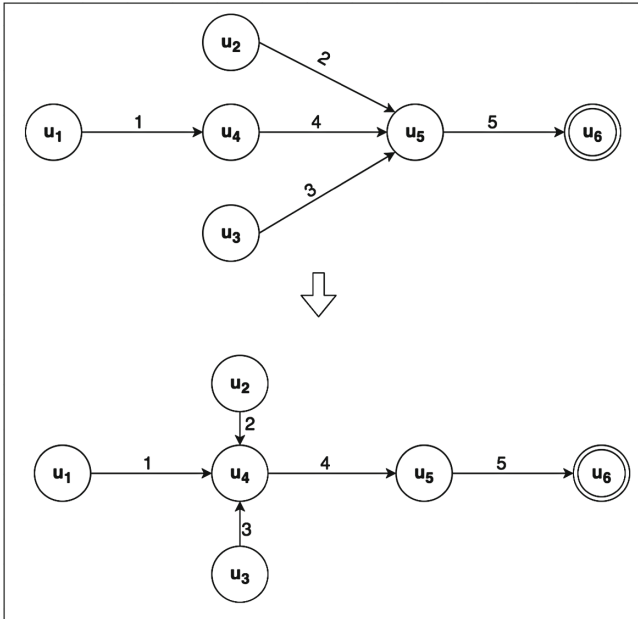


Fig. 1. Illustrating the graph transformation in the proof of Lemma 3.6. Here $o = u_6$ is the output node and $v = u_5$ is the non-output node with outdegree 1 and indegree $d = 3 \geq 2$.

The transformation from G_M to $G_{M'}$ proceeds as follows. Since O -connectivity requires each node to have at least one outgoing edge, and since $m < 2n - 2$, there must be a non-output node v whose outdegree is exactly 1. (If all outdegrees are bigger than 1, then the non-output nodes alone contribute at least $2n - 2$ edges.) Moreover, the O -connectivity of G_M also requires the indegree of v to be at least 1 (e.g., letting $v = h$ and $s = o$). By Lemma 3.6, we may assume without loss of generality that the indegree of v is also 1.

Let $e_{i_1} = (u_1, v)$ be the single edge entering v and $e_{i_2} = (v, u_2)$ be the single edge existing v . Since v should be reachable, we have $i_1 < i_2$. If $u_1 = u_2$, we can obtain $G_{M'}$ by just removing v and the two incident edges from G_M . The resulting graph $G_{M'}$ has $n - 1$ nodes and $m - 2$ edges as required, and it is O -connected because every trail in G_M that passes through v has a corresponding trail in $G_{M'}$ with the same source and destination that traverses the same set of nodes excluding v .

It remains to deal with the case where $u_1 \neq u_2$. The O -connectivity of G_M implies the existence of a trail $\tau_{u_2, v, o}$ from u_2 to the output node passing through v . We obtain $G_{M'}$ from G_M by removing the node v , replacing the two edges e_{i_1}, e_{i_2} by the single edge $e'_{i_1} = (u_1, u_2)$ (with index i_1), and removing the first edge $e_{i_0} = (u_2, u_3)$ of $\tau_{u_2, v, 1}$. Again, $G_{M'}$ has $n - 1$ nodes and $m - 2$ edges as required. Replacing e_{i_1}, e_{i_2} by e'_{i_1} clearly does not hurt O -connectivity, since

(as before) any trail passing through v can be replaced by a similar trail that only excludes v . We need to show that removing the edge e_{i_0} also does not hurt O -connectivity. Note that, since $\tau_{u_2,v,o}$ should pass through e_{i_1} and then e_{i_2} , we have $i_0 \leq i_1 < i_2$. We show that for any $h \neq u_2, o$, a trail $\tau_{u_2,h,o}$ from u_2 to o via h can be replaced by a trail $\tau'_{u_2,h,o}$ in $G_{M'}$. Indeed, by the O -connectivity of G_M , there is a trail $\tau_{v,h,o}$ in G_M from v to o via h . This trail starts with e_{i_1} , and thus all of its other edges have indices bigger than i_1 . Removing the first edge e_{i_1} , we get a trail $\tau'_{u_2,h,o}$ that does not use e_{i_0} (since $i_0 \leq i_1$), as required. \square

Finally, we extend the lower bound of Proposition 3.7 to the case of more than one output. This relies on the following lemma.

