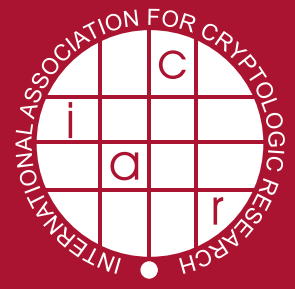
Chen-Mou Cheng · Kai-Min Chung
Giuseppe Persiano · Bo-Yin Yang (Eds.)

LNCS 9614

Public-Key Cryptography – PKC 2016

19th IACR International Conference
on Practice and Theory in Public-Key Cryptography
Taipei, Taiwan, March 6–9, 2016, Proceedings, Part I



Springer

Lecture Notes in Computer Science **9614**

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

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

Chen-Mou Cheng · Kai-Min Chung
Giuseppe Persiano · Bo-Yin Yang (Eds.)

Public-Key Cryptography – PKC 2016

19th IACR International Conference
on Practice and Theory in Public-Key Cryptography
Taipei, Taiwan, March 6–9, 2016
Proceedings, Part I

Springer

Editors
Chen-Mou Cheng
National Taiwan University
Taipei
Taiwan

Kai-Min Chung
Academia Sinica
Taipei
Taiwan

Giuseppe Persiano
Università di Salerno
Fisciano
Italy

Bo-Yin Yang
Academia Sinica
Taipei
Taiwan

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-662-49383-0 ISBN 978-3-662-49384-7 (eBook)
DOI 10.1007/978-3-662-49384-7

Library of Congress Control Number: 2016930549

LNCS Sublibrary: SL4 – Security and Cryptology

Printed on acid-free paper

This Springer imprint is published by SpringerNature
The registered company is Springer-Verlag GmbH Berlin Heidelberg

Preface

The 19th IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2016) was held March 6–9, 2016 in Taipei (Taiwan). The conference, sponsored by the International Association for Cryptologic Research (IACR), focuses on all technical aspects of public-key cryptography. These proceedings contain 34 papers selected by the Program Committee from 143 submissions. The many high-quality submissions made it easy to build a strong program but also required rejecting good papers. Each submission was judged by at least three reviewers, or four in the case of submissions by Program Committee members. The selection process included one whole month of independent review (each Program Committee member was assigned about 14 papers) followed by five more weeks of discussions. We tried to make the review and discussion system more interactive and used a new feature of the review system that allows Program Committee members to send specific questions to the authors.

We would like to thank the many authors from all over the world for submitting their papers—without them there would not be a conference. We are deeply grateful to the Program Committee for their hard work to ensure that each paper received a thorough and fair review. We gratefully acknowledge the external reviewers listed on the following pages. Our thanks go to Shai Halevi: the committee's work was tremendously simplified by his submission/review software.

January 2016

Giuseppe Persiano
Bo-Yin Yang

PKC 2016

The 19th International Conference on the Theory and Practice of Public-Key Cryptography

Taipei, Taiwan

March 6–9, 2016

Sponsored by the *International Association of Cryptologic Research*

General Chairs

Chen-Mou Cheng	National Taiwan University, Taiwan
Kai-Min Chung	Academia Sinica, Taiwan

Program Chairs

Giuseppe Persiano	Università di Salerno, Italy
Bo-Yin Yang	Academia Sinica, Taiwan

Program Committee

Joel Alwen	IST, Austria
Paulo Barreto	University of Sao Paulo, Brazil and University of Washington Tacoma, USA
Carlo Blundo	University of Salerno, Italy
Dario Catalano	University of Catania, Italy
Melissa Chase	Microsoft Research, USA
Tung Chou	TU Eindhoven, The Netherlands
Dana Dachman-Soled	University of Maryland, USA
Emiliano De Cristofaro	UCL, UK
Yvo Desmedt	UT at Dallas, USA and UCL, UK
Leo Ducas	CWI, The Netherlands
Dario Fiore	IMDEA Software Institute, Spain
Pierre-Alain Fouque	University of Rennes 1, France
Sanjam Garg	University of California, Berkeley, USA
Tim Gneysu	University of Bremen, Germany
Yuval Ishai	The Technion, Israel and UCLA, USA
Tanja Lange	TU Eindhoven, The Netherlands
Benot Libert	ENS Lyon, France
Feng-Hao Liu	Florida Atlantic University, USA
Hemanta Maji	Purdue University, USA
Alexander May	RU Bochum, Germany

Phong Q. Nguyen	Inria, France and University of Tokyo, Japan
Jesper Buus Nielsen	Aarhus University, Denmark
Tatsuaki Okamoto	NTT, Japan
Rafail Ostrovsky	UCLA, USA
Omkant Pandey	University of California, Berkeley, USA
Christophe Petit	UCL, UK
David Pointcheval	ENS Paris, France
Ahmad-Reza Sadeghi	TU Darmstadt, Germany
Dominique Schrder	Saarland University, CISPA, Germany
Peter Schwabe	Radboud University, The Netherlands
Daniel Smith-Tone	NIST, USA
Damien Stehl	ENS Lyon, France
Mehdi Tibouchi	NTT, Japan
Weng-Guey Tzeng	National Chiao Tung University, Taiwan
Daniele Venturi	Sapienza University of Rome, Italy
Moti Yung	Google and Columbia University, USA

External Reviewers

Shweta Agrawal	Cas Cremer	Vincenzo Iovino
Shashank Agrawal	Bernardo David	Malika Izabachene
Jacob Alperin-Sheriff	Gareth Davies	Zahra Jafargholi
Daniel Apon	Angelo De Caro	Aayush Jain
Saikrishna Badrinarayanan	Jean-Christophe Deneuville	Chen Jie
Shi Bai	Jérémie Detrey	Ali El Kaafarani
Lejla Batina	Julien Devigne	Dakshita Khurana
Fabrice Ben Hamouda	Nico Doettling	Eike Kiltz
Daniel J. Bernstein	Xiong Fan	Chong-Hee Kim
Sanjay Bhattacherjee	Antonio Faonio	Taechan Kim
Nina Bindel	Houda Ferradi	Paul Kirchner
Olivier Blazy	Nils Fleischhacker	Katriel Kohn-Gordon
Jonathan Bootle	Eiichiro Fujisaki	Venkata Koppula
Florian Bourse	Steven Galbraith	Luke Kowalczyk
Andrea Cerulli	Luke Garratt	Mukul Kulkarni
Jie Chen	Essam Gheddafi	Po-Chun Kuo
Ming-Shing Chen	Satrajit Ghosh	Fabien Laguillaumie
Cline Chevalier	Esha Gosh	Tancrède Lepoint
Chitchanok Chuengsatiansup	Divya Gupta	Bernardo Magri
Kai-Min Chung	Florian Göpfert	Giulio Malavolta
Michele Ciampi	Brett Hemenway	Ingo von Maurich
Sandro Coretti	Jens Hermans	Peihan Miao
Ana Costache	Gottfried Herold	Ameer Mohammed
Geoffroy Couteau	Felix Heuer	Amir Moradi
	Yun-Ju Huang	Fabrice Mouhartem
		Pratyay Mukherjee

Soheil Nematihaji
Gregory Neven
Khoa Nguyen
Ruben Niederhagen
Luca Nizzardo
Tobias Oder
Cristina Onete
Ilya Ozerov
Geovandro Pereira
Thomas Peters
Duong Hieu Phan
Krzysztof Pietrzak
Antigoni Polychroniadou
Angel Perez del Pozo
Benjamin Pring
Bartosz Przydatek
Thomas Pöppelmann
Max Rabkin
Carla Rafols
Somindu Ramanna
Carla Ràfols
Akshayaram Srinivasan
Carla Rafols Salvador
Pascal Sasdrich
Sven Schäge
Vipin Singh Sehrawat
Mark Simkin
Luisa Siniscalchi
Daniel Slamanig
Pierre-Jean Spaenlehauer
Bjrn Tackmann
Katsuyuki Takashima
Susan Thomson
Mehdi Tibouchi
Junichi Tomida
Hoang Viet Tung
Niels de Vreede
Tianhao Wang
Hoeteck Wee
Mor Weiss
Daniel Wichs
Jenny Yuan-Chun Yeh
Thomas Zacharias

Contents – Part I

Signatures

Contents – Part II

CCA Security

Trading Plaintext-Awareness for Simulatability to Achieve Chosen Ciphertext Security

Takahiro Matsuda(✉) and Goichiro Hanaoka

National Institute of Advanced Industrial Science and Technology (AIST),
Tokyo, Japan
{t-matsuda,hanaoka-goichiro}@aist.go.jp

Abstract. In PKC 2014, Dachman-Soled showed a construction of a chosen ciphertext (CCA) secure public key encryption (PKE) scheme based on a PKE scheme which simultaneously satisfies a security property called weak simulatability and (standard model) plaintext awareness (sPA1) in the presence of multiple public keys. It is not well-known if plaintext awareness for the multiple keys setting is equivalent to the more familiar notion of that in the single key setting, and it is typically considered that plaintext awareness is a strong security assumption (because to achieve it we have to rely on a "knowledge"-type assumption). In Dachman-Soled's construction, the underlying PKE scheme needs to be plaintext aware in the presence of $2k + 2$ public keys.

The main result in this work is to show that the strength of plaintext awareness required in the Dachman-Soled construction can be somehow "traded" with the strength of a "simulatability" property of other building blocks. Furthermore, we also show that we can "separate" the assumption that a single PKE scheme needs to be both weakly simulatable and plaintext aware in her construction. Specifically, in this paper we show two new constructions of CCA secure key encapsulation mechanisms (KEMs): Our first scheme is based on a KEM which is chosen plaintext (CPA) secure and plaintext aware only under the 2 keys setting, and a PKE scheme satisfying a "slightly stronger" simulatability than weak simulatability, called "trapdoor simulatability" (introduced by Choi et al. ASIACRYPT 2009). Our second scheme is based on a KEM which is 1-bounded CCA secure (Cramer et al. ASIACRYPT 2007) and plaintext aware only in the *single* key setting, and a trapdoor simulatable PKE scheme. Our results add new recipes for constructing CCA secure PKE/KEM from general assumptions (that are incomparable to those used by Dachman-Soled), and in particular show interesting trade-offs among building blocks with those used in Dachman-Soled's construction.

Keywords: Public key encryption · Key encapsulation mechanism · Chosen ciphertext security · Plaintext-awareness · Trapdoor simulatability

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 3–34, 2016.
DOI: 10.1007/978-3-662-49384-7_1

1 Introduction

1.1 Background and Motivation

For public key encryption (PKE), security (indistinguishability) against chosen ciphertext attacks (CCA) [23,46,49] is nowadays considered as a de-facto standard security notion required in most practical situations/applications in which PKE schemes are used. CCA security is quite important in both practical and theoretical points of view. It implies security against practical attacks (e.g. Bleichenbacher's attack [8]) and it also implies very strong and useful security notions, such as non-malleability [23] and universal composability [10]. Thus, constructing and understanding CCA secure PKE schemes is one of the central research themes in the area of cryptography. In this paper, we focus on the constructions of CCA secure PKE schemes and its closely related primitive called *key encapsulation mechanism* (KEM) from general cryptographic assumptions. There have been a number of works that show that several different kinds of cryptographic primitives are sufficient to realize CCA secure PKE/KEM: These include trapdoor permutations [23] (with some enhanced property [27]), identity-based encryption [12] and a weaker primitive called tag-based encryption [32], lossy trapdoor function [48] and related primitives [13,33,42,50,54], PKE schemes with weaker-than-but-close-to-CCA security [31,34,40], positive results on cryptographic obfuscation [38,52], the combination of a CPA secure PKE scheme and a strong form of hash functions [39], and very recently, the combination of a sender non-committing encryption scheme and a key-dependent-message secure symmetric key encryption (SKE) scheme [41]. (We review more works in Sect. 1.4).

In PKC 2014, Dachman-Soled [18] showed a construction of a CCA secure PKE scheme based on a PKE scheme which simultaneously satisfies a security property called weak simulatability [20,43] and (standard model) plaintext awareness (sPA1) [5] in the presence of multiple public keys [43], which is based on the earlier work by Myers et al. [43] who showed a construction of a PKE scheme that achieves security slightly weaker than CCA (the so-called cNM-CCA1 security). Plaintext awareness was first introduced by Bellare and Rogaway [7] as a useful notion for showing CCA security of a PKE scheme in the random oracle model [6], and was used in a number of random-oracle-model constructions (e.g. [7,24,25,47]). Bellare and Palacio [5] defined the standard model versions of plaintext awareness.[1] The plaintext awareness notions were further studied by subsequent works (e.g. [20]). The most works on plaintext awareness studied the notions for the single key setting. The extension to the multiple keys setting was first introduced by Myers et al. [43].

We note that it is not well-known or well-studied if plaintext awareness for the multiple keys setting is equivalent to the more familiar notion of plaintext

[1] [5] defined several versions (PA0, PA1, and PA2, with their computational/statistical/perfect variants) for standard model plaintext awareness. As in the previous works [18,43], we focus on the statistical PA1 notion in the multiple keys setting (denoted by "$\mathtt{sPA1}_\ell$", where ℓ denotes the number of public keys).

awareness in the single key setting, and it is typically considered that plaintext awareness is a strong security assumption (because to achieve it we have to rely on a "knowledge"-type assumption). In the construction of [18], the underlying PKE scheme needs to be plaintext aware in the presence of $2k + 2$ public keys. Our motivation in this work is to clarify whether we can weaken the assumption of plaintext awareness in Dachman-Soled's construction [18]. As mentioned in [18], a plaintext aware (sPA1) PKE scheme seems almost like a CCA1 secure PKE scheme [46], but it seems not possible to replace the building block PKE scheme in [18] with a CCA1 secure scheme to remove the plaintext awareness. It is currently not known if we can construct a CCA secure PKE scheme only from a CPA secure scheme or even from a CCA1 secure scheme. We believe that studying the possibility of weakening the assumption of plaintext awareness from [18] thus is expected to lead to deepening our knowledge on this topic, and generally contribute to the long line of research on clarifying the minimal general assumption that implies CCA secure PKE.

1.2 Our Contributions

Based on the motivation mentioned above, we study the possibility of weakening the requirements of plaintext awareness used in Dachman-Soled's construction [18], and come up with new results that show that the strength of plaintext awareness required in [18] can be somehow "traded" with the strength of a "simulatability" property of other building blocks. Furthermore, we also show that we can "separate" the requirement that a single PKE scheme needs to be simultaneously weakly simulatable and plaintext aware, in her construction.

Specifically, in this paper we show two new constructions of CCA secure KEMs (which are given in Sect. 4), based on the assumptions that are *incomparable* to those used in [18]:

- Our first construction (Sect. 4.1) is based on a KEM which is chosen plaintext (CPA) secure and plaintext aware only under the 2 keys setting[2], and a PKE scheme satisfying a "slightly stronger" simulatability than weak simulatability, called "trapdoor simulatability" (introduced by Choi et al. [14]). Actually, although we write that it is "slightly stronger", it is formally *incomparable* to weak simulatability. For more details, see Sect. 1.3.
- Our second construction (Sect. 4.2) is based on a KEM which is 1-bounded CCA secure [15] and plaintext aware only in the *single* key setting, and a trapdoor simulatable PKE scheme. We can in fact slightly weaken the requirement of 1-bounded CCA security to CPA security in the presence of one "plaintext-checking" query [1,47]. We will also show that we can construct a KEM satisfying simultaneously 1-bounded CCA security and plaintext awareness under the single key setting, based on a KEM satisfying CPA security and plaintext awareness under the $2k$ keys setting, via the recent result by Dodis and Fiore [21, Appendix C].

[2] Plaintext awareness for KEMs is defined analogously to that for PKE. See Sect. 2.1.

One may wonder the meaning of the second construction, because if we use a KEM that is plaintext aware under $O(k)$ keys setting, there is no merit compared to our first construction. We are however considering it to be still meaningful in several aspects, and we refer the reader to Sect. 4.2 for more discussions regarding the second construction.

Note that from CCA secure KEMs, we can immediately obtain full-fledged PKE schemes by using CCA secure SKE [16].

We emphasize that we do not require plaintext awareness and the trapdoor simulatability property to be satisfied by a single building block. This "separation" of the requirements should be contrasted with Dachman-Soled's construction [18], the building block PKE scheme of which is required to satisfy plaintext awareness and the weak simulatability property simultaneously. We also again emphasize that the assumptions on which both of our constructions are based, are *incomparable* to those used in [18]. Thus, our results add new recipes for constructing CCA secure PKE/KEM from general assumptions (and thus the assumptions that we use could be new targets that are worth pursuing), and also show interesting trade-offs regarding assumptions with Dachman-Soled's construction.

1.3 Technical Overview

Assumptions on the Building Blocks. Trapdoor simulatable PKE (TSPKE) [14] is the key building block for our constructions. TSPKE is a *weaker* (relaxed) version of *simulatable PKE* that was originally formalized by Damgård and Nielsen [19]. Simulatable PKE admits "oblivious sampling" of both public keys and ciphertexts (i.e. sampling them without knowing the randomness or plaintext) in such a way that honestly generated public keys and ciphertexts can be later convincingly explained that they were generated obliviously. These properties are realized by requiring that the key generation algorithm and the encryption algorithm have their own "oblivious sampling" algorithm and its corresponding "inverting" algorithm (where the inverting algorithm corresponds to the algorithm that "explains" that an honest generated public key (or a ciphertext) is sampled obliviously). The difference between TSPKE and simulatable PKE is whether we allow the "inverting" algorithm to take the randomness (and the plaintext) used by the ordinary algorithms (key generation and encryption algorithms) as input. TSPKE allows to take these inputs, while ordinary simulatable PKE does not, which makes the security property of TSPKE weaker but easier to achieve. For our purpose, we only need even a simplified version of TSPKE than the formalization in [14]: we only require a pair (pk, c) of public key/ciphertext (or, a "transcript") can be obliviously sampled, but not each of pk and c can be so (which is the formalization in [14]). It was shown [14,19] that we can realize TSPKE from a number of standard cryptographic assumptions, such as the computational and decisional Diffie-Hellman assumptions, RSA, Factoring, and lattice based assumptions. (For more details, see Sect. 2.2).

On the other hand, a weakly simulatable PKE scheme (used in the constructions in [18,43]) considers oblivious sampling only for the encryption algorithm.

However, the definition of weakly simulatable PKE used in [18,43] does not allow the inverting algorithms to take the randomness and the plaintext used by the ordinary encryption algorithm. Therefore, strictly speaking, the "strength" of these primitives as "general cryptographic assumptions" are actually *incomparable*. Nonetheless, the reason why we still think that weakly simulatable PKE could be viewed as a weaker primitive, is that it does not require the key generation algorithm to be obliviously samplable. In fact, this difference is very important for our work. It is this simple difference between TSPKE and weakly simulatable PKE that enables us to weaken the plaintext awareness required in [18], from plaintext awareness in the presence of $O(k)$ keys in [18] into that under only $O(1)$ keys in our constructions.

Ideas for the Constructions. Other than employing TSPKE instead of weakly simulatable PKE, the ideas for our constructions and their security analyses are similar to those in [18]. In particular, the construction of [18] and our constructions are based on the Dolev-Dwork-Naor (DDN) construction [23], but we do not require a non-interactive zero-knowledge proof to ensure the validity of a ciphertext. Instead, the approach of the "double-layered" construction of Myers and Shelat [44] (and its simplifications [31,37,40] and variants [38,39,41]) is employed, in which a ciphertext consists of the "inner"-layer and "outer"-layer, and the randomness used for generating an outer ciphertext is somehow embedded into an inner ciphertext, so that in the decryption, the validity of the outer ciphertext can be checked by "re-encryption" using the randomness recovered from the inner ciphertext. (In our constructions, the inner-layer encryption is done by a KEM). In fact, we do a simplification to [18] by removing a one-time signature scheme in [18], by using a commitment scheme, based on the ideas employed in the recent constructions [38,39,41].

Recently, Matsuda and Hanaoka [39] introduced the notion of *puncturable tag-based encryption* (PTBE) which abstracts and formalizes the "core" structure of the DDN construction [23]. We define the trapdoor simulatability property for PTBE (and call the primitive *trapdoor simulatable PTBE*) in Sect. 3, and use this primitive as an "intermediate" building block in our constructions. (This primitive could have other applications than constructing CCA secure PKE, and may be of independent interest). We also show (in the full version) how to construct a trapdoor simulatable PTBE scheme from a TSPKE scheme. This construction is exactly the same as the construction of a PTBE scheme from a CPA secure PKE scheme used in [39], which is in turn based on the original DDN construction.

Ideas for the Security Proofs. We briefly recall the construction and the security proof in [18], and explain the difference in our proofs and that in [18]. As mentioned above, the construction of [18] is double-layered, where the outer encryption is like the "DDN-lite" construction (i.e. the DDN construction without a non-interactive zero-knowledge proof), and the inner encryption is a multiple-encryption by two PKE schemes. Both the inner and outer encryption schemes use the same building block, with independently generated public keys: $2k$ keys

for the outer-layer encryption (that does DDN-lite-encryption) and 2 keys for the inner-layer encryption (that does multiple-encryption by two encryptions). Roughly speaking, in the security proof, [18] constructs a CPA adversary (reduction algorithm) for the inner-layer encryption, from a CCA adversary $\mathcal{A}$ against the entire construction. The reduction algorithm of course has to somehow answer $\mathcal{A}$'s decryption queries, and this is the place where plaintext awareness comes into play. Plaintext awareness in the ℓ keys setting ($\mathtt{sPA1}_\ell$ security) ensures that for any algorithm $\mathcal{C}$ (called "ciphertext creator") that receives a set of public keys $(pk_i)_{i\in\{1,\ldots,\ell\}}$ and a randomness $r_{\mathcal{C}}$ as input and makes decryption queries, there exists an extractor $\mathcal{E}$ that also receives $(pk_i)_{i\in\{1,\ldots,\ell\}}$ and $r_{\mathcal{C}}$ as input, and can "extract" the plaintext from a ciphertext queried by $\mathcal{C}$. (In our actual security proofs, we denote the "ciphertext creator" by "$\mathcal{A}'$", but for the explanation here we continue to use $\mathcal{C}$ for clarity). The idea in the proof in [18] is to use an extractor guaranteed by plaintext awareness to answer the CCA adversary $\mathcal{A}$'s decryption queries. The problem that arises here is: how do we design the algorithm $\mathcal{C}$ with which the extractor $\mathcal{E}$ is considered? Since the extractor $\mathcal{E}$ needs to be given the randomness $r_{\mathcal{C}}$ used by $\mathcal{C}$, if we naively design $\mathcal{C}$, the reduction algorithm cannot use the extractor $\mathcal{E}$ while embedding its instances (the public key and the challenge ciphertext) in the reduction algorithm's CPA security experiment into $\mathcal{A}$'s view. The approach in [18] is to consider a modified version of the CCA security experiment in which all component ciphertexts (i.e. ciphertexts for the outer-layer encryption) are generated obliviously using some randomness r (which can be performed due to the weak simulatability property of the underlying PKE scheme), and view this modified experiment as a ciphertext creator $\mathcal{C}$ that takes as input $\ell = 2k+2$ public keys (for both inner-/outer-layer encryptions) and a randomness $r_{\mathcal{C}}$ consisting of the randomness $r_{\mathcal{A}}$ used by $\mathcal{A}$ and the randomness r used for oblivious generation of the component ciphertexts in $\mathcal{A}$'s challenge ciphertext. ($r_{\mathcal{C}}$ actually also contains some additional randomness used for generating the remaining parts of $\mathcal{A}$'s challenge ciphertext, but we ignore it here for simplicity). Designing the algorithm $\mathcal{C}$ in this way, the extractor $\mathcal{E}$ corresponding to $\mathcal{C}$ can be used to answer $\mathcal{A}$'s decryption queries while the reduction algorithm (attacking the CPA security of the inner-layer encryption) can perform the reduction.

Our main idea for weakening the requirement of plaintext awareness for the building blocks, from $2k+2$ keys in [18] to $O(1)$ keys, is due to the observation that by relying on the trapdoor simulatability property for the outer-layer encryption, we can "push" the public keys for the outer-layer encryption, into the "randomness" $r_{\mathcal{C}}$ for the ciphertext creator $\mathcal{C}$ (with which the extractor $\mathcal{E}$ is considered), by generating the public keys regarding the outer-layer encryption also obliviously. In order to make this idea work, we thus consider a different design strategy for the ciphertext creator $\mathcal{C}$. This also enables us to "separate" the requirement that a single building block PKE scheme needs to be simultaneously plaintext aware and simulatable, because we need the simulatability only for the outer-layer encryption.

Actually, like the security proof of the construction in [18], we need to deal with a "bad" decryption query, which is a ciphertext such that its actual decryption result (by the normal decryption algorithm with a secret key) differs from the decryption result obtained by using the extractor $\mathcal{E}$. (Such a decryption query makes the simulation of the decryption oracle by the reduction algorithm fail). Our first construction uses the clever trick of Dachman-Soled [18] of using two CPA secure PKE schemes (that each encrypts a "share" of 2-out-of-2 secret sharing) and their plaintext awareness under 2 keys setting. (As mentioned earlier, in fact, we use a KEM instead of a PKE scheme for the inner encryption). Dachman-Soled's approach enables us to use the CPA security and the ability of "detecting" bad queries at the same time. Our second construction is a simplification of our first construction, where we employ a "single" KEM for the inner layer, as opposed to multiple-encryption by two KEMs in our first construction. To detect "bad" decryption queries by an adversary, we employ the ideas and techniques from [31,37,40,44] of using "1-bounded CCA" security [15]. (As mentioned earlier, in fact, CPA security in the presence of one "plaintext-checking" query [1,47] is sufficient for our purpose). For more details on these, see Sect. 4.

1.4 Related Work

The notion of CCA security for PKE was formalized by Naor and Yung [46] and Rackoff and Simon [49]. Since the introduction of the notion, CCA secure PKE schemes have been studied in a number of papers, and thus we only briefly review constructions from general cryptographic assumptions. Dolev et al. [23] showed the first construction of a CCA secure PKE scheme, from a CPA secure scheme and a NIZK proof system, based on the construction by Naor and Yung [46] that achieves weaker non-adaptive CCA (CCA1) security. These NIZK-based constructions were further improved in [35,51,53]. Canetti et al. [12] showed how to transform an identity-based encryption scheme into a CCA secure PKE scheme. Kiltz [32] showed that the transform of [12] is applicable to a weaker primitive of tag-based encryption (TBE). Peikert and Waters [48] showed how to construct a CCA secure PKE scheme from a *lossy* trapdoor function (TDF). Subsequent works showed that TDFs with weaker security/functionality properties are sufficient for obtaining CCA secure PKE schemes [13,33,42,50,54]. Hemenway and Ostrovsky [29] showed how to construct a CCA secure scheme in several ways from homomorphic encryption that has some appropriate properties, and the same authors [30] showed that one can construct a CCA secure PKE scheme from a lossy encryption scheme [4] if it can encrypt a plaintext longer than the length of randomness consumed by the encryption algorithm. Myers and Shelat [44] showed that a CCA secure PKE scheme for 1-bit messages can be turned into one with an arbitrarily large plaintext space. Hohenberger et al. [31] showed that CCA secure PKE can be constructed from a PKE with a weaker security notion called detectable CCA security, from which we can obtain a 1-bit-to-multi-bit transformation for CCA security in a simpler manner than [44]. The simplicity and efficiency of [44] were further improved by Matsuda and Hanaoka [37,40]. Lin and Tessaro [34] showed how to amplify weak CCA security

into strong (ordinary) CCA secure one. Matsuda and Hanaoka [38] showed how to construct a CCA secure PKE scheme by using a CPA secure PKE scheme and point obfuscation [9,36], and the same authors [39] showed a CCA secure PKE scheme from a CPA secure PKE scheme and a family of hash functions satisfying the very strong security notion called universal computational extractors (UCE) [3]. The same authors [41] recently also showed that a CCA secure PKE scheme can be built from the combination of a sender non-committing encryption scheme and a key-dependent-message secure SKE scheme. More recently, Hajiabadi and Kapron [28] showed how to construct a CCA secure PKE scheme, from a 1-bit PKE scheme that satisfies circular security and has the structural property called reproducibility.

As has been stated several times, Dachman-Soled [18] showed how to construct a CCA secure PKE scheme from a PKE scheme which simultaneously satisfies weak simulatability [43] and the (standard model) plaintext awareness under the multiple keys setting, which is built based on the result by Myers et al. [43] who showed a PKE scheme satisfying the so-called cNM-CCA1 security, from the same building blocks as [18]. Sahai and Waters [52] showed (among other cryptographic primitives) how CCA secure PKE and KEMs can be constructed using an indistinguishability obfuscation [2,26].

1.5 Paper Organization

In Sect. 2 (and in Appendix A), we review definitions of primitives and security notions that are necessary for explaining our results. In Sect. 3, we introduce the notion of trapdoor simulatable PTBE, which is an extension of PTBE introduced in [39], and works as one of main building blocks of our proposed KEMs in the next section. Finally, in Sect. 4, we show our main results: two constructions of KEMs that show the "trade-off" between "simulatability" property and "plaintext awareness" in Dachman-Soled's construction [18].

2 Preliminaries

In this section, we review the basic notation and the definitions for plaintext awareness ($\mathtt{sPA1}_\ell$ security) [5,18,43] of a KEM, trapdoor simulatability properties of a PKE scheme and a commitment scheme, and the syntax of a puncturable tag-based encryption (PTBE) scheme. The definitions for standard cryptographic primitives with standard security definitions that are not reviewed in this section are given in Appendix A, which include PKE, KEMs, and commitment schemes.

Basic Notation. $\mathbb{N}$ denotes the set of all natural numbers, and for $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. "$x \leftarrow y$" denotes that x is chosen uniformly at random from y if y is a finite set, x is output from y if y is a function or an algorithm, or y is assigned to x otherwise. If x and y are strings, then "$|x|$" denotes the bit-length of x, "$x\|y$" denotes the concatenation x and y, and "$(x \stackrel{?}{=} y)$" is

the operation which returns 1 if $x = y$ and 0 otherwise. "(P)PTA" stands for a *(probabilistic) polynomial time algorithm*. For a finite set S, "$|S|$" denotes its size. If $\mathcal{A}$ is a probabilistic algorithm ,then "$y \leftarrow \mathcal{A}(x; r)$" denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness, and we just write "$y \leftarrow \mathcal{A}(x)$" if we do not need to make the randomness used by $\mathcal{A}$ explicit. If furthermore $\mathcal{O}$ is a function or an algorithm, then "$\mathcal{A}^{\mathcal{O}}$" means that $\mathcal{A}$ has oracle access to $\mathcal{O}$. A function $\epsilon(k) : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all positive polynomials $p(k)$ and all sufficiently large $k \in \mathbb{N}$, we have $\epsilon(k) < 1/p(k)$. Throughout this paper, we use the character "k" to denote a security parameter.

2.1 Plaintext Awareness for Multiple Keys Setup ($\mathtt{sPA1}_\ell$ Security)

Here, we review the definition of (statistical) plaintext awareness for multiple keys setup [18,43] (denoted by $\mathtt{sPA1}_\ell$ *security*, where ℓ denotes the number of keys). Unlike these previous works, we define it for a KEM, rather than a PKE scheme, but we can define plaintext awareness for a KEM in essentially the same way as that for a PKE scheme.

Let $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM (where we review the definition of a KEM in Appendix A), and $\ell = \ell(k) > 0$ be a polynomial. Let $\mathcal{A}$ be an algorithm (called a "ciphertext creator") that takes a set of public keys $(pk_i)_{i\in[\ell]}$ as input, and makes decapsulation queries of the form $(j \in [\ell], c)$ which is supposed to be answered with $K = \mathsf{Decap}(sk_j, c)$. For this $\mathcal{A}$, we consider the corresponding "(plaintext) extractor" $\mathcal{E}$: It is a stateful algorithm that initially takes a set of public keys $(pk_i)_{i\in[\ell]}$ and the randomness $r_{\mathcal{A}}$ consumed by $\mathcal{A}$, and expects to receive "decapsulation" queries of the form $q = (j \in [\ell], c)$; Upon a query, it tries to extract a session-key K corresponding to c so that $K = \mathsf{Decap}(sk_j, c)$, where sk_j is the secret key corresponding to pk_j. After $\mathcal{E}$ extracts a session-key, it may update its internal state to prepare for the next call. Informally, a KEM Γ is said to be $\mathtt{sPA1}_\ell$ secure if for all PPTA ciphertext creators $\mathcal{A}$, there exists a corresponding PPTA extractor $\mathcal{E}$ that can work as $\mathcal{A}$'s decapsulation oracle in the experiment above.

More formally, for $\mathcal{A}$ that makes $Q = Q(k)$ decapsulation queries, $\mathcal{E}$, and ℓ, consider the following experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k)$:

$$\begin{aligned}&\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) : [\ \forall i \in [\ell] : (pk_i, sk_i) \leftarrow \mathsf{KKG}(1^k);\ r_{\mathcal{A}} \leftarrow \{0,1\}^*;\\ &\quad \mathsf{st}_{\mathcal{E}} \leftarrow ((pk_i)_{i\in[\ell]}, r_{\mathcal{A}});\ \text{Run } \mathcal{A}^{\mathcal{E}(\mathsf{st}_{\mathcal{E}},\cdot)}((pk_i)_{i\in[\ell]}; r_{\mathcal{A}}) \text{ until it terminates;}\\ &\qquad\qquad \text{If } \exists i \in [Q] : \mathsf{Decap}(sk_{j_i}, c_i) \neq K_i \text{ then return 1 else return 0.}],\end{aligned}$$

where (j_i, c_i) represents $\mathcal{A}$'s i-th decapsulation query (which $\mathcal{A}$ expects to be decapsulated as a ciphertext under pk_{j_i}), and K_i represents the answer (i.e. "decapsulation result" of c_i) computed by the algorithm $\mathcal{E}$. In the experiment, $\mathcal{E}$ is the (possibly stateful) extractor which initially takes $\mathsf{st}_{\mathcal{E}} = ((pk_i)_{i\in[\ell]}, r_{\mathcal{A}})$ as input, and works like $\mathcal{A}$'s decapsulation oracle, as explained above.

Definition 1. *Let $\ell = \ell(k) > 0$ be a polynomial. We say that a KEM Γ is $\mathtt{sPA1}_\ell$ secure if for all PPTAs (ciphertext creator) $\mathcal{A}$, there exists a stateful PPTA (extractor) $\mathcal{E}$ such that $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) := \Pr[\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma,\mathcal{A},\mathcal{E},\ell}(k) = 1]$ is negligible.*

If $\ell = 1$, then $\mathtt{sPA1}_\ell$ security is equivalent to statistical PA1 security defined by Bellare and Palacio [5]. By definition, trivially, $\mathtt{sPA1}_x$ implies $\mathtt{sPA1}_y$ for $x > y$. However, to the best of our knowledge, whether there is an implication (or separation) for the opposite direction, is not known.

2.2 (Simplified) Trapdoor Simulatable Public Key Encryption

Trapdoor simulatable PKE (TSPKE) [14] is a *relaxed* version of simulatable PKE [19]. Simulatable PKE admits "oblivious sampling" of both public keys and ciphertexts (i.e. sampling them without knowing the randomness or plaintext) in such a way that honestly generated public keys and ciphertexts can be later convincingly explained that they were generated obliviously.[3] These properties are realized by requiring that the key generation algorithm and the encryption algorithm have their own "oblivious sampling" algorithm and its corresponding "inverting" algorithm (where the inverting algorithm corresponds to the algorithm that explains that an honest generated public key (or a ciphertext) is sampled obviously). The difference between TSPKE and simulatable PKE, is whether we allow for the "inverting" algorithm to take the randomness (and the plaintext) used by the ordinary algorithms PKG and Enc as input. Since the "inverting" algorithm in TSPKE is allowed to see more information than that in simulatable PKE, the former primitive is strictly weaker (and easier to construct) than the latter.

For our purpose, we only need even a simplified version of TSPKE of [14]: we only require a pair (pk, c) of public key/ciphertext (or, "transcript) can be obliviously sampled [14], but not each of pk and c can be so. A TSPKE scheme with such a simplified syntax may not be useful for constructing non-committing encryption (as done in [14,19]), but sufficient for our purpose in this paper.

Definition 2. *We say that a PKE scheme*[4] $\Pi = (\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec})$ *is* trapdoor simulatable *(and say that Π is a trapdoor simulatable PKE (TSPKE) scheme) if Π has two additional PPTAs* $(\mathsf{oSamp}_\Pi, \mathsf{rSamp}_\Pi)$ *with the following properties:*

- oSamp_Π *is the oblivious-sampling algorithm which takes* 1^k *as input, and outputs an "obliviously generated" public key/ciphertext pair* (pk, c).
- rSamp_Π *is the inverting algorithm (corresponding to* oSamp_Π*) that takes randomness* r_g *and* r_e*, and a plaintext* m *(which are supposed to be used as* $(pk, sk) \leftarrow \mathsf{PKG}(1^k; r_g)$ *and* $c \leftarrow \mathsf{Enc}(pk, m; r_e)$*) as input, and outputs a string* $\widehat{r}$ *(that looks like a randomness used by* oSamp_Π*).*
- ***(Trapdoor Simulatability).*** *For all PPTAs* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathsf{Adv}^{\mathtt{TSPKE}}_{\Pi,\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{A}}(k) = 1]|$ *is negligible, where the experiments* $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k)$ *and* $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\Pi,\mathcal{A}}(k)$ *are defined as in Fig. 1 (upper-left and upper-right, respectively).*

[3] (Trapdoor) simulatable PKE scheme was introduced as a building block for constructing non-committing encryption [11].

[4] The syntax of PKE is reviewed in Appendix A.

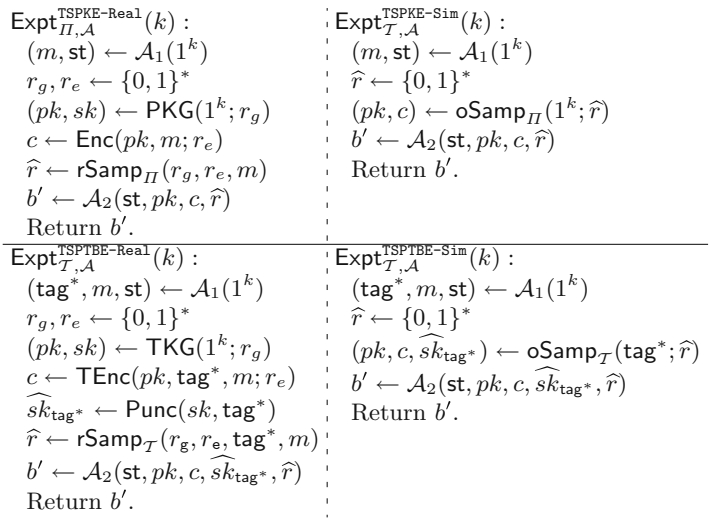


Fig. 1. Security experiments for defining security of TSPKE (upper-left and upper-right) and those for defining security of TSPTBE (bottom-left and bottom-right)

Concrete Instantiations of TSPKE. Since our definition of TSPKE is a simplified (and hence weaker) version of the definition by Choi et al. [14], and TSPKE is a weaker primitive than a simulatable PKE scheme in the sense of Damgård and Nielsen [19], we can use any of (trapdoor) simulatable PKE schemes shown in these works. In particular, we can construct a TSPKE scheme from most of the standard cryptographic assumptions such as the computational and decisional Diffie-Hellman, RSA, factoring, and learning-with-errors assumptions [14,19]. (For example, the ElGamal encryption, Damgård's ElGamal encryption, and Cramer-Shoup-Lite encryption schemes can be shown to be a TSPKE scheme if they are implemented in a simulatable group [20]). In terms of "general" cryptographic assumptions, Damgård and Nielsen [19] showed that a simulatable PKE scheme can be constructed from a family of trapdoor permutations with the simulatability property, in which the key generation and the domain-sampling algorithms have the oblivious sampling property (which is defined analogously to simulatable PKE). Hence, we can also construct a TSPKE from it.

2.3 Trapdoor Simulatable Commitment Schemes

Let $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$ be a commitment scheme. (We review the syntax of a commitment scheme and its "target-binding" property in Appendix A).

We define the trapdoor simulatability property of a commitment scheme $\mathcal{C}$ in exactly the same way as the trapdoor simulatability of a PKE scheme. Namely, we require that there be the oblivious sampling algorithm $\mathsf{oSamp}_{\mathcal{C}}$ (for sampling a key/commitment pair (ck, c)) and the corresponding inverting algorithm $\mathsf{rSamp}_{\mathcal{C}}$, whose interfaces are exactly the same as oSamp_{Π} and rSamp_{Π} of a

TSPKE scheme, respectively. We say that a commitment scheme $\mathcal{C}$ is *trapdoor simulatable* (and say that $\mathcal{C}$ is a trapdoor simulatable commitment scheme) if for all PPTA adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathtt{TSCom}}_{\mathcal{C},\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k) = 1]|$ is negligible, where the experiments $\mathsf{Expt}^{\mathtt{TSCom\text{-}Real}}_{\mathcal{C},\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSCom\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k)$ are defined in exactly the same way as $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Real}}_{\Pi,\mathcal{A}}(k)$ and $\mathsf{Expt}^{\mathtt{TSPKE\text{-}Sim}}_{\mathcal{C},\mathcal{A}}(k)$ for a TSPKE scheme, respectively (and thus we do not write down them).

We can achieve a commitment scheme which satisfies target-binding, trapdoor simulatability, and the requirement of the size of commitments (namely we require the size of commitments to be k-bit for k-bit security), only from a TSPKE scheme and a universal one-way hash function (UOWHF) [45], just by hashing a ciphertext of the TSPKE scheme by the UOWHF. This construction is given in the full version.

2.4 Puncturable Tag-Based Encryption

Here, we recall the syntax of puncturable tag-based encryption (PTBE), which was introduced by Matsuda and Hanaoka [39] as an abstraction of the "core" structure of the Dolev-Dwork-Naor (DDN) construction [23]. Similarly to [39], we use PTBE as an intermediate building block to reduce the description complexity of our proposed constructions in Sect. 4.

Intuitively, a PTBE scheme is a TBE scheme that has a mechanism for generating a "punctured" secret key $\widehat{sk}_{\mathsf{tag}^*}$, according to a "punctured point" tag tag^*. The punctured secret key can be used to decrypt all "honestly generated" ciphertexts that are generated under tags that are different from tag^*, while the punctured secret key is useless for decrypting ciphertexts generated under tag^*.

Formally, a PTBE scheme consists of the five PPTAs $(\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ among which the latter three algorithms are deterministic, with the following interface:

$$\underline{\textbf{Key Generation:}} \quad (pk, sk) \leftarrow \mathsf{TKG}(1^k) \qquad \underline{\textbf{Encryption:}} \quad c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}, m) \qquad \underline{\textbf{Decryption:}} \quad m \ (\text{or } \perp) \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$$

$$\underline{\textbf{Puncturing:}} \quad \widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*) \qquad \underline{\textbf{Punctured Decryption:}} \quad m \ (\text{or } \perp) \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$$

where (pk, sk) is a public/secret key pair, c is a ciphertext of a plaintext m under pk and a tag $\mathsf{tag} \in \{0,1\}^k$, and $\widehat{sk}_{\mathsf{tag}^*}$ is a "punctured" secret key corresponding to a tag $\mathsf{tag}^* \in \{0,1\}^k$.