Lemma 3.8. *Let $n \geq 2$ and $O \subseteq [n]$ be a set of $k = |O| \geq 2$ output nodes. Let M be a minimal interaction pattern such that G_M is O -connected. Then:*

1. *The last edge in M enters an output node in O ;*
2. *Removing this last edge results in an interaction pattern M' such that $G_{M'}$ is O' -connected for some $O' \subset O$ with $|O'| = |O| - 1$.*

Proof. If the last edge in M does not enter an output node from O , then it can be removed from M without hurting the O -connectivity of G_M , contradicting minimality. Now suppose that the last edge in M enters $o \in O$. Removing this last edge from G_M results in an O' interaction graph for $O' = O \setminus \{o\}$. Indeed, since the removed edge has a maximal index, it cannot be used as an intermediate edge in any trail ending in $o' \in O'$. \square

Combining Proposition 3.7 and Lemma 3.8 we get the main theorem of this section.

Theorem 3.9. *Let $n \geq 2$ and $O \subseteq [n]$ be a set of $k = |O| \geq 1$ output nodes. Suppose G_M is an O -connected n -party interaction graph. Then G_M has at least $2n + k - 3$ edges.*

Proof. The theorem follows by induction on k , using Proposition 3.7 as the base case ($k = 1$) and Lemma 3.8 for the induction step. \square

Together with Lemma 3.5, we get the following corollary:

Corollary 3.10. *Let $n \geq 2$ and $O \subseteq [n]$ where $|O| = k \geq 1$. Suppose Π securely realizes MOT_O in the presence of a semi-honest, static adversary who may corrupt any number of parties, where Π may use an arbitrary correlated randomness setup. If Π complies with an interaction pattern M , then M involves at least $2n + k - 3$ messages. Moreover, this holds even in the augmented semi-honest model, where the simulator can change the inputs of corrupted parties.*

4 Upper Bounds

In this section we complement the lower bound from Sect. 3 by presenting matching upper bounds in several different models. We note that our focus here is on the computational model of security, which allows us to bypass strong lower bounds for the information-theoretic model from the recent work of Damgård et al. [13].

Using standard general transformations (cf. [18]), the secure computation of any (non-reactive) randomized multi-output functionality f can be reduced to the secure computation of a related deterministic, functionality f' that delivers the same output to all parties. This reduction does not incur additional messages. We thus restrict our attention to the latter type of functionalities.

As a final simplification, it suffices to prove an upper bound of $2n - 2$ messages for the case only one party has an output. Indeed, in the case of $k > 1$ parties should receive the output, we can first deliver the output to one of these parties using $2n - 2$ messages, and then use $k - 1$ additional messages to communicate the output to the other parties. This yields a total of $2n + k - 3$ messages, as required.

Theorem 4.1. *Let f be an n -party functionality delivering output to party P_1 . Suppose there is a 2-round n -party MPC protocol Π for f in the common random string (CRS) model. Then there is a similar protocol Π' for f in the plain model in which the parties send a total of $2n - 2$ point-to-point messages. Furthermore, if Π relies on a trusted source of correlated random inputs, then Π' can be implemented using the same correlated randomness.*

Proof. We assume for simplicity that Π does not rely on correlated randomness other than (possibly) a CRS. The “furthermore” part of the theorem is obtained by a straightforward extension of the following proof.

Let $\alpha_{i,j}$ denote the message sent from P_i to P_j in Round 1, and β_i the message sent from P_i to P_1 in Round 2. The high level idea is to use a “two-way chain” interaction pattern moving from P_1 to P_n and back to P_1 , where at each point each party computes whatever messages it can given the information received so far and forwards these messages along with previous information it received to the next party. Concretely, protocol Π' emulates the messages of Π as follows:

1. P_1 picks the CRS σ , and based on σ , its local input, and its local randomness computes the messages $\alpha_{1,j}$ for all $2 \leq j \leq n$. It sends a single message consisting of σ and the $n - 1$ messages $\alpha_{1,j}$ to P_2 .
2. For $i = 2, \dots, n - 1$, party P_i uses the Π' -message α'_{i-1} received from P_{i-1} to compute the Π -messages $\alpha_{i,j}$, for all $j \neq i$, and sends these messages to P_{i+1} together with the information received from P_{i-1} .
3. Party P_n uses the CRS σ , its local input, and its local randomness to compute the messages $\alpha_{n,j}$, $1 \leq j \leq n - 1$. It additionally uses the messages $\alpha_{i,n}$ received from P_{n-1} to compute the message β_n . It sends the messages $\alpha_{n,j}$ and β_n to P_{n-1} along with the message of P_{n-1} .

4. For $i = n - 1, \dots, 2$, party P_i uses its local input, local randomness, and the information received from P_{i+1} to compute the message β_i . It sends β_i along with the message it received from P_{i+1} to P_{i-1} .
5. Party P_1 uses its local input, local randomness, and the information received from P_2 to compute the output of Π .

Overall, the protocol involves $2n - 2$ messages ($n - 1$ in each direction), as required. Correctness follows from the fact that Π' perfectly emulates the messages sent in Π . Security follows from the fact that the view of any (static, semi-honest) adversary corrupting a subset of the parties in Π' is identically distributed (up to message ordering) to the view of a similar adversary corrupting the same subset of parties in Π . \square

Using recent 2-round MPC protocols from [4,17], we get the following corollary for message-optimal MPC in the plain model.

Corollary 4.2. *Suppose a 2-message (semi-honest) oblivious transfer protocol exists. Then, any polynomial-time n -party functionality delivering output to k parties can be securely computed in the plain model with $2n + k - 3$ messages.*

We note that the assumption that a 2-message oblivious transfer protocol exists is necessary, since such a protocol is a special case of Corollary 4.2 with $n = 2$ and $k = 1$.

We are able to further reduce the computational assumptions in the offline-online model, where a trusted source of (input-independent) correlated randomness is available. The latter can be generated by the parties themselves using an interactive MPC protocol that is carried out in an offline, input-independent pre-processing phase. Given a correlated randomness setup, 2-round MPC becomes considerably easier [10,21]. In particular, such protocols can be achieved unconditionally for functionalities in low complexity classes such as NC^1 , or can be based on any one-way function for general polynomial-time computable functionalities. The following theorem is informally mentioned in [21], we provide a proof sketch for self-containment.

Theorem 4.3. *Suppose a one-way function exists. Then, any polynomial time n -party functionality f can be realized by a 2-round protocol with a correlated randomness setup. Furthermore, the same result holds unconditionally (and with information-theoretic security) for functionalities f in the complexity class NC^1 or even (uniform) NL/poly .*

Proof (sketch). Assume for simplicity that each input of f is a single bit and the output is only revealed to P_1 ; the general case can be reduced to this case. Consider any decomposable randomized encoding [1,15,22] (or projective garbling [2]) for f . Such an encoding can be expressed as an efficiently samplable joint distribution $R_f = ((r_1^0, r_1^1), \dots, (r_n^0, r_n^1))$ such that given $(r_{x_1}^1, \dots, r_{x_n}^n)$ one can recover $f(x)$ but cannot learn anything else about x . The existence of such R_f for polynomial-time computable functionalities f (with computational hiding

of x) can be based on any one-way function [24]. For functions f in NC^1 or even (uniform) NL/poly , it exists unconditionally with perfect hiding of x [1, 15].

Given R_f as above, a protocol for f in the correlated randomness model proceeds as follows. To generate the correlated randomness, sample $((r_1^0, r_1^1), \dots, (r_n^0, r_n^1))$ from R_f , pick a secret mask $\rho_i \in \{0, 1\}$ for each input bit x_i , and use n -out-of- n (e.g., additive) secret sharing to share each (r_i^0, r_i^1) between the parties, where the pair entries are permuted according to ρ_i . That is, each party gets a “left share” of $r_i^{\rho_i}$ and a “right share” of $r_i^{1-\rho_i}$. Moreover, the permutation bit ρ_i is revealed to party P_i .

In the online phase, on input x_i , party P_i sends its masked input $x'_i = x_i \oplus \rho_i$ to all other parties. In the second round, each party sends to P_1 the n shares corresponding to the bits x'_i , namely if $x'_i = 0$ then the left share (of $r_i^{\rho_i}$) is sent and otherwise the right share (of $r_i^{1-\rho_i}$) is sent. Given the shares received from all parties, P_1 reconstructs $(r_{x_1}^1, \dots, r_{x_n}^n)$, from which it can decode $f(x_1, \dots, x_n)$. Security follows from the security of the randomized encoding and the fact that the unrevealed values $r_{1-x_i}^i$ are not revealed to the adversary even when corrupting an arbitrary strict subset of the parties. \square

Combining Theorems 4.1 and 4.3, we get the following corollary for message-optimal MPC with correlated randomness setup.