We require for all $k \in \mathbb{N}$, all tags $\mathsf{tag}^*, \mathsf{tag} \in \{0,1\}^k$ such that $\mathsf{tag}^* \neq \mathsf{tag}$, all (pk, sk) output from $\mathsf{TKG}(1^k)$, all plaintexts m, and all ciphertexts c output from $\mathsf{TEnc}(pk, \mathsf{tag}, m)$, it holds that $\mathsf{TDec}(sk, \mathsf{tag}, c) = \widehat{\mathsf{TDec}}(\mathsf{Punc}(sk, \mathsf{tag}^*), \mathsf{tag}, c) = m$.

In [39], the security notion called "extended CPA security" was defined as a security notion of PTBE. In our proposed KEMs, we need a stronger security property for PTBE, which is an analogue of TSPKE, and we will introduce it in the next section.

3 Trapdoor Simulatable PTBE

In this section, we define trapdoor simulatability of a PTBE scheme, in the same way as that of a PKE scheme and a commitment scheme. However, for the oblivious sampling algorithm, we let it take a "punctured point" tag tag^* as input, and require that it output the punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$ (corresponding to tag^*) in addition to a public key/ciphertext pair (pk, c).

Formally, we define a trapdoor simulatable PTBE (TSPTBE) as follows:

Definition 3. *We say that a PTBE scheme* $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ *is* trapdoor simulatable *(and say that* $\mathcal{T}$ *is a trapdoor simulatable PTBE (TSPTBE) scheme) if* $\mathcal{T}$ *has two additional PPTAs* $(\mathsf{oSamp}_{\mathcal{T}}, \mathsf{rSamp}_{\mathcal{T}})$ *with the following properties:*

- $\mathsf{oSamp}_{\mathcal{T}}$ *is the oblivious sampling algorithm which takes a "punctured point" tag* tag^* *as input, and outputs an "obliviously generated" public key/ciphertext pair* (pk, c) *and a punctured secret key* $\widehat{sk}_{\mathsf{tag}^*}$.
- $\mathsf{rSamp}_{\mathcal{T}}$ *is the inverting algorithm (corresponding to* $\mathsf{oSamp}_{\mathcal{T}}$*) that takes* 1^k*, randomness* r_g *and* r_e*, a "punctured point" tag* tag^**, and a plaintext* m *(which are supposed to be used as* $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r_g)$ *and* $c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, m; r_e)$*) as input, and outputs a string* $\widehat{r}$ *(that looks like a randomness used by* $\mathsf{oSamp}_{\mathcal{T}}$*).*
- ***(Trapdoor Simulatability)*** *For all PPTAs* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathsf{Adv}^{\mathtt{TSPTBE}}_{\mathcal{T},\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k) = 1]|$ *is negligible, where the experiments* $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Real}}_{\mathcal{T},\mathcal{A}}(k)$ *and* $\mathsf{Expt}^{\mathtt{TSPTBE\text{-}Sim}}_{\mathcal{T},\mathcal{A}}(k)$ *are defined as in Fig. 1 (bottom-left and bottom-right, respectively).*

On the Existence of TSPTBE. Though it might look complicated, we can construct a TSPTBE scheme from a TSPKE scheme, by a Dolev-Dwork-Naor-style approach [23]. The construction is exactly the same as the construction of a PTBE scheme from any CPA secure PKE shown in [39], which is the "core" structure of the DDN construction, namely, the DDN construction without a NIZK proof and without its one-time signature. (For this construction, we can straightforwardly consider the oblivious sampling algorithm and the corresponding inverting algorithm). We prove the following lemma in the full version.

Lemma 1. *If a TSPKE scheme exists, then so does a TSPTBE scheme.*

Useful Fact. For the security proofs of our constructions in Sect. 4, we will use the fact that the straightforward concatenation of a "transcript" of a trapdoor simulatable commitment and that of a TSPTBE scheme, also admits the trapdoor simulatable property.

More formally, for a TSPTBE scheme $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}}, \mathsf{oSamp}_{\mathcal{T}}, \mathsf{rSamp}_{\mathcal{T}})$ and a trapdoor simulatable commitment scheme $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com}, \mathsf{oSamp}_{\mathcal{C}}, \mathsf{rSamp}_{\mathcal{C}})$ such that the plaintext space of $\mathcal{T}$ and that of $\mathcal{C}$ are identical, and for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, consider the following "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ and the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$ as described in Fig. 2 (left and right, respectively).

$\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$:
$(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
$r_g, r'_g, r_c, r_t \leftarrow \{0,1\}^*$
$ck \leftarrow \mathsf{CKG}(1^k; r_g)$
$\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, m; r_c)$
$\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r_c, m)$
$(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$
$c^* \leftarrow \mathsf{TEnc}(ck, \mathsf{tag}^*, m; r_e)$
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$
$\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r_t, \mathsf{tag}^*, m)$
$b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$
Return b'.

$\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k)$:
$(m, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k)$
$\widehat{r}_c, \widehat{r}_t \leftarrow \{0,1\}^*$
$(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$
$(pk, c^*, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$
$b' \leftarrow \mathcal{A}_2(\mathsf{st}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$
Return b'.

Fig. 2. Security experiments for defining the trapdoor simulatability of the concatenation of a "transcript" of a commitment scheme and that of a TSPTBE scheme.

Then, we can prove the following lemma, whose proof is almost straightforward due to the trapdoor simulatability property of $\mathcal{C}$ and $\mathcal{T}$. The proof is by a standard hybrid argument, and is given in the full version.

Lemma 2. *Assume that the commitment scheme $\mathcal{C}$ and the PTBE scheme $\mathcal{T}$ are trapdoor simulatable. Then, for all PPTAs $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) := |\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{A}}(k) = 1]|$ is negligible.*

4 Proposed KEMs

In this section, we show our main results: two KEMs that show the "trade-off" between the strength of (standard model) plaintext awareness and the simulatability property with those of the construction by Dachman-Soled [18].

In Sect. 4.1, we show our first construction, which is $\mathtt{CCA}$ secure based on a KEM satisfying $\mathtt{CPA}$ security and $\mathtt{sPA1}_2$ security, and a TSPKE scheme. In Sect. 4.2, we show our second construction which is $\mathtt{CCA}$ secure based on a KEM satisfying $1\text{-}\mathtt{CCA}$ security and $\mathtt{sPA1}_1$ security, and a TSPKE scheme.

4.1 First Construction

Let $\Gamma_{\mathtt{in}} = (\mathsf{KKG}_{\mathtt{in}}, \mathsf{Encap}_{\mathtt{in}}, \mathsf{Decap}_{\mathtt{in}})$ be a KEM whose ciphertext length is $n = n(k)$ and whose session-key space is $\{0,1\}^{3k}$ for k-bit security.[5] Let $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ be a PTBE scheme and $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$ be a commitment scheme. We require the plaintext space of TEnc and the message space of Com to be $\{0,1\}^{2n}$, and the randomness space of TEnc and

[5] Note that the session-key space of a KEM can be adjusted "for free" by applying a pseudorandom generator to a session-key. Such a construction preserves $\mathtt{CPA}$ and $\mathtt{sPA1}_\ell$ security.

that of Com to be $\{0,1\}^k$ for k-bit security.[6] Then, our first proposed KEM $\Gamma = (\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ is constructed as in Fig. 3.

$\mathsf{KKG}(1^k)$:
$(pk_{\mathtt{in}0}, sk_{\mathtt{in}0}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$
$(pk_{\mathtt{in}1}, sk_{\mathtt{in}1}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$
$(pk, sk) \leftarrow \mathsf{TKG}(1^k)$
$ck \leftarrow \mathsf{CKG}(1^k)$
$PK \leftarrow (pk_{\mathtt{in}0}, pk_{\mathtt{in}1}, pk, ck)$
$SK \leftarrow (sk_{\mathtt{in}0}, sk_{\mathtt{in}1}, sk, PK)$
Return (PK, SK).

$\mathsf{Encap}(PK)$:
$(pk_{\mathtt{in}0}, pk_{\mathtt{in}1}, pk, ck) \leftarrow PK$
$(c_{\mathtt{in}0}, \alpha_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}0})$
$(c_{\mathtt{in}1}, \alpha_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}1})$
$\alpha \leftarrow \alpha_0 \oplus \alpha_1$
Parse α as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
$\mathsf{tag} \leftarrow \mathsf{Com}(ck, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_c)$
$c \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_t)$
$C \leftarrow (\mathsf{tag}, c)$.
Return (C, K).

$\mathsf{Decap}(SK, C)$:
$(sk_{\mathtt{in}0}, sk_{\mathtt{in}1}, sk, PK) \leftarrow SK$
$(pk_{\mathtt{in}0}, pk_{\mathtt{in}1}, pk, ck) \leftarrow PK$
$(\mathsf{tag}, c) \leftarrow C$
$(c_{\mathtt{in}0} \| c_{\mathtt{in}1}) \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$
If TDec has returned $\perp$ then return $\perp$.
$\alpha_0 \leftarrow \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}0}, c_{\mathtt{in}0})$
$\alpha_1 \leftarrow \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}1}, c_{\mathtt{in}1})$
If $\alpha_0 = \perp$ or $\alpha_1 = \perp$ then return $\perp$.
$\alpha \leftarrow \alpha_0 \oplus \alpha_1$
Parse α as $(r_c, r_t, K) \in (\{0,1\}^k)^3$
If $\mathsf{Com}(ck, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_c) = \mathsf{tag}$
and $\mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_t) = c$
then return K else return $\perp$.

Fig. 3. The first proposed construction: the KEM Γ based on a KEM $\Gamma_{\mathtt{in}}$, a commitment scheme $\mathcal{C}$, and a PTBE scheme $\mathcal{T}$.

Alternative Decapsulation Algorithm. Similarly to the constructions in [37–39], to show the $\mathtt{CCA}$ security of the proposed KEM Γ, it is useful to consider the following alternative decapsulation algorithm $\mathsf{AltDecap}$. For a k-bit string $\mathsf{tag}^* \in \{0,1\}^k$ and a key pair (PK, SK) output by $\mathsf{KKG}(1^k)$, where $PK = (pk_{\mathtt{in}0}, pk_{\mathtt{in}1}, pk, ck)$ and $SK = (sk_{\mathtt{in}0}, sk_{\mathtt{in}1}, sk, PK)$, we define an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ associated with $\mathsf{tag}^* \in \{0,1\}^k$ by $\widehat{SK}_{\mathsf{tag}^*} = (sk_{\mathtt{in}0}, sk_{\mathtt{in}1}, \mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, PK)$, where $\widehat{sk}_{\mathsf{tag}^*} = \mathsf{Punc}(sk, \mathsf{tag}^*)$. $\mathsf{AltDecap}$ takes an "alternative" secret key $\widehat{SK}_{\mathsf{tag}^*}$ defined as above and a ciphertext $C = (\mathsf{tag}, c)$ as input, and runs as follows:

$\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$**:** First check if $\mathsf{tag}^* = \mathsf{tag}$, and return $\perp$ if this is the case. Otherwise, run in exactly the same way as $\mathsf{Decap}(SK, C)$, except that "$(c_{\mathtt{in}0} \| c_{\mathtt{in}1}) \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$" is executed in the fourth step, instead of "$(c_{\mathtt{in}0} \| c_{\mathtt{in}1}) \leftarrow \mathsf{TDec}(sk, \mathsf{tag}, c)$."

[6] The requirements of the randomness space of TEnc and Com are without loss of generality, because we can adjust them using a pseudorandom generator. (The trapdoor simulatability property is preserved even if we use a pseudorandom generator).

Regarding $\mathsf{AltDecap}$, the following lemma is easy to see due to the correctness of the underlying PTBE scheme $\mathcal{T}$ and the validity check of c by re-encryption performed at the last step. (The formal proof is given in the full version).

Lemma 3. *Let* $\mathsf{tag}^* \in \{0,1\}^k$ *be a string and let* (PK, SK) *be a key pair output by* $\mathsf{KKG}(1^k)$*. Furthermore, let* $\widehat{SK}_{\mathsf{tag}^*}$ *be an alternative secret key as defined above. Then, for any ciphertext* $C = (\mathsf{tag}, c)$ *(which could be outside the range of* $\mathsf{Encap}(PK)$*) satisfying* $\mathsf{tag} \neq \mathsf{tag}^*$*, it holds that* $\mathsf{Decap}(SK, C) = \mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$.

$\mathtt{CCA}$ *Security.* The security of Γ is guaranteed by the following theorem.

Theorem 1. *Assume that the KEM* $\Gamma_{\mathtt{in}}$ *is* $\mathtt{CPA}$ *secure and* $\mathtt{sPA1}_2$ *secure, the commitment scheme* $\mathcal{C}$ *is target-binding and trapdoor simulatable, and the PTBE scheme* $\mathcal{T}$ *is trapdoor simulatable. Then, the KEM* Γ *constructed as in Fig. 3 is* $\mathtt{CCA}$ *secure.*

Note that as mentioned in Sect. 2.3, a commitment scheme with trapdoor simulatability and target-binding can be constructed from any TSPKE scheme, and thus the above theorem shows that we can indeed construct a $\mathtt{CCA}$ secure KEM (and thus $\mathtt{CCA}$ secure PKE) from the combination of a KEM satisfying $\mathtt{CPA}$ and $\mathtt{sPA1}_2$ security and a TSPKE scheme.

We have provided ideas for the security proof in Sect. 1.3, and thus we directly proceed to the proof.

Proof of Theorem 1. Let $\mathcal{A}$ be any PPTA adversary that attacks the $\mathtt{CCA}$ security of the KEM Γ. Our security proof is via the sequence of games argument. To describe the games, we will need an extractor $\mathcal{E}$ corresponding to some "ciphertext creator" $\mathcal{A}'$ that is guaranteed to exist by the $\mathtt{sPA1}_2$ security of $\Gamma_{\mathtt{in}}$. Specifically, consider the following $\mathcal{A}'$ (that internally runs $\mathcal{A}$) that runs in the experiment $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$, with a corresponding extractor $\mathcal{E}$:

$\mathcal{A}'^{\mathcal{E}(\mathsf{st}_{\mathcal{E}},\cdot)}(pk_1, pk_2; r_{\mathcal{A}'} = (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*))$**:** $\mathcal{A}'$ firstly sets $pk_{\mathtt{in0}} \leftarrow pk_1$ and $pk_{\mathtt{in1}} \leftarrow pk_2$ (which implicitly sets $sk_{\mathtt{in0}} \leftarrow sk_1$ and $sk_{\mathtt{in1}} \leftarrow sk_2$, where sk_1 (resp. sk_2) is the secret key corresponding to pk_1 (resp. pk_2)), and runs $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*}) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$. Then $\mathcal{A}'$ sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $C^* \leftarrow (\mathsf{tag}^*, c^*)$, and then runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$. When $\mathcal{A}$ submits a decapsulation query C, $\mathcal{A}'$ responds to it as if it runs $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$, where the oracle calls (to the extractor $\mathcal{E}$) of the form $(1, c_{\mathtt{in0}})$ and $(2, c_{\mathtt{in1}})$ are used as substitutes for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}})$ and $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}})$, respectively. More precisely, $\mathcal{A}'$ answers $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ as follows:

1. If $\mathsf{tag} = \mathsf{tag}^*$, then return $\perp$ to $\mathcal{A}$.
2. Run $(c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \leftarrow \widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c)$, and return $\perp$ to $\mathcal{A}$ if $\widehat{\mathsf{TDec}}$ has returned $\perp$.
3. Submit queries $(1, c_{\mathtt{in0}})$ and $(2, c_{\mathtt{in1}})$ to the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$ and receive the answers α_0 and α_1, respectively. (Here, the answers α_0 and α_1 are expected to be $\alpha_0 = \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}})$ and $\alpha_1 = \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, c_{\mathtt{in1}})$, respectively, and the extractor $\mathcal{E}$ may update its state upon each call).

4. If $\alpha_0 = \bot$ or $\alpha_1 = \bot$, then return $\bot$ to $\mathcal{A}$.
5. Let $\alpha \leftarrow \alpha_0 \oplus \alpha_1$ and parse α as $(r_c, r_t, K) \in (\{0,1\}^k)^3$.
6. If $\mathsf{Com}(ck, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_c) = \mathsf{tag}$ and $\mathsf{TEnc}(pk, (c_{\mathtt{in0}} \| c_{\mathtt{in1}}); r_t) = c$, then return K, otherwise return $\bot$, to $\mathcal{A}$.
   When $\mathcal{A}$ terminates, $\mathcal{A}'$ also terminates.

The above completes the description of the algorithm $\mathcal{A}'$. The randomness $r_{\mathcal{A}'}$ consumed by $\mathcal{A}'$ is of the form $(r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$, where $r_{\mathcal{A}}$, $\widehat{r}_c$, and $\widehat{r}_t$ are the randomness used by $\mathcal{A}$, $\mathsf{oSamp}_{\mathcal{C}}$, and $\mathsf{oSamp}_{\mathcal{T}}$, respectively, and K^* is a k-bit string. The corresponding extractor $\mathcal{E}$ thus receives (pk_1, pk_2) and $r_{\mathcal{A}'}$ as its initial state $\mathsf{st}_{\mathcal{E}}$. Note that since $\Gamma_{\mathtt{in}}$ is assumed to be $\mathtt{sPA1}_2$ secure and $\mathcal{A}'$ is a PPTA, $\mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ is negligible for this extractor $\mathcal{E}$, which will be used later in the proof. (Looking ahead, we will design the sequence of games so that $\mathcal{A}$'s view in the case $\mathcal{A}$ is internally run by $\mathcal{A}'$ and $\mathcal{A}'$ is run in $\mathsf{Expt}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$, is identical to $\mathcal{A}$'s view in Game 6).

For convenience, we refer to the procedure of using the extractor $\mathcal{E}$ as substitutes for $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, \cdot)$ and $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in1}}, \cdot)$, as $\mathsf{AltDecap}'_{\mathcal{E}}$. Here, $\mathsf{AltDecap}'_{\mathcal{E}}$ is a stateful procedure that initially takes tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and an initial state $\mathsf{st}_{\mathcal{E}}$ of $\mathcal{E}$ (i.e. $\mathsf{st}_{\mathcal{E}} = ((pk_{\mathtt{in0}}, pk_{\mathtt{in1}}), r_{\mathcal{A}'})$) as input, and expects to receive a ciphertext $C = (\mathsf{tag}, c)$ as an input. If it receives a ciphertext $C = (\mathsf{tag}, c)$, it calculates the decapsulation result K (or $\bot$) as $\mathcal{A}'$ does for $\mathcal{A}$, using $\widehat{sk}_{\mathsf{tag}^*}$ and the extractor $\mathcal{E}$, where $\mathcal{E}$'s internal state could be updated upon each execution.

Now, using the adversary $\mathcal{A}$ and the extractor $\mathcal{E}$, consider the following sequence of games: (Here, the values with asterisk (*) represent those related to the challenge ciphertext for $\mathcal{A}$).

Game 1: This is the experiment $\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ itself.

Game 2: Same as Game 1, except that all decapsulation queries $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ are answered with $\bot$.

Game 3: Same as Game 2, except that all decapsulation queries C are answered with $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$, where $\widehat{SK}_{\mathsf{tag}^*}$ is the alternative secret key corresponding to (PK, SK) and tag^*. Furthermore, we pick a random bit $\gamma \in \{0,1\}$ uniformly at random just before executing $\mathcal{A}$, which will be used to define the events in this game and the subsequent games. (γ does not appear in $\mathcal{A}$'s view in this and all subsequent games, and thus does not affect its behavior at all).

Game 4: In this game, we use $\mathsf{AltDecap}'_{\mathcal{E}}$ (defined as above) as $\mathcal{A}$'s decapsulation oracle, where the initial state of $\mathcal{E}$ (used internally by $\mathsf{AltDecap}'_{\mathcal{E}}$) is prepared using the "inverting algorithms" $\mathsf{rSamp}_{\mathcal{C}}$ of $\mathcal{C}$ and $\mathsf{rSamp}_{\mathcal{T}}$ of $\mathcal{T}$. Moreover, we also change the ordering of the steps so that they do not affect $\mathcal{A}$'s view. More precisely, this game is defined as follows:

Game 4:
$(pk_{\mathsf{in}0}, sk_{\mathsf{in}0}) \leftarrow \mathsf{KKG}_{\mathsf{in}}(1^k)$;
$(pk_{\mathsf{in}1}, sk_{\mathsf{in}1}) \leftarrow \mathsf{KKG}_{\mathsf{in}}(1^k)$;
$(c^*_{\mathsf{in}0}, \alpha^*_0) \leftarrow \mathsf{Encap}_{\mathsf{in}}(pk_{\mathsf{in}0})$;
$(c^*_{\mathsf{in}1}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathsf{in}}(pk_{\mathsf{in}1})$;
$\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$;
Parse α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$;
$r_g \leftarrow \{0,1\}^*$;
$ck \leftarrow \mathsf{CKG}(1^k; r_g)$;
$\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}); r^*_c)$;
$\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}))$;
(Continue to the right column ↗)

$r'_g \leftarrow \{0,1\}^*$;
$(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$;
$\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$;
$c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}); r^*_t)$;
$\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}))$;
$PK \leftarrow (pk_{\mathsf{in}0}, pk_{\mathsf{in}1}, pk, ck)$;
$C^* \leftarrow (\mathsf{tag}^*, c^*)$;
$K^*_0 \leftarrow \{0,1\}^k$;
$b \leftarrow \{0,1\}$;
$r_{\mathcal{A}} \leftarrow \{0,1\}^*$;
$r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_b)$;
$\mathsf{st}_{\mathcal{E}} \leftarrow ((pk_{\mathsf{in}0}, pk_{\mathsf{in}1}), r_{\mathcal{A}'})$;
$\gamma \leftarrow \{0,1\}$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(PK, C^*, K^*_b; r_{\mathcal{A}})$

where the decapsulation oracle $\mathcal{O}$ that $\mathcal{A}$ has access in Game 4 is $\mathsf{AltDecap}'_{\mathcal{E}}$ (which initially receives $\mathsf{tag}^*, \widehat{sk}_{\mathsf{tag}^*}, \mathsf{st}_{\mathcal{E}} = (pk_{\mathsf{in}0}, pk_{\mathsf{in}1}, r_{\mathcal{A}'})$ as input). Note that the extractor $\mathcal{E}$ used internally by $\mathsf{AltDecap}'_{\mathcal{E}}$ may update its state $\mathsf{st}_{\mathcal{E}}$ upon each execution.