Corollary 4.4. *Suppose a one-way function exists. Then, any polynomial time n -party functionality f delivering output to k parties can be securely computed with a correlated randomness setup and $2n+k-3$ online messages. Furthermore, the same result holds unconditionally (and with information-theoretic security) for functionalities f in the complexity class NC^1 or even (uniform) NL/poly .*

5 Conclusions and Future Research

In this work we provide a tight characterization of the message complexity of computationally secure MPC in the presence of semi-honest adversaries that can corrupt any number of parties. Our work leaves several natural directions for future research.

One direction is understanding the type of achievable security and necessary setup for extending the positive results to accommodate malicious adversaries. While such an extension is fairly simple in some settings (e.g., for NC^1 functions with a correlated randomness setup and settling for “security with selective abort” [21]), characterizing the minimal message complexity in the plain model or with stronger forms of security seems like a challenging problem.

Another direction is to better understand the message complexity of MPC in the case where at most t parties can be corrupted. This relaxed setting is more sensitive to the distinction between static vs. adaptive corruption (with or without erasures) and between fixed vs. dynamic interaction pattern. Partial results are given in [5, 7, 8, 11, 13, 16].

Acknowledgements. The first and third authors were supported in part by NSF-BSF grant 2015782 and BSF grant 2012366. The first author was additionally supported by ERC grant 742754, ISF grant 1709/14, DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the DARPA through the ARL under Contract W911NF-15-C-0205. The third author was additionally supported by NSF grant 1619348, DARPA, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect the official policy or position of the DoD, the NSF, or the U.S. Government.

A Secure Multiparty Computation

For completeness, we provide here an overview of the standard definition of MPC we use. We refer the reader to [18] for a more complete treatment.

We consider by default an n -party *functionality* f to be a deterministic mapping of n inputs to n outputs.

An n -party *protocol* Π prescribes a randomized interaction between parties P_1, \dots, P_n on their local inputs x_i . This interaction may proceed in rounds, where in each round each party can send a message to each other party. Since our current focus on message complexity rather than round complexity, we may assume without loss of generality that only a single message is sent in each round. Formally, Π is a polynomial-time computable next message function that on input i (party identity), 1^k (global security parameter), x_i (local input of P_i), r_i (local random input of P_i) and $(m_{i,j})$ (sequence of messages received so far by P_i) specifies the next message P_i should send and its destination, or alternatively the local output y_i of P_i . In the plain model, the r_i are independently random bit-strings, whereas in the correlated randomness model they can be picked by a PPT sampling algorithm $D(1^k)$.

We make the following correctness requirement: if parties P_1, \dots, P_n interact according to Π on inputs 1^k and (x_1, \dots, x_n) , then they end up with local outputs $(y_1, \dots, y_n) = f(x_1, \dots, x_n)$ except with negligible probability in k .

The security of a protocol (with respect to the functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary Adv , which attacks the real execution of the protocol, there exist an adversary Sim , also referred to as a simulator, which can *learn essentially the same information* in the ideal-world. Since we consider security against semi-honest adversaries and deterministic functionalities, we are only concerned with simulating the view of Adv and not its effect on the outputs of uncorrupted parties.

The real execution. In the real execution of Π , the adversary Adv , given an auxiliary input z , corrupts a set $I \subset [n]$ of the parties and outputs their entire

view. This view consists (without loss of generality) of their inputs x_i , random inputs r_i , and messages received from other parties. (The outgoing messages are determined by the above information.) The output of Adv on a protocol Π defines a random variable $\text{REAL}_{\pi, \text{Adv}(z), I}(k, \mathbf{x})$.

The ideal execution. In the ideal world, there is a trusted party who computes f on behalf of the parties. The simulator Sim , given an auxiliary input z , corrupts a set $I \subset [n]$, receives the inputs and outputs of parties in I , and computes some (randomized) function of this information. The interaction of Sim with f defines a random variable $\text{IDEAL}_{f, \text{Sim}(z), I}(k, \mathbf{x})$ whose value is determined by the random coins of Sim .