Game 5: Same as Game 4, except that $r^*_c, r^*_t, K^*_1 \in \{0,1\}^k$ are picked uniformly at random, independently of $\alpha^* = \alpha^*_0 \oplus \alpha^*_1$. That is, the steps "$\alpha^* \leftarrow \alpha^*_0 \oplus \alpha^*_1$; Parse α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$" in Game 4 are replaced with the step "$r^*_c, r^*_t, K^*_1 \leftarrow \{0,1\}^k$," and we do not use α^* anymore.

Game 6: Same as Game 5, except that the key/commitment pair (ck, tag^*) and the key/ciphertext pair (pk, c^*) and a punctured secret key $\widehat{sk}_{\mathsf{tag}^*}$ are sampled obliviously, and correspondingly the randomness $\widehat{r}_c$ and $\widehat{r}_t$ used for oblivious sampling are used in $r_{\mathcal{A}'}$.
More precisely, the steps "$r_g, r^*_c \leftarrow \{0,1\}^*$; $ck \leftarrow \mathsf{CKG}(1^k; r_g)$; $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}); r^*_c)$; $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}))$" in Game 5 are replaced with the steps "$\widehat{r}_c \leftarrow \{0,1\}^*$; $(ck, \mathsf{tag}^*) \leftarrow \mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$".
Furthermore, the steps "$r'_g, r^*_t \leftarrow \{0,1\}^k$; $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$; $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}); r^*_t)$; $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathsf{in}0} \| c^*_{\mathsf{in}1}))$" in Game 5 are replaced with the steps "$\widehat{r}_t \leftarrow \{0,1\}^*$; $(pk, \widehat{sk}_{\mathsf{tag}^*}, c^*) \leftarrow \mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$".

The above completes the description of the games.

For $i \in [5]$, let Succ_i denote the event that $\mathcal{A}$ succeeds in guessing the challenge bit (i.e. $b' = b$ occurs) in Game i. Furthermore, for $i \in \{3, \ldots, 6\}$, we define the following *bad* events in Game i:

Bad_i**:** $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ satisfying the following conditions simultaneously: (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = (c_{\mathsf{in}0} \| c_{\mathsf{in}1}) \neq \perp$, and (3) $\mathsf{Decap}_{\mathsf{in}}(sk_{\mathsf{in}0}, c_{\mathsf{in}0}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (1, c_{\mathsf{in}0}))$ *or* $\mathsf{Decap}_{\mathsf{in}}(sk_{\mathsf{in}1}, c_{\mathsf{in}1}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (2, c_{\mathsf{in}1}))$.

$\mathsf{Bad}^{(\sigma)}_i$**:** (where $\sigma \in \{0,1\}$) $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the same conditions as Bad_i, except that the condition (3) is replaced with the condition: $\mathsf{Decap}_{\mathsf{in}}(sk_{\mathsf{in}\sigma}, c_{\mathsf{in}\sigma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\sigma + 1, c_{\mathsf{in}\sigma}))$.

Bad_i^*: $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the same conditions as Bad_i, except that the condition (3) is replaced with the condition: $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\gamma + 1, c_{\mathtt{in}\gamma}))$ (where γ is the random bit chosen just before executing $\mathcal{A}$).

Note that for all $i \in \{3, \ldots, 6\}$, the events $\mathsf{Bad}_i^{(0)}$, $\mathsf{Bad}_i^{(1)}$, and Bad_i^* all imply the event Bad_i, and thus we have $\Pr[\mathsf{Bad}_i^{(0)}], \Pr[\mathsf{Bad}_i^{(1)}], \Pr[\mathsf{Bad}_i^*] \leq \Pr[\mathsf{Bad}_i]$.

By the definitions of the games and events, we have

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) &= 2 \cdot \left|\Pr[\mathsf{Succ}_1] - \frac{1}{2}\right| \\ &\leq 2 \cdot \Big(\sum_{i \in [4]} \left|\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]\right| + \left|\Pr[\mathsf{Succ}_5] - \frac{1}{2}\right| \Big). \quad (1)\end{aligned}$$

In the following, we will upperbound each term that appears in the right hand side of the above inequality.

Claim 1. *There exists a PPTA* $\mathcal{B}_{\mathsf{b}}$ *such that* $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) \geq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$.

Proof of Claim 1. For $i \in \{1, 2\}$, let NoBind_i be the event that in Game i, $\mathcal{A}$ submits at least one decapsulation query $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ and $\mathsf{Decap}(SK, C) \neq \bot$. Recall that $\mathcal{A}$'s query C must satisfy $C \neq C^* = (\mathsf{tag}^*, c^*)$, and thus $\mathsf{tag} = \mathsf{tag}^*$ implies $c \neq c^*$. The difference between Game 1 and Game 2 is how $\mathcal{A}$'s decapsulation query $C = (\mathsf{tag}, c)$ satisfying $\mathsf{tag} = \mathsf{tag}^*$ is answered. Hence, these games proceed identically unless NoBind_1 or NoBind_2 occurs in the corresponding games, and thus we have

$$\left|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]\right| \leq \Pr[\mathsf{NoBind}_1] = \Pr[\mathsf{NoBind}_2]. \quad (2)$$

Thus, it is sufficient to upperbound $\Pr[\mathsf{NoBind}_2]$.

Observe that for a decapsulation query $C = (\mathsf{tag}^*, c)$ satisfying the condition of NoBind_2, it is guaranteed that $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c_{\mathtt{in}0} \| c_{\mathtt{in}1}) \neq (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1})$. Indeed, if $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1})$ and $\mathsf{Decap}(SK, C) \neq \bot$, then by the validity check of c in Decap, we have $c^* = c$, which is because c must satisfy $\mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1}); r_t^*) = c$ where r_t^* is the $(k+1)$-to-$2k$-th bits of $\alpha^* = (\alpha_0^* \oplus \alpha_1^*) = (\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}0}, c^*_{\mathtt{in}0}) \oplus \mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}1}, c^*_{\mathtt{in}1}))$. However, $\mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1}); r_t^*) = c^*$ also holds due to how c^* is generated, and thus contradicting the condition $c \neq c^*$ implied by NoBind_2.

We use the above fact to show how to construct a PPTA adversary $\mathcal{B}_{\mathsf{b}}$ that attacks the target-binding property of the commitment scheme $\mathcal{C}$ with advantage $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) = \Pr[\mathsf{NoBind}_2]$. The description of $\mathcal{B}_{\mathsf{b}} = (\mathcal{B}_{\mathsf{b}1}, \mathcal{B}_{\mathsf{b}2})$ is as follows:

$\mathcal{B}_{\mathsf{b}1}(1^k)$: $\mathcal{B}_{\mathsf{b}1}$ first runs $(pk_{\mathtt{in}0}, sk_{\mathtt{in}0}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(pk_{\mathtt{in}1}, sk_{\mathtt{in}1}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(c^*_{\mathtt{in}0}, \alpha_0^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}0})$, and $(c^*_{\mathtt{in}1}, \alpha_1^*) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in}1})$. $\mathcal{B}_{\mathsf{b}1}$ then sets $\alpha^* \leftarrow (\alpha_0^* \oplus \alpha_1^*)$, and parses α^* as $(r_c^*, r_t^*, \alpha^*) \in (\{0,1\}^k)^3$. Finally, $\mathcal{B}_{\mathsf{b}1}$ sets $M \leftarrow (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1})$, $R \leftarrow r_c^*$, and $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{b}1}$'s entire view), and terminates with output $(M, R, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{b}2}(\mathsf{st}_{\mathcal{B}}, ck)$**:** $\mathcal{B}_{\mathsf{b}2}$ first runs $(pk, sk) \leftarrow \mathsf{TKG}(1^k)$, and then sets $PK \leftarrow (pk_{\mathtt{in}0}, pk_{\mathtt{in}1}, pk, ck)$ and $SK \leftarrow (sk_{\mathtt{in}0}, sk_{\mathtt{in}1}, sk, PK)$. $\mathcal{B}_{\mathsf{b}2}$ next runs $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1}); r^*_c)$ and $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1}); r^*_t)$, sets $C^* \leftarrow (\mathsf{tag}^*, c^*)$, and also chooses $K^*_0 \in \{0,1\}^k$ and $b \in \{0,1\}$ uniformly at random. Then, $\mathcal{B}_{\mathsf{b}2}$ runs $\mathcal{A}$, where the decapsulation queries from $\mathcal{A}$ are answered as Game 2 does, which is possible because $\mathcal{B}_{\mathsf{b}2}$ possesses SK.
When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{b}2}$ checks if $\mathcal{A}$ has made a decapsulation query $C = (\mathsf{tag}, c)$ satisfying the conditions of NoBind_2, namely, $\mathsf{tag} = \mathsf{tag}^*$, $c \neq c^*$, $\mathsf{TDec}(sk, \mathsf{tag}, c) = (c_{\mathtt{in}0} \| c_{\mathtt{in}1}) \notin \{(c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1}), \perp\}$, $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}0}, c_{\mathtt{in}0}) = \alpha_0 \neq \perp$, $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}1}, c_{\mathtt{in}1}) = \alpha_1 \neq \perp$, $(\alpha_0 \oplus \alpha_1) = (r_c \| r_t \| K) \in \{0,1\}^{3k}$, and $\mathsf{Com}(ck, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_c) = \mathsf{tag}^*$, and $\mathsf{TEnc}(pk, \mathsf{tag}, (c_{\mathtt{in}0} \| c_{\mathtt{in}1}); r_t) = c$. (Actually, the last condition is redundant for $\mathcal{B}_{\mathsf{b}2}$'s purpose). If such a query is found, then $\mathcal{B}_{\mathsf{b}2}$ terminates with output $M' = (c_{\mathtt{in}0} \| c_{\mathtt{in}1})$ and $R' = r_c$. Otherwise, $\mathcal{B}_{\mathsf{b}2}$ gives up and aborts.

The above completes the description of $\mathcal{B}_{\mathsf{b}}$. It is easy to see that $\mathcal{B}_{\mathsf{b}}$ does a perfect simulation of Game 2 for $\mathcal{A}$, and whenever $\mathcal{A}$ makes a query that causes the event NoBind_2, $\mathcal{B}_{\mathsf{b}2}$ can find such a query by using SK and output a pair $(M', R') = ((c_{\mathtt{in}0} \| c_{\mathtt{in}1}), r_c)$ satisfying $\mathsf{Com}(ck, M; R) = \mathsf{Com}(ck, M'; R') = \mathsf{tag}^*$ and $M \neq M'$, violating the target-binding property of the commitment scheme $\mathcal{C}$. Therefore, we have $\mathsf{Adv}^{\mathsf{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) = \Pr[\mathsf{NoBind}_2]$. Then, by Eq. (2), we have $\mathsf{Adv}^{\mathsf{TBind}}_{\mathcal{C},\mathcal{B}_{\mathsf{b}}}(k) \geq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$, as required. □ (Claim 1)

Claim 2. $\Pr[\mathsf{Succ}_2] = \Pr[\mathsf{Succ}_3]$.

Proof of Claim 2. It is sufficient to show that the behavior of the oracle given to $\mathcal{A}$ in Game 2 and that in Game 3 are identical. Let $C = (\mathsf{tag}, c)$ be a decapsulation query that $\mathcal{A}$ makes. If $\mathsf{tag} = \mathsf{tag}^*$, then the query is answered with $\perp$ in Game 2 by definition, while the oracle $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ that is given access to $\mathcal{A}$ in Game 3 also returns $\perp$ by definition. Otherwise (i.e. $\mathsf{tag} \neq \mathsf{tag}^*$), by Lemma 3, the result of $\mathsf{Decap}(SK, C)$ and that of $\mathsf{AltDecap}(\widehat{SK}_{\mathsf{tag}^*}, C)$ always agree. This completes the proof. □ (Claim 2)

Claim 3. *There exist PPTAs* $\mathcal{B}_{\mathsf{g}}$ *and* $\mathcal{B}_{\mathsf{d}}$ *such that*

$$\left|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]\right| \leq 2 \cdot \left(\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathsf{g}}}(k) + \mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) + \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)\right).$$

We postpone the proof of this claim to the end of the proof of Theorem 1.

Claim 4. *There exists a PPTA* $\mathcal{B}'_{\mathsf{g}}$ *such that* $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathsf{g}}}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$.

Proof of Claim 4. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathtt{CPA}$ adversary $\mathcal{B}'_{\mathsf{g}}$ with the claimed advantage. The description of $\mathcal{B}'_{\mathsf{g}}$ is as follows:

$\mathcal{B}'_{\mathsf{g}}(pk', c'^*, \alpha'^*_\beta)$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}'_{\mathsf{g}}$'s challenge bit in its $\mathtt{CPA}$ experiment) $\mathcal{B}'_{\mathsf{g}}$ sets $pk_{\mathsf{in0}} \leftarrow pk'$, $c^*_{\mathsf{in0}} \leftarrow c'^*$, and $\alpha^*_0 \leftarrow \alpha'^*_\beta$. Next, $\mathcal{B}'_{\mathsf{g}}$ generates $(pk_{\mathsf{in1}}, sk_{\mathsf{in1}}) \leftarrow \mathsf{KKG}_{\mathsf{in}}(1^k)$ and $(c^*_{\mathsf{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathsf{in}}(pk_{\mathsf{in1}})$, sets $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$, and parses α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$. Then, $\mathcal{B}'_{\mathsf{g}}$ picks $r_g, r'_g \leftarrow \{0,1\}^*$ uniformly at random, and runs $ck \leftarrow \mathsf{CKG}(1^k; r_g)$, $\mathsf{tag}^* \leftarrow \mathsf{Com}(ck, (c^*_{\mathsf{in0}} \| c^*_{\mathsf{in1}}); r^*_c)$, $\widehat{r}_c \leftarrow \mathsf{rSamp}_{\mathcal{C}}(r_g, r^*_c, (c^*_{\mathsf{in0}} \| c^*_{\mathsf{in1}}))$, $(pk, sk) \leftarrow \mathsf{TKG}(1^k; r'_g)$, $\widehat{sk}_{\mathsf{tag}^*} \leftarrow \mathsf{Punc}(sk, \mathsf{tag}^*)$, $c^* \leftarrow \mathsf{TEnc}(pk, \mathsf{tag}^*, (c^*_{\mathsf{in0}} \| c^*_{\mathsf{in1}}); r^*_t)$, and $\widehat{r}_t \leftarrow \mathsf{rSamp}_{\mathcal{T}}(r'_g, r^*_t, \mathsf{tag}^*, (c^*_{\mathsf{in0}} \| c^*_{\mathsf{in1}}))$. Then $\mathcal{B}'_{\mathsf{g}}$ picks $r_{\mathcal{A}} \in \{0,1\}^*$, $K^*_0 \in \{0,1\}^k$, and $b \in \{0,1\}$ all uniformly at random, and sets $PK \leftarrow (pk_{\mathsf{in0}}, pk_{\mathsf{in1}}, pk, ck)$, $C^* \leftarrow (\mathsf{tag}^*, c^*)$, $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*_b)$, and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathsf{in0}}, pk_{\mathsf{in1}}, r_{\mathcal{A}'})$. Finally, $\mathcal{B}'_{\mathsf{g}}$ runs $\mathcal{A}(PK, C^*, K^*_b; r_{\mathcal{A}})$.

$\mathcal{B}'_{\mathsf{g}}$ answers $\mathcal{A}$'s decapsulation queries as $\mathsf{AltDecap}'_{\mathcal{E}}$ does, where the initial state of $\mathsf{AltDecap}'_{\mathcal{E}}$ is tag^*, $\widehat{sk}_{\mathsf{tag}^*}$, and $\mathsf{st}_{\mathcal{E}}$. (Note that $\mathsf{st}_{\mathcal{E}}$ is used by $\mathcal{E}$, and may be updated upon each call of $\mathsf{AltDecap}'_{\mathcal{E}}$).

When $\mathcal{A}$ terminates with output b', $\mathcal{B}'_{\mathsf{g}}$ sets $\beta' \leftarrow (b' \stackrel{?}{=} b)$, and terminates with output β'.

The above completes the description of $\mathcal{B}'_{\mathsf{g}}$. $\mathcal{B}'_{\mathsf{g}}$'s $\mathtt{CPA}$ advantage can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathsf{in}},\mathcal{B}'_{\mathsf{g}}}(k) &= 2 \cdot \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| = \left|\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]\right| \\ &= \left|\Pr[b' = b | \beta = 1] - \Pr[b' = b | \beta = 0]\right|.\end{aligned}$$

Consider the case when $\beta = 1$. It is easy to see that in this case, $\mathcal{B}'_{\mathsf{g}}$ simulates Game 4 perfectly for $\mathcal{A}$. Specifically, the real session-key $\alpha'^*_\beta = \alpha'^*_1$ (corresponding to $c^*_{\mathsf{in0}} = c'^*$) is used as α^*_0, and thus $\alpha^* = (\alpha^*_0 \oplus \alpha^*_1) = (r^*_c \| r^*_t \| K^*_1)$ is generated exactly as that in Game 4. All other values are distributed identically to those in Game 4. Furthermore, $\mathcal{B}'_{\mathsf{g}}$ uses $\mathsf{AltDecap}'_{\mathcal{E}}$ for answering $\mathcal{A}$'s decapsulation queries, where the initial state of $\mathsf{AltDecap}'_{\mathcal{E}}$ (and thus the initial state of $\mathcal{E}$) is appropriately generated as those in Game 4. Under this situation, the probability that $\mathcal{A}$ succeeds in guessing b (i.e. $b' = b$ occurs) is exactly the same as the probability that $\mathcal{A}$ does so in Game 4, i.e. $\Pr[b' = b | \beta = 1] = \Pr[\mathsf{Succ}_4]$.

On the other hand, when $\beta = 0$, then $\mathcal{B}'_{\mathsf{g}}$ simulates Game 5 perfectly for $\mathcal{A}$. Specifically, in this case, a uniformly random value $\alpha'^*_\beta = \alpha'^*_0$ is used as α^*_0. Therefore, $\alpha^* = (\alpha^*_0 \oplus \alpha^*_1)$ is also a uniformly random $3k$-bit string, and thus each of r^*_c, r^*_t, and K^*_1 is a uniformly random k-bit string, which is exactly how these values are chosen in Game 5. Since this is the only change from the case of $\beta = 1$, with a similar argument to the above, we have $\Pr[b' = b | \beta = 0] = \Pr[\mathsf{Succ}_5]$.

In summary, we have $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathsf{in}},\mathcal{B}'_{\mathsf{g}}}(k) = |\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_5]|$, as required. □ (Claim 4)

Claim 5. $\Pr[\mathsf{Succ}_5] = 1/2$.

Proof of Claim 5. This is obvious because in Game 5, the real session-key K^*_1 is made independent of the challenge ciphertext C^*. Since both K^*_1 and K^*_0 are

now uniformly random, the view of $\mathcal{A}$ does not contain any information on b. This means that the probability that $\mathcal{A}$ succeeds in guessing the challenge bit is exactly $1/2$. □ (Claim 5)

Claims 1, 2, 3, 4 and 5 and Eq. (1) guarantee that there exist PPTAs $\mathcal{B}_{\mathtt{b}}$, $\mathcal{B}_{\mathtt{g}}$, $\mathcal{B}_{\mathtt{d}}$, and $\mathcal{B}'_{\mathtt{g}}$ such that

$$\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) \leq 2 \cdot \mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{B}_{\mathtt{b}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathtt{g}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) + 4 \cdot \mathsf{Adv}^{\mathtt{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k) + 2 \cdot \mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}'_{\mathtt{g}}}(k),$$

which, due to our assumptions on the building blocks and Lemma 2, implies that $\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ is negligible. Recall that the choice of the PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ was arbitrarily, and thus for any PPTA $\mathtt{CCA}$ adversary $\mathcal{A}$ we can show a negligible upperbound for $\mathsf{Adv}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ as above.

In order to finish the proof of Theorem 1, it remains to prove Claim 3.

Proof of Claim 3. Note that the difference between Game 3 and Game 4 is how a query $C = (\mathsf{tag}, c)$ satisfying the conditions of Bad_3 (or Bad_4) is answered, and Game 3 and Game 4 proceed identically unless Bad_3 or Bad_4 occurs in the corresponding games. This means that we have

$$\Big|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]\Big| \leq \Pr[\mathsf{Bad}_3] = \Pr[\mathsf{Bad}_4]. \tag{3}$$

We claim the following:

Subclaim 1. $\Pr[\mathsf{Bad}_4] \leq 2 \cdot \Pr[\mathsf{Bad}^*_4]$.

Proof of Subclaim 1. The argument here is essentially the same as the one used in the proof of Claim 4.13 in [17].

Note that the event Bad_4, $\mathsf{Bad}^{(0)}_4$, $\mathsf{Bad}^{(1)}_4$, and Bad^*_4 are triggered once $\mathcal{A}$ makes a query $C = (\mathsf{tag}, c)$ satisfying the conditions that cause these events. Moreover, by definition, if any of the latter three events occurs, then Bad_4 occurs. Furthermore, the bit γ is information-theoretically hidden from $\mathcal{A}$'s view in Game 4. This means that the probability of Bad^*_4 occurring is identical to the probability of the event (in Game 4) that is triggered when (1) $\mathcal{A}$ first makes a query satisfying the conditions of Bad_4, (2) γ is picked "on-the-fly" at this point, and then (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (\gamma + 1, c_{\mathtt{in}\gamma}))$ holds. The probability of this event occurring is $\Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}_4 \wedge \mathsf{Bad}^{(\gamma)}_4] = \Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}^{(\gamma)}_4]$ (where the probability is also over Game 4 except the choice of γ). This can be further estimated as follows:

$$\begin{aligned}\Pr_{\gamma \leftarrow \{0,1\}}[\mathsf{Bad}^{(\gamma)}_4] &= \frac{1}{2}\left(\Pr[\mathsf{Bad}^{(0)}_4] + \Pr[\mathsf{Bad}^{(1)}_4]\right) \\ &\geq \frac{1}{2}\Pr[\mathsf{Bad}^{(0)}_4 \vee \mathsf{Bad}^{(1)}_4] = \frac{1}{2}\Pr[\mathsf{Bad}_4],\end{aligned}$$

where we used $\Pr[\mathsf{Bad}^{(0)}_4 \vee \mathsf{Bad}^{(1)}_4] = \Pr[\mathsf{Bad}_4]$, which is by definition.