Having defined the real and the ideal executions, we now proceed to define our notion of security. We say that Π securely computes f in the presence of semi-honest adversaries if for every $I \subset [n]$ and PPT adversary Adv (whose running time is polynomial in k) there exists a PPT simulator Sim , such that for every sequence of polynomial-size auxiliary inputs z_k and inputs $\mathbf{x} = (x_1, \dots, x_n)$, the following quantity is negligible in k :

$$|\Pr[\text{REAL}_{\Pi, \text{Adv}(z), I}(k, \mathbf{x}) = 1] - \Pr[\text{IDEAL}_{f, \text{Sim}(z), I}(k, \mathbf{x}) = 1]|.$$

We also consider the case of *information-theoretic security*, in which both Adv and Sim are computationally unbounded.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0 . In: FOCS, pp. 166–175 (2004)
2. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of CCS 2012, Raleigh, NC, USA, 16–18 October 2012, pp. 784–796 (2012)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
4. Benhamouda, F., Lin, H.: k-round MPC from k-round ot via garbled interactive circuits. Cryptology ePrint Archive, Report 2017/1125 (2017). <https://eprint.iacr.org/2017/1125>
5. Boyle, E., Chung, K.-M., Pass, R.: Large-scale secure computation: multi-party computation for (parallel) RAM programs. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 742–762. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_36
6. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: Proceedings of ITCS 2018, pp. 21:1–21:21 (2018). <https://eprint.iacr.org/2017/1248>
7. Boyle, E., Goldwasser, S., Tessaro, S.: Communication locality in secure multi-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 356–376. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_21
8. Chandran, N., Chongchitmate, W., Garay, J.A., Goldwasser, S., Ostrovsky, R., Zikas, V.: The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In: Proceedings ITCS 2015, pp. 153–162 (2015)

9. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
10. Choi, S.G., Elbaz, A., Malkin, T., Yung, M.: Secure multi-party computation minimizing online rounds. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 268–286. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_16
11. Chor, B., Kushilevitz, E.: A communication-privacy tradeoff for modular addition. *Inf. Process. Lett.* **45**(4), 205–210 (1993)
12. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_18
13. Damgård, I., Nielsen, J.B., Ostrovsky, R., Rosén, A.: Unconditionally secure computation with reduced interaction. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 420–447. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_15
14. Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.: On the communication required for unconditionally secure multiplication. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 459–488. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_16
15. Feige, U., Killian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC 1994, pp. 554–563. ACM, New York (1994)
16. Garay, J., Ishai, Y., Ostrovsky, R., Zikas, V.: The price of low communication in secure multi-party computation. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 420–446. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_14
17. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. Cryptology ePrint Archive, Report 2017/1156 (2017). <https://eprint.iacr.org/2017/1156>
18. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press, Cambridge (2004)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
20. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Proceedings of ITCS 2016, pp. 157–168 (2016). <https://eprint.iacr.org/2015/1173.pdf>
21. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 600–620. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_34
22. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC, pp. 433–442 (2008)
23. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_26
24. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)