In summary, we have $\Pr[\mathsf{Bad}_4^*] \geq \frac{1}{2}\Pr[\mathsf{Bad}_4]$, as required. □ (Subclaim 1)

Using Subclaim 1, we can further estimate $\Pr[\mathsf{Bad}_4]$ as follows:

$$\begin{aligned}\Pr[\mathsf{Bad}_4] &\leq 2 \cdot \Pr[\mathsf{Bad}_4^*] \\ &\leq 2 \cdot \Big(\Big|\Pr[\mathsf{Bad}_4^*] - \Pr[\mathsf{Bad}_5^*]\Big| + \Pr[\mathsf{Bad}_5^*]\Big) \\ &\leq 2 \cdot \Big(\Big|\Pr[\mathsf{Bad}_4^*] - \Pr[\mathsf{Bad}_5^*]\Big| + \Pr[\mathsf{Bad}_5]\Big) \\ &\leq 2 \cdot \Big(\Big|\Pr[\mathsf{Bad}_4^*] - \Pr[\mathsf{Bad}_5^*]\Big| + \Big|\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]\Big| + \Pr[\mathsf{Bad}_6]\Big),\end{aligned} \tag{4}$$

where we used $\Pr[\mathsf{Bad}_5^*] \leq \Pr[\mathsf{Bad}_5]$ in the third inequality, which is again by definition. It remains to upperbound the right hand side of the above inequality.

Subclaim 2. *There exists a PPTA* $\mathcal{B}_\mathsf{g}$ *such that* $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_\mathsf{in},\mathcal{B}_\mathsf{g}}(k) = |\Pr[\mathsf{Bad}_4^*] - \Pr[\mathsf{Bad}_5^*]|$.

Proof of Subclaim 2. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathtt{CPA}$ adversary $\mathcal{B}_\mathsf{g}$ with the claimed advantage. The description of $\mathcal{B}_\mathsf{g}$ is as follows:

$\mathcal{B}_\mathsf{g}(pk', c'^*, \alpha'^*_\beta)$**:** (where $\beta \in \{0,1\}$ is $\mathcal{B}_\mathsf{g}$'s challenge bit in its $\mathtt{CPA}$ experiment) $\mathcal{B}_\mathsf{g}$ picks $\gamma \in \{0,1\}$ uniformly at random, then sets $pk_{\mathtt{in}(1-\gamma)} \leftarrow pk'$, $c^*_{\mathtt{in}(1-\gamma)} \leftarrow c'^*$, and $\alpha^*_{1-\gamma} \leftarrow \alpha'^*_\beta$. Next, $\mathcal{B}_\mathsf{g}$ generates $(pk_{\mathtt{in}\gamma}, sk_{\mathtt{in}\gamma}) \leftarrow \mathsf{KKG}_\mathtt{in}(1^k)$ and $(c^*_{\mathtt{in}\gamma}, \alpha^*_\gamma) \leftarrow \mathsf{Encap}_\mathtt{in}(pk_{\mathtt{in}\gamma})$, sets $\alpha^* \leftarrow (\alpha^*_0 \oplus \alpha^*_1)$, and parses α^* as $(r^*_c, r^*_t, K^*_1) \in (\{0,1\}^k)^3$. Then, $\mathcal{B}_\mathsf{g}$ prepares $K^*_1, K^*_0 \in \{0,1\}^k$, $b \in \{0,1\}$, $PK = (pk_\mathtt{in0}, pk_\mathtt{in1}, pk, c)$, $C^* = (\mathsf{tag}^*, c^*)$, $\widehat{sk}_{\mathsf{tag}^*}$, and $\mathsf{st}_\mathcal{E} = (pk_\mathtt{in0}, pk_\mathtt{in1}, r_{\mathcal{A}'} = (r_\mathcal{A}, \widehat{r}_c, \widehat{r}_t, K^*_b))$, exactly as $\mathcal{B}'_\mathsf{g}$ in the proof of Claim 4 does. Finally, $\mathcal{B}_\mathsf{g}$ runs $\mathcal{A}(PK, C^*, K^*_b; r_\mathcal{A})$ until it terminates, where $\mathcal{B}_\mathsf{g}$ answers $\mathcal{A}$'s queries in exactly the same way as $\mathcal{B}'_\mathsf{g}$ does.

When $\mathcal{A}$ terminates, $\mathcal{B}_\mathsf{g}$ checks whether $\mathcal{A}$ has submitted a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the conditions of Bad_4^* (i.e. (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = (c_\mathtt{in0} \| c_\mathtt{in1}) \neq \perp$, and (3) $\mathsf{Decap}_\mathtt{in}(sk_{\mathtt{in}\gamma}, c_{\mathtt{in}\gamma}) \neq \mathcal{E}(\mathsf{st}_\mathcal{E}, c_{\mathtt{in}\gamma})$ hold), which can be checked by using $sk_{\mathtt{in}\gamma}$. If such a query is found, the $\mathcal{B}_\mathsf{g}$ sets $\beta' \leftarrow 1$, otherwise sets $\beta' \leftarrow 0$, and terminates with output β'.

The above completes the description of $\mathcal{B}_\mathsf{g}$. Let $\mathsf{Bad}^*_\mathcal{B}$ be the event that $\mathcal{A}$ submits a decapsulation query that satisfies the conditions (1), (2), and (3) of Bad_4^*, in the experiment simulated by $\mathcal{B}_\mathsf{g}$. Note that $\mathcal{B}_\mathsf{g}$ outputs $\beta' = 1$ only when $\mathsf{Bad}^*_\mathcal{B}$ occurs. Therefore, $\mathcal{B}_\mathsf{g}$'s $\mathtt{CPA}$ advantage can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_\mathsf{in},\mathcal{B}_\mathsf{g}}(k) &= 2 \cdot \left|\Pr[\beta' = \beta] - \frac{1}{2}\right| = \Big|\Pr[\beta' = 1 | \beta = 1] - \Pr[\beta' = 1 | \beta = 0]\Big| \\ &= \Big|\Pr[\mathsf{Bad}^*_\mathcal{B} | \beta = 1] - \Pr[\mathsf{Bad}^*_\mathcal{B} | \beta = 0]\Big|.\end{aligned}$$

With essentially the same arguments as in the proof of Claim 4, we can see that $\mathcal{B}_{\mathsf{g}}$ does a perfect simulation of Game 4 for $\mathcal{A}$ if $\beta = 1$, and does a perfect simulation of Game 5 for $\mathcal{A}$ if $\beta = 0$. In particular, the only difference from the proof of Claim 4 is in which of the positions $(pk_{\mathtt{in0}}, c^*_{\mathtt{in0}}, \alpha^*_0)$ or $(pk_{\mathtt{in1}}, c^*_{\mathtt{in1}}, \alpha^*_1)$ $\mathcal{B}_{\mathsf{g}}$ embeds $\mathcal{B}_{\mathsf{g}}$'s instance of the $\mathtt{CPA}$ experiment. In the proof of Claim 4, the reduction algorithm $\mathcal{B}'_{\mathsf{g}}$ embeds its challenge into $(pk_{\mathtt{in0}}, c^*_{\mathtt{in0}}, \alpha^*_0)$, while in the current proof, the reduction algorithm $\mathcal{B}_{\mathsf{g}}$ embeds its challenge into $(pk_{\mathtt{in}(1-\gamma)}, c^*_{\mathtt{in}(1-\gamma)}, \alpha^*_{1-\gamma})$ for a random $\gamma \in \{0,1\}$. It is easy to see that even after this change, if $\beta = 1$, then the view of $\mathcal{A}$ is identical to that in Game 4, and if $\beta = 0$, then the view of $\mathcal{A}$ is identical to that in Game 5.

Under the situation, the probability that $\mathsf{Bad}^*_{\mathcal{B}}$ occurs in the experiment simulated by $\mathcal{B}_{\mathsf{g}}$ in case $\beta = 1$ (resp. $\beta = 0$) is identical to the probability that Bad^*_4 (resp. Bad^*_5) occurs in Game 4 (resp. Game 5), namely, we have $\Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 1] = \Pr[\mathsf{Bad}^*_4]$ and $\Pr[\mathsf{Bad}^*_{\mathcal{B}}|\beta = 0] = \Pr[\mathsf{Bad}^*_5]$.

In summary, we have $\mathsf{Adv}^{\mathtt{CPA}}_{\Gamma_{\mathtt{in}},\mathcal{B}_{\mathsf{g}}}(k) = |\Pr[\mathsf{Bad}^*_4] - \Pr[\mathsf{Bad}^*_5]|$, as required. □ (Subclaim 2)

Subclaim 3. *There exists a PPTA* $\mathcal{B}_{\mathsf{d}}$ *such that* $\mathsf{Adv}^{\mathtt{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathsf{d}}}(k) = |\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]|$.

Proof of Subclaim 3. Using $\mathcal{A}$ and $\mathcal{E}$ as building blocks, we show how to construct a PPTA $\mathcal{B}$ that has the claimed advantage in distinguishing the distributions considered in Lemma 2. The description of $\mathcal{B}_{\mathsf{d}} = (\mathcal{B}_{\mathsf{d1}}, \mathcal{B}_{\mathsf{d2}})$ as follows:

$\mathcal{B}_{\mathsf{d1}}(1^k)$**:** $\mathcal{B}_{\mathsf{d1}}$ runs $(pk_{\mathtt{in0}}, sk_{\mathtt{in0}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(pk_{\mathtt{in1}}, sk_{\mathtt{in1}}) \leftarrow \mathsf{KKG}_{\mathtt{in}}(1^k)$, $(c^*_{\mathtt{in0}}, \alpha^*_0) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in0}})$, $(c^*_{\mathtt{in1}}, \alpha^*_1) \leftarrow \mathsf{Encap}_{\mathtt{in}}(pk_{\mathtt{in1}})$. Then $\mathcal{B}_{\mathsf{d1}}$ sets $M \leftarrow (c^*_{\mathtt{in0}} \| c^*_{\mathtt{in1}})$ and $\mathsf{st}_{\mathcal{B}} \leftarrow$ ($\mathcal{B}_{\mathsf{d1}}$'s entire view), and terminates with output $(M, \mathsf{st}_{\mathcal{B}})$.

$\mathcal{B}_{\mathsf{d2}}(\mathsf{st}_{\mathcal{B}}, ck, \mathsf{tag}^*, pk, c^*, \widehat{sk}_{\mathsf{tag}^*}, \widehat{r}_c, \widehat{r}_t)$**:** $\mathcal{B}_{\mathsf{d2}}$ sets $PK \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, pk, ck)$ and $C^* \leftarrow (\mathsf{tag}^*, c^*)$, picks $K^* \in \{0,1\}^*$ and $r_{\mathcal{A}} \in \{0,1\}^*$ uniformly at random, and then sets $r_{\mathcal{A}'} \leftarrow (r_{\mathcal{A}}, \widehat{r}_c, \widehat{r}_t, K^*)$ and $\mathsf{st}_{\mathcal{E}} \leftarrow (pk_{\mathtt{in0}}, pk_{\mathtt{in1}}, r_{\mathcal{A}'})$. (Recall that K^*_0 and K^*_1 in Games 5 and 6 are distributed identically, and thus it is sufficient to choose just a single value K^* and pretend as if K^* is K^*_b). Then $\mathcal{B}_{\mathsf{d2}}$ runs $\mathcal{A}(PK, C^*, K^*; r_{\mathcal{A}})$.

$\mathcal{B}_{\mathsf{d2}}$ answers $\mathcal{A}$'s queries as Game 5 does, which is possible because $\mathcal{B}_{\mathsf{d2}}$ possesses $\widehat{sk}_{\mathsf{tag}^*}$ and $\mathsf{st}_{\mathcal{E}}$, and thus $\mathcal{B}_{\mathsf{d2}}$ can run $\mathsf{AltDecap}'_{\mathcal{E}}$ (which internally runs the extractor $\mathcal{E}(\mathsf{st}_{\mathcal{E}}, \cdot)$).

When $\mathcal{A}$ terminates, $\mathcal{B}_{\mathsf{d2}}$ checks whether $\mathcal{A}$ has submitted a query that satisfies the conditions of Bad_5, which can be checked by using $sk_{\mathtt{in0}}$ and $sk_{\mathtt{in1}}$ that $\mathcal{B}_{\mathsf{d2}}$ possesses. If such a query is found, then $\mathcal{B}_{\mathsf{d2}}$ outputs 1, otherwise outputs 0, and terminates.

The above completes the description of $\mathcal{B}_{\mathsf{d}}$. Let $\mathsf{Bad}_{\mathcal{B}}$ be the event that $\mathcal{A}$ submits a decapsulation query $C = (\mathsf{tag}, c)$ that satisfies the conditions of Bad_5 in the experiment simulated by $\mathcal{B}_{\mathsf{d}}$ (i.e. the query satisfying (1) $\mathsf{tag} \neq \mathsf{tag}^*$, (2) $\widehat{\mathsf{TDec}}(\widehat{sk}_{\mathsf{tag}^*}, \mathsf{tag}, c) = (c_{\mathtt{in0}} \| c_{\mathtt{in1}}) \neq \perp$, and (3) $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in0}}, c_{\mathtt{in0}}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in0}})$

or $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}1}, c_{\mathtt{in}1}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, c_{\mathtt{in}1})$). Note that $\mathcal{B}_{\mathtt{d}}$ submits 1 only when $\mathsf{Bad}_{\mathcal{B}}$ occurs. Therefore, $\mathcal{B}_{\mathtt{d}}$'s advantage $\mathsf{Adv}^{\mathsf{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k)$ can be calculated as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) &= \left|\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) = 1] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) = 1]\right| \\ &= \left|\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}} : \mathsf{Bad}_{\mathcal{B}}] - \Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}]\right|.\end{aligned}$$

Consider the case when $\mathcal{B}_{\mathtt{d}}$ is run in the "real" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k)$. It is easy to see that in this case, $\mathcal{B}_{\mathtt{d}}$ simulates Game 5 perfectly for $\mathcal{A}$. Specifically, ck, pk, tag^*, c^*, and $\widehat{sk}_{\mathsf{tag}^*}$ are generated from CKG, TKG, Com, TEnc, and Punc, respectively, in such a way that tag^* is a commitment of $(c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1})$ and c^* is an encryption of $(c^*_{\mathtt{in}0} \| c^*_{\mathtt{in}1})$ under the tag tag^*. Furthermore, $\widehat{r}_c$ and $\widehat{r}_t$ are generated from $\mathsf{rSamp}_{\mathcal{C}}$ and $\mathsf{rSamp}_{\mathcal{T}}$, respectively, which is how they are generated in Game 5. Under the situation, the probability that $\mathcal{A}$ submits a decapsulation query that causes the event $\mathsf{Bad}_{\mathcal{B}}$ is exactly the same as the probability that $\mathcal{A}$ does so in Game 5. That is, we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Real}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}] = \Pr[\mathsf{Bad}_5]$.

On the other hand, consider the case when $\mathcal{B}_{\mathtt{d}}$ is run in the "simulated" experiment $\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k)$. In this case, $\mathcal{B}_{\mathtt{d}}$ simulates Game 6 perfectly for $\mathcal{A}$. Specifically, (ck, tag^*) and $(pk, c^*, \widehat{sk}_{\mathsf{tag}^*})$ are generated by $\mathsf{oSamp}_{\mathcal{C}}(1^k; \widehat{r}_c)$ and $\mathsf{oSamp}_{\mathcal{T}}(\mathsf{tag}^*; \widehat{r}_t)$ with uniformly chosen randomness $\widehat{r}_c$ and $\widehat{r}_t$, respectively, and this is exactly how these values are generated in Game 6. Since this is the only change from the above case, with a similar argument we have $\Pr[\mathsf{Expt}^{\mathtt{TS\text{-}Sim}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) : \mathsf{Bad}_{\mathcal{B}}] = \Pr[\mathsf{Bad}_6]$.

In summary, we have $\mathsf{Adv}^{\mathsf{TS}}_{[\mathcal{C},\mathcal{T}],\mathcal{B}_{\mathtt{d}}}(k) = |\Pr[\mathsf{Bad}_5] - \Pr[\mathsf{Bad}_6]|$, as required. □ (Subclaim 3)

Subclaim 4. $\mathsf{Adv}^{\mathsf{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k) = \Pr[\mathsf{Bad}_6]$.

Proof of Subclaim 4. Note that the view of $\mathcal{A}$ in Game 6 is exactly the same as the view of $\mathcal{A}$ when it is internally run by $\mathcal{A}'$ in the situation where $\mathcal{A}'$ is run in the experiment $\mathsf{Expt}^{\mathsf{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ with the extractor $\mathcal{E}$. Therefore, the probability that $\mathcal{A}$ submits a query that causes the event Bad_6 in Game 6, is exactly the same as the probability that $\mathcal{A}'$ submits a query to $\mathcal{E}$ that makes the experiment $\mathsf{Expt}^{\mathsf{sPA1}}_{\Gamma_{\mathtt{in}},\mathcal{A}',\mathcal{E},2}(k)$ outputs 1 (i.e. $\mathcal{A}'$ submits a query of the form $(j+1, c_{\mathtt{in}j})$ such that $\mathsf{Decap}_{\mathtt{in}}(sk_{\mathtt{in}j}, c_{\mathtt{in}j}) \neq \mathcal{E}(\mathsf{st}_{\mathcal{E}}, (j+1, c_{\mathtt{in}j}))$ for some $j \in \{0,1\}$). □ (Subclaim 4)

Equations (3) and (4), and Subclaims 2, 3 and 4 imply Claim 3. □ (Claim 3)

This concludes the proof of Theorem 1. □ (Theorem 1)

4.2 Second Construction

Let $\Gamma_{\mathtt{in}} = (\mathsf{KKG}_{\mathtt{in}}, \mathsf{Encap}_{\mathtt{in}}, \mathsf{Decap}_{\mathtt{in}})$ be a KEM whose ciphertext length is $n = n(k)$ and whose session-key space is $\{0,1\}^{3k}$ for k-bit security. Let $\mathcal{T} = (\mathsf{TKG}, \mathsf{TEnc}, \mathsf{TDec}, \mathsf{Punc}, \widehat{\mathsf{TDec}})$ be a PTBE scheme and $\mathcal{C} = (\mathsf{CKG}, \mathsf{Com})$

be a commitment scheme. We require the plaintext space of TEnc and the message space of Com to be $\{0,1\}^n$, and the randomness space of TEnc and that of Com to be $\{0,1\}^k$ for k-bit security. Then, our second proposed KEM $\overline{\Gamma} = (\overline{\mathsf{KKG}}, \overline{\mathsf{Encap}}, \overline{\mathsf{Decap}})$ is constructed as in Fig. 4.

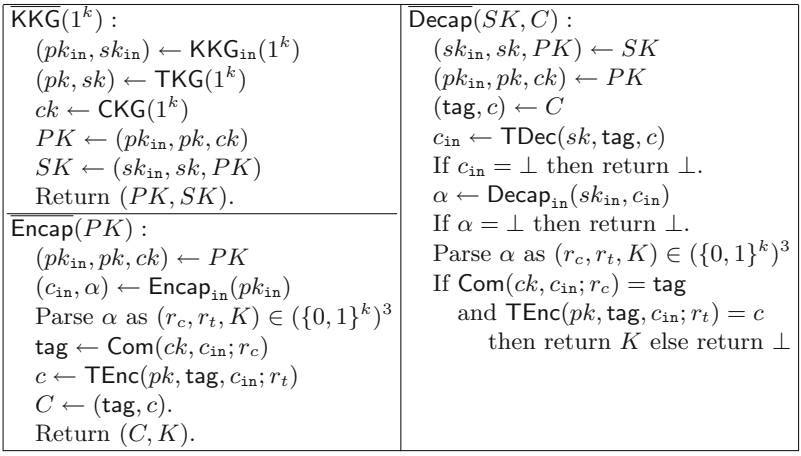


Fig. 4. The second proposed construction: the KEM $\overline{\Gamma}$ based on a KEM $\Gamma_{\mathtt{in}}$, a commitment scheme $\mathcal{C}$, and a PTBE scheme $\mathcal{T}$.

The security of $\overline{\Gamma}$ is guaranteed by the following theorem.

Theorem 2. *Assume that the KEM $\Gamma_{\mathtt{in}}$ is* 1-CCA *secure and* $\mathtt{sPA1}_1$ *secure, the commitment scheme $\mathcal{C}$ is target-binding and trapdoor simulatable, and the PTBE scheme $\mathcal{T}$ is trapdoor simulatable. Then, the KEM $\overline{\Gamma}$ constructed as in Fig. 4 is* CCA *secure.*

The proof of this theorem proceeds very similarly to the proof of Theorem 1, and thus we only explain the difference here, and will give the formal proof in the full version.

Recall that in the proof of Theorem 1, the "bad" queries (for which the extractor fails to extract correct decapsulation results) are dealt with due to the property of "multiple encryption" of two instances of the KEM $\Gamma_{\mathtt{in}}$ with public keys $(pk_{\mathtt{in0}}, pk_{\mathtt{in1}})$. In particular, the reduction algorithm in the proof of Subclaim 2 that attacks the CPA security of the underlying KEM $\Gamma_{\mathtt{in}}$, uses one of secret keys $sk_{\mathtt{in}\gamma}$ (corresponding to $pk_{\mathtt{in}\gamma}$) to detect whether the bad event occurs, while embedding its CPA instance regarding $\Gamma_{\mathtt{in}}$ into the other position, i.e. into $(pk_{\mathtt{in}(1-\gamma)}, c_{\mathtt{in}(1-\gamma)})$. This strategy works thanks to the argument regarding the probabilities given in the proof of Subclaim 1 (which is in turn based on the proof of [17, Claim 4.13]). However, for this argument to work, it seems to us that we inherently have to rely on the $\mathtt{sPA1}_2$ security of $\Gamma_{\mathtt{in}}$, in order for the reduction algorithms (especially, the reduction algorithms attacking the CPA of $\Gamma_{\mathtt{in}}$) to simulate the decapsulation oracle for an adversary $\mathcal{A}$.

The simple idea employed in our second construction is to change the mechanism of detecting the bad queries by relying on the 1-CCA security of $\Gamma_{\mathtt{in}}$, so that a reduction algorithm can check (by its access to the decapsulation oracle) whether $\mathcal{A}$ has submitted a bad decapsulation query. This allows us to use $\Gamma_{\mathtt{in}}$ only in the "single" key setting, leading to only requiring it to be $\mathtt{sPA1}_1$ secure. By employing this idea, a security analysis similar to the recent constructions [31,37,40,44] works, and for the other parts of the security proof (other than the analysis regarding dealing with the bad decapsulation queries) are essentially the same as those in the proof of Theorem 1. For more details, see the full version.

On the Merits of the Second Construction. Since we need to use a KEM which simultaneously satisfies 1-CCA and $\mathtt{sPA1}_1$ security for our second construction, a natural question would be whether we can construct such a scheme. We note that we can achieve such a KEM from a CPA secure PKE (or a KEM) which is also $\mathtt{sPA1}_{2k}$ secure. Specifically, Dodis and Fiore [21, Appendix C] showed how to construct a 1-CCA secure PKE scheme from the combination of a CPA secure PKE scheme and a one-time secure signature scheme (in which $2k$ independently generated public keys are arranged as in the "DDN-lite" construction, but a message is encoded and encrypted in a k-out-of-k fashion, rather than encrypting the same message under k public keys). It is straightforward to see that their construction is $\mathtt{sPA1}_1$ secure if the underlying PKE scheme is $\mathtt{sPA1}_{2k}$ secure. We note that we can slightly optimize their construction by using a CPA secure KEM, instead of a PKE scheme, as a building block. We provide the construction and its security proof in the full version.

However, if we implement a 1-CCA and $\mathtt{sPA1}_1$ secure KEM from a CPA and $\mathtt{sPA1}_{2k}$ secure KEM, there is no merit compared to our first construction (that only requires a CPA and $\mathtt{sPA1}_2$ secure KEM), both in terms of the assumptions and the efficiency. So far, we do not know a better way to construct a 1-CCA and $\mathtt{sPA1}_1$ secure scheme than the approach that relies on [21, Appendix C]. We would like to however emphasize that the point of our second construction is that it may in the future be possible to come up with a direct construction of a KEM (or a PKE scheme) satisfying the requirements for the second construction, from assumptions weaker than those required in our first construction or the combination of our second construction and the Dodis-Fiore construction. We believe that such a possibility of the existence of better constructions can be a raison d'etre of our second construction. In particular, we actually do not need the "full" power of 1-CCA security, but a (seemingly) much weaker security notion such that CPA security holds in the presence of one "plaintext-checking" query [1,47]. More specifically, a plaintext-checking query (for a KEM it could be called a session-key-checking query, but we stick to the terminology in [47]) is a query of the form (c, K), and its reply is the one-bit $(\mathsf{Decap}(sk, c) \stackrel{?}{=} K)$. This could be a hint for the next step.

We would also like to note that even if using the result based on [21], we still achieve the property of "separating" the requirement that a single PKE scheme (or a KEM) needs to satisfy "plaintext awareness" and a "simulatability property" simultaneously in [18]. This is another merit of our second construction.

Acknowledgement. The authors would like to thank the members of the study group "Shin-Akarui-Angou-Benkyou-Kai," and the anonymous reviewers for their helpful comments and suggestions.

A Standard Cryptographic Primitives

Public Key Encryption. A public key encryption (PKE) scheme Π consists of the three PPTAs $(\mathsf{PKG}, \mathsf{Enc}, \mathsf{Dec})$ with the following interface:

$$\underline{\textbf{Key Generation:}} \quad (pk, sk) \leftarrow \mathsf{PKG}(1^k) \qquad \underline{\textbf{Encryption:}} \quad c \leftarrow \mathsf{Enc}(pk, m) \qquad \underline{\textbf{Decryption:}} \quad m \text{ (or } \perp) \leftarrow \mathsf{Dec}(sk, c)$$

where Dec is a deterministic algorithm, (pk, sk) is a public/secret key pair, and c is a ciphertext of a plaintext m under pk. We say that a PKE scheme satisfies *correctness* if for all $k \in \mathbb{N}$, all keys (pk, sk) output from $\mathsf{PKG}(1^k)$, and all plaintexts m, it holds that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.

Since we do not directly use the ordinary security notions for PKE in this paper, we do not introduce them. In Sect. 2.2, we review the (simplified version of) trapdoor simulatability property [14] of a PKE scheme.

Key Encapsulation Mechanism. A key encapsulation mechanism (KEM) Γ consists of the three PPTAs $(\mathsf{KKG}, \mathsf{Encap}, \mathsf{Decap})$ with the following interface:

$$\underline{\textbf{Key Generation:}} \quad (pk, sk) \leftarrow \mathsf{KKG}(1^k) \qquad \underline{\textbf{Encapsulation:}} \quad (c, K) \leftarrow \mathsf{Encap}(pk) \qquad \underline{\textbf{Decapsulation:}} \quad K \text{ (or } \perp) \leftarrow \mathsf{Decap}(sk, c)$$

where Decap is a deterministic algorithm, (pk, sk) is a public/secret key pair that defines a session-key space $\mathcal{K}$, and c is a ciphertext of a session-key $K \in \mathcal{K}$ under pk. We say that a KEM satisfies *correctness* if for all $k \in \mathbb{N}$, all keys (pk, sk) output from $\mathsf{KKG}(1^k)$ and all ciphertext/session-key pairs (c, K) output from $\mathsf{Encap}(pk)$, it holds that $\mathsf{Decap}(sk, c) = K$.

Let $\mathtt{ATK} \in \{\mathtt{CPA}, \mathtt{1\text{-}CCA}, \mathtt{CCA}\}$. We say that a KEM Γ is $\mathtt{ATK}$ secure if for all PPTAs $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathtt{ATK}}_{\Gamma,\mathcal{A}}(k) := 2 \cdot |\Pr[\mathsf{Expt}^{\mathtt{ATK}}_{\Gamma,\mathcal{A}}(k) = 1] - 1/2|$ is negligible, where the $\mathtt{CCA}$ experiment $\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k)$ is defined as follows:

$$\mathsf{Expt}^{\mathtt{CCA}}_{\Gamma,\mathcal{A}}(k) : [\ (pk, sk) \leftarrow \mathsf{KKG}(1^k);\ (c^*, K_1^*) \leftarrow \mathsf{Encap}(pk);\ K_0^* \leftarrow \{0,1\}^k;$$
$$b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}^{\mathsf{Decap}(sk,\cdot)}(pk, c^*, K_b^*);\ \text{Return } (b' \stackrel{?}{=} b)\ ],$$

where in the experiment, $\mathcal{A}$ is not allowed to submit c^* to the oracle. The $\mathtt{1\text{-}CCA}$ (1-bounded $\mathtt{CCA}$) experiment $\mathsf{Expt}^{\mathtt{1\text{-}CCA}}_{\Gamma,\mathcal{A}}(k)$ is defined in the same way as the $\mathtt{CCA}$ experiment, except that $\mathcal{A}$ is allowed to submit a decapsulation query only once. Furthermore, the $\mathtt{CPA}$ experiment $\mathsf{Expt}^{\mathtt{CPA}}_{\Gamma,\mathcal{A}}(k)$ is also defined similarly to the $\mathtt{CCA}$ experiment, except that $\mathcal{A}$ is not allowed to submit any query.

Commitment. A commitment scheme $\mathcal{C}$ consists of the two PPTAs $(\mathsf{CKG}, \mathsf{Com})$ with the following interface:

$$\underline{\textbf{Key Generation:}} \quad ck \leftarrow \mathsf{CKG}(1^k) \qquad \underline{\textbf{Commitment Generation:}} \quad c \leftarrow \mathsf{Com}(ck, m)$$

where ck is a commitment key, and c is a commitment of the message m under ck.

As a (non-standard) requirement, we require the size of a commitment to be k-bit for k-bit security, no matter how long a committed message is.[7]

We say that a commitment scheme $\mathcal{C}$ is *target-binding*[8] if for all PPTAs $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage function $\mathsf{Adv}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) := \Pr[\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) = 1]$ is negligible, where the experiment $\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k)$ is defined as follows:

$$\mathsf{Expt}^{\mathtt{TBind}}_{\mathcal{C},\mathcal{A}}(k) : [\ (m, r, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k);\ ck \leftarrow \mathsf{CKG}(1^k);\ (m', r') \leftarrow \mathcal{A}_2(\mathsf{st}, ck);$$
$$\text{Return 1 iff } \mathsf{Com}(ck, m'; r') = \mathsf{Com}(ck, m; r) \wedge m' \neq m.\ ].$$

Since we do not directly use the hiding property, we do not introduce its formal definition. In Section 2.3, we define the trapdoor simulatability property for a commitment scheme, which is defined in essentially the same way as that for a TSPKE scheme.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 332–352. Springer, Heidelberg (2015)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
3. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013)
4. 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)
5. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)

[7] This requirement (together with the following binding property and the "trapdoor simulatability property") can be easily realized if we are given a TSPKE scheme and a UOWHF. We give the construction in the full version.

[8] Note that the target-binding property is slightly weaker than the ordinary binding notion in the sense that an adversary has to choose its first message before seeing a key ck. The relation between the ordinary binding and target-binding is similar to the relation between collision resistance and target collision resistance of a hash function family. The target-binding was also used in [39].

6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993, pp. 62–73 (1993)
7. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
8. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
9. Canetti, R.: Towards realizing random oracles: hash functions that hide all partial information. In: Kaliski Jr, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
10. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
11. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC 1996, pp. 639–648 (1996)
12. 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)
13. Chen, Y., Zhang, Z.: Publicly evaluable pseudorandom functions and their applications. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 115–134. Springer, Heidelberg (2014)
14. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
15. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)
16. 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)
17. Dachman-Soled, D.: A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme (2013). Full version of [18]. http://eprint.iacr.org/2013/680
18. Dachman-Soled, D.: A black-box construction of a CCA2 encryption scheme from a plaintext aware (sPA1) encryption scheme. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 37–55. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_3
19. Damgård, I.B., 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)
20. Dent, A.W.: The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 289–307. Springer, Heidelberg (2006)
21. Dodis, Y., Fiore, D.: Interactive Encryption and Message Authentication (2013). Full version in [22]. http://eprint.iacr.org/2013/817
22. Dodis, Y., Fiore, D.: Interactive Encryption and Message Authentication. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 494–513. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10879-7_28
23. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: STOC 1991, pp. 542–552 (1991)

24. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
25. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all curcuits. In: FOCS 2013, pp. 40–49 (2013)
27. Goldreich, O., Rothblum, R.D.: Enhancements of trapdoor permutations. J. Crypt. **26**(3), 484–512 (2013)
28. Hajiabadi, M., Kapron, B.M.: Reproducible circularly-secure bit encryption: applications and realizations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 224–243. Springer, Heidelberg (2015)
29. Hemenway, B., Ostrovsky, R.: On homomorphic encryption and chosen-ciphertext security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 52–65. Springer, Heidelberg (2012)
30. Hemenway, B., Ostrovsky, R.: Building lossy trapdoor functions from lossy encryption. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 241–260. Springer, Heidelberg (2013)
31. 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)
32. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
33. Kiltz, E., Mohassel, P., O'Neill, A.: Adaptive trapdoor functions and chosen-ciphertext security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 673–692. Springer, Heidelberg (2010)
34. Lin, H., Tessaro, S.: Amplification of chosen-ciphertext security. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 503–519. Springer, Heidelberg (2013)
35. Lindell, Y.: A simpler construction of CCA2-secure public-key encryption under general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 241–254. Springer, Heidelberg (2003)
36. Lynn, B.Y.S., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004)
37. Matsuda, T., Hanaoka, G.: Achieving chosen ciphertext security from detectable public key encryption efficiently via hybrid encryption. In: Sakiyama, K., Terada, M. (eds.) IWSEC 2013. LNCS, vol. 8231, pp. 226–243. Springer, Heidelberg (2013)
38. Matsuda, T., Hanaoka, G.: Chosen ciphertext security via point obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 95–120. Springer, Heidelberg (2014)
39. Matsuda, T., Hanaoka, G.: Chosen ciphertext security via UCE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 56–76. Springer, Heidelberg (2014)
40. Matsuda, T., Hanaoka, G.: An asymptotically optimal method for converting bit encryption to multi-bit encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 415–442. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_18

41. Matsuda, T., Hanaoka, G.: Constructing and understanding chosen ciphertext security via puncturable key encapsulation mechanisms. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 561–590. Springer, Heidelberg (2015)
42. Mol, P., Yilek, S.: Chosen-ciphertext security from slightly lossy trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 296–311. Springer, Heidelberg (2010)
43. Myers, S., Sergi, M., Shelat, A.: Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 149–165. Springer, Heidelberg (2012)
44. Myers, S., Shelat, A.: Bit encryption is complete. In: FOCS 2009, pp. 607–616 (2009)
45. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC 1989, pp. 33–43 (1989)
46. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990, pp. 427–437 (1990)
47. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–174. Springer, Heidelberg (2001)
48. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC 2008, pp. 187–196 (2008)
49. 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)
50. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)
51. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS 1999, pp. 543–553 (1999)
52. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC 2014, pp. 475–484 (2014)
53. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
54. Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)

Chosen-Ciphertext Security from Subset Sum

Sebastian Faust[1(✉)], Daniel Masny[1], and Daniele Venturi[2]

[1] Faculty of Mathematics, Horst-Görtz Institute for IT Security, Ruhr-Universität Bochum, Bochum, Germany
Sebastian.Faust@ruhr-uni-bochum.de
[2] Department of Computer Science, Sapienza University of Rome, Rome, Italy

Abstract. We construct a public-key encryption (PKE) scheme whose security is polynomial-time equivalent to the hardness of the Subset Sum problem. Our scheme achieves the standard notion of indistinguishability against chosen-ciphertext attacks (IND-CCA) and can be used to encrypt messages of arbitrary polynomial length, improving upon a previous construction by Lyubashevsky, Palacio, and Segev (TCC 2010) which achieved only the weaker notion of semantic security (IND-CPA) and whose concrete security decreases with the length of the message being encrypted.

At the core of our construction is a trapdoor technique which originates in the work of Micciancio and Peikert (Eurocrypt 2012).

Keywords: Public-key cryptography · Chosen-ciphertext security · Subset Sum problem

1 Introduction

Public-Key Encryption (PKE) is perhaps the most basic application of public-key cryptography [10]. Intuitively a PKE scheme allows Alice to encrypt a message M for Bob, given just Bob's public key pk; the received ciphertext C can be decrypted by Bob using the secret key sk corresponding to pk.

Security of a PKE scheme can be formulated in different ways, depending on the assumed adversarial capabilities. The most basic and natural notion is that of indistinguishability against chosen-plaintext attacks (IND-CPA, a.k.a. semantic security) [14]; here we demand that a passive (computationally bounded) adversary only given pk should not be able to distinguish the encryption of two (adversarially chosen) messages M_0, M_1.

Whilst already sufficient for some applications, IND-CPA security is not enough to deal with active adversaries. Hence, researchers have put forward

D. Masny—Supported by DFG Research Training Group GRK 1817/1.

D. Venturi—Supported by the European Commission (Directorate General Home Affairs) under the GAINS project HOME/2013/CIPS/AG/4000005057, and by the European Unions Horizon 2020 research and innovation programme under grant agreement No 644666.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 35–46, 2016.
DOI: 10.1007/978-3-662-49384-7_2

stronger security notions. The de-facto standard notion of security for PKE is that of indistinguishability against chosen-ciphertext attacks [29] (IND-CCA), where we now demand that an active (computationally bounded) adversary given pk should not be able to distinguish the encryption of two (adversarially chosen) messages M_0, M_1 even given access to an oracle decrypting arbitrarily chosen ciphertexts.[1]

By now we dispose of many PKE schemes satisfying IND-CCA security under a variety of assumptions, including factoring [15], decisional and computational Diffie-Hellman [6,8], and learning parity with noise [18].

The Subset Sum assumption. Since its introduction, the Subset Sum problem has been considered a valid alternative to number-theoretic assumptions. In its basic computational version, the Subset Sum problem $\mathrm{SS}(n, \mu)$ (parametrized by integers μ and n) asks to find a secret vector $\mathbf{s} \in \{0,1\}^n$ given a vector $\mathbf{a} \in \mathbb{Z}_\mu^n$ together with the target value $T := \langle \mathbf{a} \cdot \mathbf{s} \rangle \bmod \mu$, where both $\mathbf{a}$ and $\mathbf{s}$ are chosen uniformly at random, and $\langle \cdot, \cdot \rangle$ denotes the inner product. The hardness of $\mathrm{SS}(n, \mu)$ depends on the so-called *density*, which is defined by the ratio $\delta := n/\log \mu$. In case $\delta < 1/n$ or $\delta > n/\log^2 n$, the problem can be solved in polynomial time [12,13,20,21,32]. In case δ is $o(1)$ or even as small as $O(1/\log n)$, the problem is considered to be hard. The best classical algorithm for solving Subset Sum is due to [19], and takes sub-exponential time for solving instances with $\delta = o(1)$ and time $2^{(\ln 2/2 + o(1))n/\log\log n}$ for instances with $\delta = O(1/\log n)$.

One nice feature of the Subset Sum problem is its believed hardness against quantum attacks. At the time of writing, the best quantum attack—due to Bernstein et al. [3]—on Subset Sum requires complexity $2^{(0.241+o(1))n}$ to solve a random instance of the problem.

PKE from Subset Sum. The first PKE scheme based on the hardness of Subset Sum was constructed in the seminal work of Ajtai and Dwork [2], who presented a scheme whose semantic security is as hard to break as solving worst-case instances of a lattice problem called "the unique shortest vector problem" (uSVP). It is well known that Subset Sum can be reduced to uSVP [13,20].

A disadvantage of the scheme in [2] (and its extensions [27,30,31]) is that they are based on Subset Sum only in an indirect way (i.e., via a non-tight reduction to uSVP). This limitation was overcome by the work of Lyubashevsky, Palacio, and Segev [22] that proposed a new PKE scheme achieving IND-CPA security with a simple and direct reduction to solving random instances of the Subset Sum problem.

More precisely, the security of the scheme in [22] is based on the assumption that a random instance $(\mathbf{a}, T)$ of the Subset Sum problem is indistinguishable from uniform. Such a decisional variant of the problem was shown to be equivalent to the above introduced computational version (i.e., to the task of recovering $\mathbf{s}$) by Impagliazzo and Naor [16].

[1] Clearly, the decryption oracle cannot be queried on the challenge ciphertext.

1.1 Our Contributions and Techniques

The work of [22] left as an explicit open problem to construct a PKE scheme achieving IND-CCA security with a direct reduction to the hardness of Subset Sum.

Contributions. In this paper we present a new PKE scheme resolving the above open problem. Previous to our work, the only known PKE schemes with IND-CCA security from Subset Sum were the ones based on uSVP [27,28] (which are not directly based on the hardness of Subset Sum). An additional advantage of our scheme is that it can be used to encrypt an arbitrary polynomial number of bits; this stands in sharp contrast with the scheme of [22], whose concrete security starts to decrease when encrypting messages of length longer than $n \log n$ (where n is, as usual, the Subset Sum dimension).[2] The theorem below summarizes our main result.

Theorem 1 (Main result, informal). *For $q = \Theta(n^2 \log^6 n)$ there exists a PKE scheme with IND-CCA security based on the hardness of* $\mathrm{SS}(n, 2^{n \log n})$.

Techniques. Our scheme (as the one of [22]) is based on the decisional variant of $\mathrm{SS}(n, q^m)$, where q is a small integer and m is an integer. The main observation (also made in [22]) is that, in case $\mu = q^m$, the target value $T := \langle \mathbf{a} \cdot \mathbf{s} \rangle \bmod q^m$ written in base q is equal to $\mathbf{A}\mathbf{s} + e(\mathbf{A}, \mathbf{s})$ where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is a matrix whose i-th column corresponds to the i-th element of vector $\mathbf{a}$ written in base q, and $e(\mathbf{A}, \mathbf{s})$ is a vector in $\mathbb{Z}_q^m$ (function of $\mathbf{A}$ and $\mathbf{s}$) which corresponds to the carries when performing "grade-school" addition. This particular structure resembles the structure of an instance of the learning with errors (LWE) problem [31], with the important difference that the noise term is "deterministic" and, in fact, completely determined by the matrix $\mathbf{A}$ and the vector $\mathbf{s}$.

We use the above similarity between LWE and Subset Sum to construct our new PKE scheme, using a trapdoor technique due to Micciancio and Peikert [24]. Essentially our scheme relies on a tag-based trapdoor function, where the trapdoor is associated with a hidden tag. Whenever the function is evaluated w.r.t. the hidden tag, the trapdoor disappears and the function is hard to invert; for all other tags the function can be inverted efficiently given the trapdoor. Using the leftover hash lemma, one can switch the hidden tag without the adversary noticing.

The above technique allows us to prove that our PKE scheme achieves a weaker (tag-based) CCA notion. This means that each ciphertext is associated with a tag τ, and in the security game the adversary has to commit in advance to the tag τ^* which will be associated with the challenge ciphertext.[3] In the security proof we first switch the tag associated with the hidden trapdoor with

[2] In particular, for message length n^2 the scheme of [22] can be broken in polynomial time.