Author Index

- Agrikola, Thomas [II-341](#)
Alamati, Navid [II-619](#)
Au, Man Ho [I-253](#)
Auerbach, Benedikt [I-348, II-403](#)
- Badertscher, Christian [I-494](#)
Bellare, Mihir [I-348](#)
Benhamouda, Fabrice [II-644](#)
Blazy, Olivier [II-644](#)
Bootle, Jonathan [II-561](#)
Brakerski, Zvika [II-702](#)
Broadnax, Brandon [II-312](#)
- Chen, Ming-Shing [II-3](#)
Chen, Wenbin [I-253](#)
Chen, Yu [II-589](#)
- Dachman-Soled, Dana [II-281](#)
Damgård, Ivan [II-530](#)
Data, Deepesh [I-675](#)
Datta, Pratish [II-245](#)
Deng, Yi [II-589](#)
Derler, David [I-219](#)
Doröz, Yarkin [I-125](#)
Döttling, Nico [I-3](#)
Ducas, Léo [II-644](#)
- El Kaafarani, Ali [II-89](#)
- Fan, Xiong [II-218](#)
Farshim, Pooya [II-371](#)
Fetzer, Valerie [II-312](#)
Frederiksen, Tore K. [I-587](#)
Fuchsbauer, Georg [I-315, II-153](#)
- Ganesh, Chaya [II-499](#)
Garay, Juan A. [II-465](#)
Garg, Sanjam [I-3](#)
Gay, Romain [II-153](#)
Gentry, Craig [II-34](#)
Giacon, Federico [I-159, I-190](#)
- Groth, Jens [II-561](#)
Gu, Dawu [I-62](#)
- Hajjabadi, Mohammad [I-3](#)
Hamlin, Ariel [I-95](#)
Han, Shuai [I-62](#)
Hanaoka, Goichiro [I-437](#)
Hart, Daniel [I-381](#)
Herold, Gottfried [I-407](#)
Hesse, Julia [II-371](#)
Heuer, Felix [I-190](#)
Hoffstein, Jeffrey [I-125](#)
Hofheinz, Dennis [II-341, II-371](#)
Huang, Zhengan [I-253](#)
Hülsing, Andreas [II-3, II-728](#)
- Ishai, Yuval [I-698](#)
- Jarecki, Stanislaw [I-644, II-431](#)
Jutla, Charanjit S. [II-123](#)
- Kashiwabara, Kenji [I-437](#)
Katsumata, Shuichi [II-89](#)
Kiayias, Aggelos [II-465](#)
Kiltz, Eike [I-159, I-348](#)
Kim, DoHoon [I-381](#)
Kirshanova, Elena [I-407, II-702](#)
Kitagawa, Fuyuki [I-32, II-187](#)
Kondi, Yashvanth [II-499](#)
Krawczyk, Hugo [II-431](#)
Krenn, Stephan [I-219](#)
Kulkarni, Mukul [II-281](#)
- Laarhoven, Thijs [I-407](#)
Lai, Junzuo [I-253](#)
Lange, Tanja [II-728](#)
Larraia, Enrique [II-371](#)
Leonardos, Nikos [II-465](#)
Li, Baiyu [I-527](#)
Li, Jin [I-253](#)
Lindell, Yehuda [I-620](#)
Ling, San [II-58](#)

- Liu, Shengli I-62
 Lorünser, Thomas I-219
 Luo, Ji II-530
 Lyu, Lin I-62
- Masny, Daniel I-3
 Matsuda, Takahiro I-280
 Maurer, Ueli I-494
 Mechler, Jeremias I-463
 Micciancio, Daniele I-527
 Micheli, Giacomo I-381
 Mittal, Manika I-698
 Müller-Quade, Jörn I-463, II-312
- Nguyen, Khoa II-58
 Nilges, Tobias I-463
 Nishimaki, Ryo II-187
- O'Neill, Adam II-34
 Oechsner, Sabine II-530
 Ohkubo, Miyako II-123
 Okamoto, Tatsuaki II-245
 Ostrovsky, Rafail I-698
- Panagiotakos, Giorgos II-465
 Pascual-Perez, Guillermo I-381
 Patra, Arpita II-499
 Peikert, Chris II-619, II-675
 Peng, Zhen I-253
 Petit, Christophe I-381
 Pinkas, Benny I-587
 Pipher, Jill I-125
 Poettering, Bertram I-159, I-190, II-403
 Prabhakaran, Manoj I-675
- Quach, Willy II-644
 Quek, Yuxuan I-381
- Ramacher, Sebastian I-219
 Reyzin, Leonid II-34
- Rijneveld, Joost II-3
 Roy, Arnab II-123
 Rupp, Andy II-312
- Samardjiska, Simona II-3
 Sarkar, Pratik II-499
 Saxena, Nitesh II-431
 Scholl, Peter I-554, II-530
 Schuldt, Jacob C. N. I-280
 Schwabe, Peter II-3
 Shahverdi, Aria II-281
 Shelat, Abhi I-95
 Shiehian, Sina II-675
 Shirvanian, Maliheh II-431
 Silverman, Joseph H. I-125
 Simkin, Mark II-530
 Slammanig, Daniel I-219
 Smeets, Kit II-728
 Song, Xuyang II-589
 Stehlé, Damien II-702
 Stephens-Davidowitz, Noah II-619
 Striecks, Christoph I-219
 Sunar, Berk I-125
- Tackmann, Björn I-494
 Tanaka, Keisuke I-32, II-187
 Tang, Qiang II-218
 Teruya, Tadanori I-437
 Tomida, Junichi II-245
- Wang, Huaxiong II-58
 Weiss, Mor I-95
 Wen, Weiqiang II-702
 Whyte, William I-125
 Wicks, Daniel I-95
- Xu, Yanhong II-58
- Yanai, Avishay I-587, I-620
 Yu, Jingyue II-589
- Zhang, Zhenfei I-125