[3] Decryption queries for the challenge tag τ^* are disallowed.

the challenge tag (using the trapdoor technique outlined above). Now, the simulator is not able to decrypt a message related to the challenge tag which allows us to argue about indistinguishability of the PKE scheme.

It is well known that the above weak tag-based CCA notion can be generically enhanced to full-fledged IND-CCA security using a one-time signature scheme [17]. This allows us to conclude Theorem 1.

Efficiency. Let ℓ be the length of the messages to be encrypted, and denote by n, q and m the parameters of the Subset Sum problem. The secret key of our PKE scheme consists of a binary matrix of dimension $n \times m$; the public key consists of 3 matrices of elements in $\mathbb{Z}_q$, with dimensions (respectively) $m \times n$, $n \times n$, and $\ell \times n$. A ciphertext consists of 3 vectors of elements in $\mathbb{Z}_q$, with dimensions (respectively) m, n, and ℓ.

1.2 Related Work

Pioneered by Merkle and Hellman [23], the first construction of PKE schemes based on Subset Sum were based on instances of the problem with special structure. All these constructions have been subsequently broken. (See [26] for a survey.)

In a seminal paper, Impagliazzo and Naor [16] presented constructions of universal one-way hash functions, pseudorandom generators and bit commitment schemes based on the hardness of random instances of Subset Sum.

Besides constructing PKE schemes, [22] additionally presents an oblivious transfer protocol with security against malicious senders and semi-honest receivers. The Subset Sum problem has also recently been used to solve the problem of outsourced pattern matching [11] in the cloud setting.

2 Preliminaries

For two distributions $\mathcal{D}$ and $\mathcal{D}'$ over Ω, $\mathcal{D}(x)$ is the probability assigned to $x \in \Omega$ and $\Delta[\mathcal{D}, \mathcal{D}'] := \frac{1}{2} \sum_{x \in \Omega} |\mathcal{D}(x) - \mathcal{D}'(x)|$ is the statistical distance between $\mathcal{D}$ and $\mathcal{D}'$. We denote with $x \leftarrow X$ that x is sampled according to the distribution X. If X is a set, then this denotes that x is sampled uniformly at random from X. $\lfloor \cdot \rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ is the rounding function defined by $\lfloor x \rceil_2 := \lfloor x \cdot \frac{2}{q} \rceil$.

Vectors and matrices are denoted in boldface. For two vectors $\mathbf{u}$,$\mathbf{v}$, with $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, the inner product between $\mathbf{u}$ and $\mathbf{v}$ is defined as $\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{i=1}^{n} u_i \cdot v_i$. We represent elements in $\mathbb{Z}_q$ by integers in the range $[-(q-1)/2; (q-1)/2]$. For an element $v \in \mathbb{Z}_q$, its length, denoted by $|v|$ is the absolute value of its representative in the range $[-(q-1)/2; (q-1)/2]$. For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, we define $\|\mathbf{v}\|_\infty := \max_{1 \le i \le n} |v_i|$.

We say that a function ν is negligible in the security parameter n, if it is asymptotically smaller than the inverse of any polynomial in n, i.e. $\nu(n) = n^{-\omega(1)}$. An algorithm A is probabilistic polynomial-time (PPT) if A is randomized, and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x; r)$ (i.e., A with input x and random coins r) terminates in at most poly($|x|$) steps.

2.1 Subset Sum

Traditionally a Subset Sum $\mathrm{SS}(n, \mu)$ instance is defined as $\mathbf{a} := (a_1, \ldots, a_n)$ and a target $T := \langle \mathbf{a}, \mathbf{s} \rangle \bmod \mu$, where the goal is to recover $\mathbf{s} \in \{0,1\}^n$. For a modulus $\mu = q^m$, Lyubashevsky, Palacio, and Segev [22] gave an alternative description which shows its similarities with the LWE problem more clearly. First they define matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, where $a_{j,i} := \left\lfloor \frac{a_i}{q^{j-1}} \right\rfloor \bmod q$. Thus,

$$\mathbf{A} \odot \mathbf{s} := \sum_{i=1}^{n} s_i \cdot \left(\sum_{j=1}^{m} a_{j,i} q^{j-1} \right) \bmod \mathrm{q^m} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{s_i} \cdot \mathrm{a_i} \bmod \mathrm{q^m} = \langle \mathbf{a}, \mathbf{s} \rangle \bmod \mathrm{q^m}$$

where $\mathbf{s} \in \{0,1\}^n$. Notice that when $\mathbf{As}$ is the matrix vector multiplication $mod q$, then $\mathbf{A} \odot \mathbf{s} = \mathbf{As} + e(\mathbf{A}, \mathbf{s}) \bmod q \in \mathbb{Z}_q^m$. Here $e(\mathbf{A}, \mathbf{s})_1 := 0$, and for $1 < j \leq m$ the j-th component of $e(\mathbf{A}, \mathbf{s})$ is given by

$$e(\mathbf{A}, \mathbf{s})_j := \left\lfloor \frac{\sum_{i=1}^{n} s_i a_{j-1,i}}{q} \right\rfloor + c_j \mod q,$$

for carry c_j which is recursively defined by $c_2 := 0$ and

$$c_j := \left\lfloor \frac{(\sum_{i=1}^{n} s_i a_{j-1,i}) \bmod \mathrm{q} + \mathrm{e}(\mathbf{A}, \mathbf{s})_{\mathrm{j-1}}}{q} \right\rfloor \mod q.$$

Since c_j is small, and moreover it is the only part of $e(\mathbf{A}, \mathbf{s})_j$ which depends on $e(\mathbf{A}, \mathbf{s})_{j-1}$, one has that $e(\mathbf{A}, \mathbf{s})_j - c_j$ is bound by the Hoeffding bound. This implies an overall bound on $e(\mathbf{A}, \mathbf{s})_j$:

Lemma 1 (*[22]* Lemma 3.3). *For any $n, m \in \mathbb{N}$ and $\mathbf{s} \in \{0,1\}^n$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that*

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}} [\|e(\mathbf{A}, \mathbf{s})\|_\infty \geq \sqrt{n} \log n] \leq \nu(n).$$

The main difference between Subset Sum and LWE is that error term $e(\mathbf{A}, \mathbf{s})$ is uniquely determined given $\mathbf{A}$ and $\mathbf{s}$ where as in case of LWE, error e is sampled from a discrete Gaussian distribution independent of $\mathbf{A}$, $\mathbf{s}$.

The Subset Sum assumption. A $\mathrm{SS}(n, q^m)$ instance has the following distribution:

$$\mathcal{D}_{\mathrm{SS}(n,q^m)} := \{(\mathbf{A}, \mathbf{A} \odot \mathbf{s}) \mid \mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}, \mathbf{s} \leftarrow \{0,1\}^n\}.$$

The challenge is to distinguish $\mathcal{D}_{\mathrm{SS}(n,q^m)}$ from a uniform $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. The advantage of an algorithm A in breaking the $\mathrm{SS}(n, q^m)$ assumption is

$$\mathbf{Adv}_{\mathrm{SS}(n,q^m)}(\mathsf{A}) = |\Pr[\mathsf{A}(\mathbf{A}, \mathbf{b}) = 1] - \Pr[\mathsf{A}(\mathbf{A}', \mathbf{b}') = 1]|,$$

where $(\mathbf{A}, \mathbf{b}) \leftarrow \mathcal{D}_{\mathrm{SS}(n,q^m)}$ and $(\mathbf{A}', \mathbf{b}') \leftarrow \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. It was shown by Impagliazzo and Naor [16] that this decisional version of Subset Sum is as hard as recovering the hidden vector $\mathbf{s}$.

Re-randomizing Subset Sum. We use a technique introduced by Lyubashevsky [21] allowing to re-randomize a Subset Sum sample. This technique is based on the leftover hash lemma:

Lemma 2 (Leftover hash lemma). *For* $2m \geq n + 1 + \omega(\log n + 1)/\log q$ *and polynomial* ℓ*, there exists a negligible function* $\nu : \mathbb{N} \to [0,1]$ *such that the statistical distance*

$$\Delta\big[(\mathbf{A}, \mathbf{RA}, \mathbf{a}, \mathbf{Ra}), (\mathbf{A}, \mathbf{C}, \mathbf{a}, \mathbf{c})\big] \leq \nu(n),$$

for $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{R} \leftarrow [-\sqrt{q}/2, \sqrt{q}/2]^{\ell \times m}$, $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, $\mathbf{C} \leftarrow \mathbb{Z}_q^{\ell \times n}$, $\mathbf{c} \leftarrow \mathbb{Z}_q^{\ell}$.

A Subset Sum sample $(\mathbf{A}, \mathbf{b}) \leftarrow \mathcal{D}_{\mathrm{SS}(n,q^m)}$ can now be re-randomized to $(\mathbf{RA}, \mathbf{Rb})$ where $\mathbf{RA}$ is statistically close to uniform given $\mathbf{A}, \mathbf{b}$ and $\mathbf{Rb}$. Note that $(\mathbf{RA}, \mathbf{Rb})$ is not $\mathrm{SS}(n, q^m)$-distributed anymore.

Given this re-randomization technique, we are able to construct a tag-based trapdoor function [24] and a PKE scheme whose hardness is independent of the amount of simultaneously encrypted bits. Of major significance is the fact that, after re-randomization, the noise is still bounded:

Lemma 3 (*[22]* Lemma 3.4). *For any* $n, m \in \mathbb{N}$, $\mathbf{s} \in \{0,1\}^n$ *and* $\mathbf{r} \in [-\sqrt{q}/2, \sqrt{q}/2]^m$*, there exists a negligible function* $\nu : \mathbb{N} \to [0,1]$ *such that*

$$\Pr_{\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}}[\mathbf{r} \cdot e(\mathbf{A}, \mathbf{s}) \geq \sqrt{qnm}\log^2 n + \sqrt{q}m] \leq \nu(n).$$

This bound will be crucial to show the correctness of our proposed PKE.

2.2 Tag-Based Encryption

The main motivation behind the concept of tag-based encryption (TBE) comes from the fact that it is possible to transform an identity-based encryption scheme into an IND-CCA secure PKE scheme [4,5]. Kiltz [17] showed that these transformations already work starting from TBE.

A TBE scheme with tag-space $\mathcal{T}$, message-space $\mathcal{M}$, and security parameter n, consists of the following three PPT algorithms $\mathrm{TBE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Gen}(1^n)$: Outputs a secret key sk and a public key pk.
$\mathsf{Enc}(pk, \tau, M)$: Outputs a ciphertext C for $M \in \mathcal{M}$, and tag $\tau \in \mathcal{T}$.
$\mathsf{Dec}(sk, \tau, C)$: Outputs the decrypted message M of ciphertext C with respect to tag $\tau \in \mathcal{T}$, or an invalid symbol $\perp$.

For correctness, we require that for any τ, M and $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$:

$$\mathsf{Dec}(sk, \tau, \mathsf{Enc}(pk, \tau, M)) = M$$

holds with overwhelming probability. As for security, we define the following selective-tag weak CCA game $\mathrm{G}_{\mathrm{TBE}}$ [17]:

1. Adversary A picks a tag $\tau^* \in \mathcal{T}$.
2. Run $(sk, pk) \leftarrow \mathsf{Gen}(1^n)$. Adversary A receives public key pk and gets permanent access to an oracle which outputs $\mathsf{Dec}(sk, \tau, C)$ upon input requests of the form $\mathsf{QueryDec}(C, \tau)$ for all $\tau \neq \tau^*$, and $\perp$ otherwise.
3. A chooses M_0 and M_1 from $\mathcal{M}$ and receives $C \leftarrow \mathsf{Enc}(pk, \tau^*, M_u)$ for $u \leftarrow \{0,1\}$.
4. Finally A outputs u' and $\mathrm{G}_{\mathrm{TBE}}$ outputs 1 iff $u' = u$.

The advantage of an adversary A in game $\mathrm{G}_{\mathrm{TBE}}$ is defined as

$$\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) := \left| \Pr[\mathrm{G}_{\mathrm{TBE}}(\mathsf{A}) = 1] - \frac{1}{2} \right|,$$

and a TBE scheme is called secure against selective-tag weak CCA adversaries, if for all PPT A there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that $\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) \leq \nu(n)$.

Given an exponential tag-space, there is a transformation from a TBE scheme satisfying the above notion to an IND-CCA secure PKE; the transformation requires a one-time signature scheme or a message authentication code plus a commitment [17].

We embed the tags in our proposed TBE using a full-rank differences (FRD) encoding $\mathcal{H}$ [1,7]. This means that $\mathcal{H} : \mathbb{Z}_2^n \to \mathbb{Z}_2^{n \times n}$, $\tau \mapsto \mathbf{H}_\tau$ and $\forall \tau \neq \tau' \in \mathbb{Z}_2^n$ $\mathbf{H}_\tau - \mathbf{H}_{\tau'}$ has full rank.

3 A Subset Sum Based TBE

For security parameter n, let $q = \Theta(n^2 \log^6 n)$, $2 \mid q$, and $m = \Theta(n)$ for appropriate constant factors. The following three algorithms describe our TBE $=$ $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ based on $\mathrm{SS}(n, q^m)$ with tag space $\mathcal{T} := \mathbb{Z}_2^n \setminus \{\mathbf{0}\}$ (where $\mathbf{0}$ is the all-zero vector of length n) and message space $\mathcal{M} := \{0,1\}^\ell$.

$\mathsf{Gen}(1^n)$: Sample $\mathbf{R} \leftarrow \{0,1\}^{n \times m}$ and $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{C} \leftarrow \mathbb{Z}_q^{\ell \times n}$. Define $\mathbf{B} := \mathbf{RA}$. The private and public key are defined as

$$sk := \mathbf{R}, \quad pk := (\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^{\ell \times n}.$$

$\mathsf{Enc}(pk, \tau, M)$: Pick $\mathbf{R}' \leftarrow [-\sqrt{q}/2, \sqrt{q}/2]^{n \times m}$, $\mathbf{R}'' \leftarrow [-\sqrt{q}/2, \sqrt{q}/2]^{\ell \times m}$, $\mathbf{s} \leftarrow \{0,1\}^n$ and define

$$\begin{aligned}
\mathbf{c}_0 &:= \mathbf{As} + e(\mathbf{A}, \mathbf{s}) && \in \mathbb{Z}_q^m \\
\mathbf{c}_1 &:= \left(\mathbf{B} + \frac{q}{2} \cdot \mathbf{H}_\tau\right) \mathbf{s} + \mathbf{R}' \cdot e(\mathbf{A}, \mathbf{s}) && \in \mathbb{Z}_q^n \\
\mathbf{c}_2 &:= \mathbf{Cs} + \mathbf{R}'' \cdot e(\mathbf{A}, \mathbf{s}) + \frac{q}{2} \cdot M && \in \mathbb{Z}_q^\ell
\end{aligned}$$

where $\mathbf{H}_\tau$ is the matrix representation of τ.

$\mathsf{Dec}(sk, \tau, C)$: Compute

$$\hat{\mathbf{s}} := \left\lfloor (\mathbf{R}\ \mathbf{I}) \cdot \begin{pmatrix} -\mathbf{c}_0 \\ \mathbf{c}_1 \end{pmatrix} \right\rceil_2 .$$

and $\mathbf{s} = \mathbf{H}_\tau^{-1}\hat{\mathbf{s}}$. If $\mathbf{c}_0 \neq \mathbf{A} \odot \mathbf{s}$ or $\|\mathbf{c}_1 - \left(\mathbf{B} + \frac{q}{2} \cdot \mathbf{H}_\tau\right)\mathbf{s}\|_\infty \geq \frac{q}{4}$ output $\perp$. Otherwise output message $M = \lfloor \mathbf{c}_2 - \mathbf{C}\mathbf{s} \rceil_2$.

3.1 Correctness

The correctness of the scheme follows basically from the bounds on the noise of re-randomized Subset Sum instances. Given these bounds, the noise will be smaller than $q/4$ such that it will be rounded away by the rounding function $\lfloor \cdot \rceil_2$.

Theorem 2 (Correctness). *Let $q = O(n^2 \log^6 n)$, $2 \mid q$, $m = \Theta(n)$, and $\ell \in O(n^c)$ for some constant c. Then for any $\tau \in \mathbb{Z}_2^n$, $M \in \{0,1\}^\ell$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that*

$$\Pr_{(sk,pk) \leftarrow \mathsf{Gen}(1^n)}[\mathsf{Dec}(sk, \tau, \mathsf{Enc}(pk, \tau, M)) \neq M] \leq \nu(n).$$

Proof. Given a ciphertext $C = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$, in case Dec successfully reconstruct $\mathbf{s}$, the decryption algorithm computes

$$\lfloor \mathbf{c}_2 - \mathbf{C}\mathbf{s} \rceil_2 = \left\lfloor \mathbf{R}'' \cdot e(\mathbf{A}, \mathbf{s}) + \frac{q}{2} \cdot M \right\rceil_2 = \lfloor \mathbf{R}'' \cdot e(\mathbf{A}, \mathbf{s}) \rceil_2 + M.$$

By Lemma 3, $\|\mathbf{R}'' \cdot e(\mathbf{A}, \mathbf{s})\|_\infty \leq \sqrt{qnm} \log^2 n + \sqrt{q}m < q/4$ with overwhelming probability over $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ (for appropriately chosen constants). Hence $\lfloor \mathbf{R}'' \cdot e(\mathbf{A}, \mathbf{s}) \rceil_2 = \mathbf{0}$ and Dec outputs M.

For the same reason $\lfloor (\mathbf{R}' - \mathbf{R}) \cdot e(\mathbf{A}, \mathbf{s}) \rceil_2 = \mathbf{0}$ holds with overwhelming probability over $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ (for appropriately chosen constants). Therefore Dec reconstructs

$$\left\lfloor (\mathbf{R}\ \mathbf{I}) \cdot \begin{pmatrix} -\mathbf{c}_0 \\ \mathbf{c}_1 \end{pmatrix} \right\rceil_2 = \left\lfloor \frac{q}{2} \cdot \mathbf{H}_\tau \mathbf{s} + (\mathbf{R}' - \mathbf{R}) e(\mathbf{A}, \mathbf{s}) \right\rceil_2 = \left\lfloor \frac{q}{2} \cdot \mathbf{H}_\tau \mathbf{s} \right\rceil_2 .$$

The coordinates of $\mathbf{H}_\tau$ are in $\mathbb{Z}_2$, and as $2 \mid q$, we get $\hat{\mathbf{s}} = \left\lfloor \frac{q}{2} \cdot \mathbf{H}_\tau \mathbf{s} \mod q \right\rceil_2 = \mathbf{H}_\tau \mathbf{s} \mod 2$. This results in the correct reconstruction of $\mathbf{s} = \mathbf{H}_\tau^{-1}\hat{\mathbf{s}} = \mathbf{H}_\tau^{-1}\mathbf{H}_\tau \mathbf{s}$.

3.2 Proof of Security

The intuition behind the security proof is that $\mathbf{B} = \mathbf{R}\mathbf{A}$ is statistically indistinguishable from $\mathbf{B}' = \mathbf{R}\mathbf{A} - \frac{q}{2}\mathbf{H}_{\tau^*}$. But when $\mathbf{B}'$ is used as part of the public key, ciphertexts with tag τ^* can not be decrypted using Dec anymore. During the proof, we will show that there are ciphertexts for τ^* which are at least as hard to decrypt as solving $\mathrm{SS}(n, q^m)$. Given any algorithm guessing the message encrypted in such a ciphertext and therefore breaking the security of TBE, there will be also an algorithm solving $\mathrm{SS}(n, q^m)$.

Theorem 3 (CCA Security). *Let $q = \Theta(n^2 \log^6 n)$, $2 \mid q$, and $m = \Theta(n)$ for appropriate constant factors. If the* $\mathrm{SS}(n, q^m)$ *assumption holds (which corresponds to density $\delta \in O(1/\log n)$), then the proposed* TBE *scheme is secure against* selective-tag weak CCA *adversaries. In particular, for every PPT algorithm* A *there exist a PPT algorithm* D *and a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:*

$$\mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) \leq \mathbf{Adv}_{\mathrm{SS}(n,q^m)}(\mathsf{D}) + \nu(n).$$

Proof. We construct an algorithm D which will distinguish $\mathrm{SS}(n, q^m)$ from uniform invoking a successful adversary A in game $\mathrm{G}_{\mathrm{TBA}}$. If D receives a $\mathrm{SS}(n, q^m)$ instance D will simulate game $\mathrm{G}_{\mathrm{TBA}}$ and a successful A will guess b correctly with probability $\frac{1}{2} + \mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) > \frac{1}{2} + \nu(n)$. When D receives a uniform input, D will simulate a game in which the challenge ciphertext is independent of message M_u, and hence independent of u. Therefore guess u' of A will be correct (i.e., $u' = u$) with probability $\frac{1}{2}$.

In the following, we describe algorithm D interacting with A and afterwards we analyse its success probability.

1. D receives a $\mathrm{SS}(n, q^m)$ challenge $(\mathbf{A}, \mathbf{b})$ and invokes A which will send a tag $\tau^* \in \mathcal{T}$.
2. D samples $\mathbf{R}' \leftarrow [-\sqrt{q}/2, \sqrt{q}/2]^{n \times m}$, $\mathbf{R}'' \leftarrow [-\sqrt{q}/2, \sqrt{q}/2]^{\ell \times m}$ and sets $pk = (\mathbf{A}, \mathbf{B} := \mathbf{R}'\mathbf{A} - \frac{q}{2}\mathbf{H}_{\tau^*}, \mathbf{C} := \mathbf{R}''\mathbf{A})$ which is by Lemma 2 statistically close to the output distribution of public keys of Gen. The public key pk is given to A.

   Thus, D uses $\mathbf{R}'$ to respond to $\mathsf{QueryDec}(C, \tau)$ queries as follows: If $\tau = \tau^*$ output $\perp$. Otherwise D uses $\mathsf{Dec}(\mathbf{R}', \tau, C)$ to reconstruct:

$$\hat{\mathbf{s}} := \left\lfloor (\mathbf{R}' \ \mathbf{I}) \cdot \begin{pmatrix} -\mathbf{c}_0 \\ \mathbf{c}_1 \end{pmatrix} \right\rceil_2 .$$

   For a properly distributed C,

$$\begin{aligned} \hat{\mathbf{s}} &= \left\lfloor \left(\tfrac{q}{2} \cdot \mathbf{H}_\tau - \tfrac{q}{2}\mathbf{H}_{\tau^*} \mod q\right) \mathbf{s} \right\rceil_2 \\ &= \left\lfloor \left(\tfrac{q}{2} \cdot (\mathbf{H}_\tau - \mathbf{H}_{\tau^*} \mod 2)\right) \mathbf{s} \right\rceil_2 = (\mathbf{H}_\tau - \mathbf{H}_{\tau^*})\mathbf{s}. \end{aligned}$$

   $\mathbf{s}$ is reconstructed by computing $\mathbf{s} = (\mathbf{H}_\tau - \mathbf{H}_{\tau^*})^{-1}\hat{\mathbf{s}}$. If $\mathbf{c}_0 \neq \mathbf{A} \odot \mathbf{s}$ or $\|\mathbf{c}_1 - \left(\mathbf{B} + \frac{q}{2} \cdot \mathbf{H}_\tau\right)\mathbf{s}\|_\infty \geq \frac{q}{4}$ output $\perp$. This ensures that the output of $\mathsf{QueryDec}(C, \tau)$ is independent of $\mathbf{R}'$ conditioned on $\mathbf{B} = \mathbf{R}'\mathbf{A} - \frac{q}{2}\mathbf{H}_{\tau^*}$ and C is a proper cipertext for randomness $\mathbf{s}$. D follows now the description of $\mathsf{Dec}(\mathbf{R}', \tau, C)$ such that by Theorem 2 for all properly generated C and $\tau \neq \tau^*$ $\mathsf{QueryDec}(C, \tau)$ outputs the correct message M.
3. A sends M_0 and M_1. Now D samples $u \leftarrow \{0,1\}$, sets $C^* := (\mathbf{b}, \mathbf{R}'\mathbf{b}, \mathbf{R}''\mathbf{b} + \frac{q}{2}M_u)$, and sends C^* to A.
4. Finally A outputs u' and D outputs 1 iff $u' = u$.

When $\mathbf{b} = \mathbf{As} + e(\mathbf{A}, \mathbf{s})$, the challenge ciphertext C^* is a proper ciphertext for public key pk and randomness $\mathbf{s}$:

$$\mathbf{c}_0 := \mathbf{b} = \mathbf{As} + e(\mathbf{A}, \mathbf{s})$$
$$\mathbf{c}_1 := \mathbf{R}'\mathbf{b} = \mathbf{R}'\mathbf{As} + \mathbf{R}'e(\mathbf{A}, \mathbf{s}) = \left(\mathbf{B} + \frac{q}{2} \cdot \mathbf{H}_\tau\right)\mathbf{s} + \mathbf{R}'e(\mathbf{A}, \mathbf{s})$$
$$\mathbf{c}_2 := \mathbf{R}''\mathbf{b} + \frac{q}{2}M_u = \mathbf{R}''\mathbf{As} + \mathbf{R}''e(\mathbf{A}, \mathbf{s}) + \frac{q}{2}M_u = \mathbf{Cs} + \mathbf{R}''e(\mathbf{A}, \mathbf{s}) + \frac{q}{2}M_u.$$

Note that, by Lemma 2, there is enough entropy in $\mathbf{R}', \mathbf{R}''$ such that $\mathbf{B}, \mathbf{R}'e(\mathbf{A}, \mathbf{s})$ and $\mathbf{C}, \mathbf{R}''e(\mathbf{A}, \mathbf{s})$ are independent. In this case B outputs 1 with roughly probability $\frac{1}{2} + \mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A})$.

In the other case, i.e. when $\mathbf{A}, \mathbf{b} \leftarrow \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, we know that $\mathbf{c}_2 := \mathbf{R}''\mathbf{b} + \frac{q}{2}M_u$ is uniform and independent of $\mathbf{A}, \mathbf{C}, \mathbf{c}_0$ and $\mathbf{c}_1$ by Lemma 2. Therefore C^* is independent of u and for any output u' of A:

$$\Pr_{u \leftarrow \{0,1\}}[u = u'] = \frac{1}{2}.$$

Summing up, D outputs 1 for a $\mathrm{SS}(n, q^m)$ instance with roughly probability $\frac{1}{2} + \mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A})$, and it outputs 1 otherwise with probability $\frac{1}{2}$. This implies

$$\mathbf{Adv}_{\mathrm{SS}(n,q^m)}(\mathsf{D}) = \mathbf{Adv}_{\mathrm{TBE}}(\mathsf{A}) - \nu(n),$$

for a negligible function ν, concluding the proof.

4 Conclusions and Open Problems

We presented a construction of a new PKE scheme with a simple and direct security proof based on the hardness of random instances of the Subset Sum problem. Our scheme achieves IND-CCA security and its concrete security does not depend on the length of the messages being encrypted. This resolves the main open problems from the previous work by Lyubashevsky, Palacio, and Segev [22].

Similarly to one of the constructions in [22], it is not hard to see that actually our PKE scheme achieves the stronger notion of IND-CCA security against non-adaptive leakage attacks.[4] We leave it as an open problem to construct a PKE scheme with IND-CCA security against fully adaptive leakage attacks. An approach towards answering this question would be to construct a hash proof system [9] based on Subset Sum, as this would directly yield a leakage-resilient IND-CCA secure PKE [25].

It would also be interesting to construct PKE schemes with additional properties (always based on Subset Sum), such as circular security, key-dependent message security, and security against related-key attacks.

[4] Since the latter notion is a very weak form of leakage resilience, we preferred to not work out the details.

References

1. 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)
2. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: ACM STOC, pp. 284–293 (1997)
3. Bernstein, D.J., Jeffery, S., Lange, T., Meurer, A.: Quantum algorithms for the subset-sum problem. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 16–33. Springer, Heidelberg (2013)
4. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
5. 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)
6. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. J. Cryptol. **22**(4), 470–504 (2009)
7. Cramer, R., Damgård, I.: On the amortized complexity of zero-knowledge protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009)
8. 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)
9. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
11. Faust, S., Hazay, C., Venturi, D.: Outsourced pattern matching. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part II. LNCS, vol. 7966, pp. 545–556. Springer, Heidelberg (2013)
12. Flaxman, A.D., Przydatek, B.: Solving medium-density subset sum problems in expected polynomial time. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 305–314. Springer, Heidelberg (2005)
13. Frieze, A.M.: On the Lagarias-Odlyzko algorithm for the subset sum problem. SIAM J. Comput. **15**(2), 536–539 (1986)
14. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
15. Hofheinz, D., Kiltz, E., Shoup, V.: Practical chosen ciphertext secure encryption from factoring. J. Cryptol. **26**(1), 102–118 (2013)
16. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. J. Cryptol. **9**(4), 199–216 (1996)
17. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
18. Kiltz, E., Masny, D., Pietrzak, K.: Simple chosen-ciphertext security from low-noise LPN. In: PKC, pp. 1–18. (2014)
19. Kirchner, P., Fouque, P.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: CRYPTO, pp. 43–62. (2015)

20. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. J. ACM **32**(1), 229–246 (1985)
21. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005)
22. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)
23. Merkle, R.C., Hellman, M.E.: Hiding information and signatures in trapdoor knapsacks. IEEE Trans. Inf. Theor. **24**(5), 525–530 (1978)
24. 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)
25. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM J. Comput. **41**(4), 772–814 (2012)
26. Odlyzko, A.M.: The rise and fall of knapsack cryptosystems. In: Symposia of Applied Mathematics, pp. 75–88. (1990)
27. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: ACM STOC, pp. 333–342. (2009)
28. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. SIAM J. Comput. **40**(6), 1803–1844 (2011)
29. 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)
30. Regev, O.: New lattice based cryptographic constructions. In: ACM STOC, pp. 407–416. (2003)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 1–40 (2009)
32. Shallue, A.: An improved multi-set algorithm for the dense subset sum problem. In: van der Poorten, A.J., Stein, A. (eds.) ANTS-VIII 2008. LNCS, vol. 5011, pp. 416–429. Springer, Heidelberg (2008)

On the Hardness of Proving CCA-Security of Signed ElGamal

David Bernhard[1(✉)], Marc Fischlin[2], and Bogdan Warinschi[1]

[1] University of Bristol, Bristol, UK
bernhard@cs.bris.ac.uk
[2] Technische Universität Darmstadt, Darmstadt, Germany

Abstract. The well-known Signed ElGamal scheme consists of ElGamal encryption with a non-interactive Schnorr proof of knowledge. While this scheme should be intuitively secure against chosen-ciphertext attacks in the random oracle model, its security has not yet been proven nor disproven so far, without relying on further non-standard assumptions like the generic group model. Currently, the best known positive result is that Signed ElGamal is non-malleable under chosen-plaintext attacks. In this paper we provide some evidence that proving Signed ElGamal to be CCA secure in the random oracle model is hard. That is, building on previous work of Shoup and Gennaro (Eurocrypt'98), Seurin and Treger (CT-RSA 2013), and Bernhard et al. (PKC 2015), we exclude a large class of potential reductions that could be used to establish CCA security of the scheme.

1 Introduction

Indistinguishability under chosen-ciphertext attacks (IND-CCA, or CCA for short) is widely considered to be the appropriate security notion for public-key encryption. Most known CCA-secure public-key schemes are built from a basic IND-CPA scheme (like ElGamal) and a non-interactive proof system. Examples of this approach include Cramer-Shoup [7], TDH2 [16], and the Chaum-Pedersen-Signed ElGamal scheme of Seurin and Treger [15], but also more theoretical constructions like the DDN encryption scheme [8] fall under this paradigm.

The difference between IND-CPA and IND-CCA security is that the latter notion allows the adversary to see decryptions of ciphertexts via a decryption oracle. Informally, "encrypt-then-prove" schemes require an adversary to prove knowledge of a plaintext as part of a valid ciphertext. But then a decryption oracle which the adversary can only call for ciphertexts on messages that she already knows is, intuitively, redundant. Hence, the encrypt-then-prove should reduce CCA-security to IND-CPA of the basic scheme. Interestingly, this intuition appears to be hard to turn into a formal proof, as we discuss for the case of the Signed ElGamal encryption scheme.

1.1 Signed ElGamal

Signed ElGamal [14,16] is a well-known encrypt-then-prove scheme combining ElGamal encryption with a Fiat-Shamir-Schnorr proof [9,13] of the randomness

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 47–69, 2016.
DOI: 10.1007/978-3-662-49384-7_3

used to encrypt (from which you can recover the plaintext from a ciphertext). It is the most efficient encrypt-then-prove scheme known to date both in terms of ciphertext size and computation cost. Further, it is submission-secure [18] (i.e., one can work homomorphically with the "core" ElGamal ciphertexts) and publicly verifiable, making it suitable for applications such as electronic voting[1]. All these properties would make Signed ElGamal the "primary choice" for a CCA encryption scheme, unless one objects to the Random Oracle Model (ROM) methodology [3]. Remarkably, however, Signed ElGamal has never been proven to be CCA-secure, even in the Random Oracle Model!

Shoup and Gennaro [16] were the first to consider the security of Signed ElGamal. They found that the "obvious" proof strategy to show CCA-security based on the encrypt-then-prove intuition did not work and gave a concrete example why the common strategy fails, but neither proved nor disproved CCA-security of the scheme. Instead, they developed the slightly less efficient TDH2 scheme which does come with a CCA proof. Schnorr and Jakobsson [14] proved Signed ElGamal to be CCA-secure under a combination of the ROM and the generic group model (GGM). Tsiounis and Yung [17] gave yet another proof under a non-standard "knowledge assumption" that resembles the approach behind the GGM.

More abstractly, the two mentioned proofs of CCA-security of Signed ElGamal [14,17] both rely on variants of a property known as plaintext awareness, which together with IND-CPA security suffices to show CCA; this property is in fact strictly stronger than CCA-security [4]. Plaintext awareness requires the existence of a plaintext extractor who, given some trapdoor key, can extract plaintexts from ciphertexts in an "online" manner, i.e., without interacting further with the party who created the ciphertext. However, in 2013, Seurin and Treger [15] showed that a plaintext extractor for Signed ElGamal in the ROM, without extra assumptions such as the GGM, could not exist, unless plain ElGamal is already insecure (i.e. one can solve the Computational Diffie-Hellman (CDH) problem in the underlying group). This result calls into question the proofs based on plaintext awareness as a route to show CCA-security of Signed ElGamal.

We stress again that Seurin and Treger, like Shoup and Gennaro, did not prove or disprove the CCA-security of Signed ElGamal in the ROM. Their result only rules out proofs based on plaintext awareness. Also, the recent result of Bernhard et al. [5], showing that Fiat-Shamir-Schnorr proofs of knowledge are not adaptively secure, only gives a limited answer about the CCA-security of Signed ElGamal. Their result relies on the fact the knowledge extractor has to return the full witness (i.e., the randomness for Signed ElGamal), whereas a clever CCA-to-CPA reduction only needs to simulate the decryption oracle, returning the message as a fraction of the full witness.

In this paper we provide further evidence against the CCA security of Signed ElGamal, even if one takes a direct route, without going through plaintext awareness. To this end, we rule out a large class of common proof techniques. The obstacle encountered by Shoup and Gennaro seems to be very solid indeed.

[1] The Helios [2] voting scheme used by the IACR uses a variant of Signed ElGamal.

1.2 State of the Art

We summarise previous work on the security of Signed ElGamal in the table in Fig. 1, ordered by the strength of the model.

paper	result	model
[6]	NM-CPA secure	ROM
[16]	obvious CCA proof fails	ROM
[15]	not PA2 unless CDH easy	ROM
[5]	no adaptive extractor under OMDL	ROM
this work	**no CCA reduction under IES**	**ROM**
[1]	CCA	ROM + algebraic adv.
[14,17]	CCA	ROM+GGM

Fig. 1. Overview over security results for Signed ElGamal. Here, OMDL is the one-more discrete log problem, IES is our Interactive ElGamal-Schnorr assumption. In case of negative results, further restrictions on the reductions (such as black-box use of adversaries) may apply.

Although CCA security of Signed ElGamal has sometimes been claimed informally, the strongest formal result in the ROM to date [6] only shows the weaker notion of non-malleability (NM-CPA). If one extends the ROM to include either the generic group model [14], a generic knowledge assumption [17] or restricts to algebraic adversaries [1] then one can prove CCA security. Conversely, we know that in the plain ROM the "obvious" CCA proof fails [16]. Signed ElGamal cannot be ROM-PA2 plaintext aware unless CDH is easy [15] which rules out proofs based on non-rewinding extractors for the contained ZK proof. The strongest negative result to date [5] rules out any CCA proof based on adaptive extractors for the ZK proof, by showing that the proof scheme in question (Fiat-Shamir-Schnorr) is not adaptively secure unless the one-more discrete logarithm (OMDL) problem is easy.

Adaptive proofs of knowledge. To put our results in context, we outline and discuss the results of Bernhard et al. [5], explain their limits and how we improve on them. Their work identifies the notion of adaptive proofs of knowledge as a potential bottleneck towards proving IND-CCA security for Signed ElGamal: this notion is what seems to be necessary to make the intuition behind encypt-then-prove work, yet it is provably not achieved by its implementation based on Fiat-Shamir-Schnorr proofs in Signed ElGamal.

In a hypothetical CCA-to-IND-CPA reduction of Signed to plain ElGamal, when the adversary asks a decryption query on a ciphertext, the reduction rewinds the adversary to extract the plaintext. Shoup and Gennaro [16] considered an adversary who makes a chain of n ciphertexts, the plaintext of each one depending on the last ciphertext (e.g. through a hash function), then asks decryption queries in reverse order. The effect of this adversary is to make a

straightforward rewinding strategy take exponential time in n, as the reduction ends up re-rewinding the rewound adversaries each time.

Bernhard et al. [5] were the first to show that such an exponential expansion is unavoidable under certain conditions. A non-interactive proof of knowledge is, informally speaking, a construction in which you can extract a witness from a single proof given suitable powers (e.g. the ability to rewind the prover). Bernhard et al. proposed a notion of adaptive proofs in which the prover can make a sequence of proofs and the extractor must return the found witnesses to the prover. In this way, should the prover ever succeed in making a proof for which she does not know a witness, she gains knowledge from the extractor. The game is adaptive in the sense that the extractor must deliver the witness for the k-th proof before the prover prepares the $(k+1)$-st proof.

Bernhard et al. proved that (unlike a construction of Fischlin [10]) Fiat-Shamir-Schnorr proofs are not adaptively secure, unless the one-more discrete logarithm (OMDL) problem is easy in the group concerned. Specifically, any adaptive extractor must either take at least 2^n time on an adapted version of Shoup/Gennaro's adversary, or reduce to solving OMDL.

Their results rules out a proof of IND-CCA security for Signed ElGamal which considers the basic encryption scheme and the non-interactive proofs in isolation, such as one might do following the encrypt-then-prove intuition. The intuition is that the reduction would not be able to answer decryption queries by relying on an extractor for the Fiat-Shamir-Schnorr proofs, as such an extractor does not exists.

Strictly speaking however, their result only rules out an extractor that obtains the randomness used in the ciphertexts and not one that somehow obtains only the underlying plaintexts. Yet, there is a significant complexity gap between these two problems: finding the plaintext is equivalent to solving a CDH problem (with the aid of rewinding) whereas finding the randomness (again with the aid of rewinding) is equivalent to taking a discrete logarithm. This means that the result outlined above does not rule out all plausible reductions. In addition, the result does not immediately apply to the combination of ElGamal ciphertext and proof that makes up Signed ElGamal.

1.3 Our Contribution

We narrow the gap between the positive and negative results by showing that, in the ROM, one cannot construct any black-box key-passing reduction from CCA-security of Signed ElGamal to IND-CPA security of plain ElGamal unless Schnorr proofs are insecure (specifically, they can help you solve CDH). We view this result as strong evidence in favour of the hypothesis that Signed ElGamal is *not CCA secure* in the ROM.

Technically, we show a metareduction whose starting point is any reduction from the IND-CCA security to IND-CPA of ElGamal, where the reduction makes only black-box use of the adversary and which is key-passing in the sense that it hands the public key in the ElGamal scheme to the adversary. Our metareduction turns such a reduction into an algorithm against an assumption which we call

the "Interactive ElGamal-Schnorr" assumption, or IES in short. Informally, the IES assumption is the following.

> You are given an ElGamal public key and a ciphertext on an unknown, random message. You can play the verifier in a single interactive Schnorr proof of the randomness in the ciphertext. Then you cannot extract the encrypted message.

We remark that we are *not* proposing a new assumption for the purpose of giving a cryptosystem that is secure under this assumption. Instead, we are showing that an already well-known cryptosystem cannot be proven CCA secure unless a plausible assumption is actually *false* — in which case we would be distrustful of any cryptosystem employing Schnorr proofs. Since IES is closely related to CDH, we would also have concerns about the use of any ElGamal-based scheme in a group in which IES is easy.

1.4 Outline of This Work

We begin by recalling the definition of Signed ElGamal and the IND-CPA/CCA notions for encryption. We then present and justify the IES assumption and prove that for any group, if there is an efficient key-passing reduction from CCA of Signed ElGamal to IND-CPA of ElGamal then IES is efficiently breakable in the group concerned. Our result even shows that proving CCA1 security, where the adversary makes all decryption queries *before* learning the challenge ciphertext, is hard.

2 Preliminaries

2.1 Cryptographic Groups

A cryptographic group is a group G of some prime order q together with a designated generator g, in which one can perform the group operation and inversion efficiently. It follows that one can also efficiently exponentiate in such groups. Typical examples (that have interesting security properties) are subgroups of the multiplicative group $\mathbb{Z}_p^\times$ for primes p and groups derived from elliptic curves over finite fields.

2.2 Public-Key Encryption and ElGamal

A public-key encryption scheme consists of three algorithms: KeyGen which produces a public and a secret key, $\mathsf{Encrypt}$ which takes a message and a public key and produces a ciphertext and $\mathsf{Decrypt}$ which takes a secret key and ciphertext and produces either a message or the symbol $\perp$ to indicate failure. Decryption is deterministic. If you generate a key pair, encrypt a message with the public key then decrypt the ciphertext with the matching secret key then you get the same message back.

The ElGamal encryption scheme over a group G (generated by g, of order q) has key pairs of the form (g^x, x) for $x \in \mathbb{Z}_q$; to generate a secret key one picks a random integer x modulo q. To encrypt a message $m \in G$ to public key $y \in G$, pick a random $r \in \mathbb{Z}_q$, your ciphertext is $(g^r, m \cdot y^r)$. To decrypt a ciphertext (c, d) with secret key x compute d/c^x.

The IND-CPA and IND-CCA security notions are given by the following game. To begin, the game generates a key pair and returns the public key. Once in the game, you may pick two messages (m_0, m_1) of the same length[2] in response to which the game picks $\beta \in \{0, 1\}$ randomly and gives you a challenge encryption c^* of m_β. In the CCA version of the game only, you may ask the game to decrypt any ciphertext for you, as often as you like and both before and after obtaining the challenge — except that after obtaining the challenge c^*, you may not ask for c^* itself to be decrypted. Your aim is to guess β. Your success probability σ is the probability that you guess β correctly (taken over all random choices made by the game) and your advantage α is defined as $2\sigma - 1$, so a perfect guesser has advantage 1 and a uniform random guesser has advantage 0.

For a sequence of groups G_λ indexed by a security parameter $\lambda \in \mathbb{N}$, the ElGamal encryption scheme is said to be (asymptotically) IND-CPA/CCA secure if the advantage of any efficient adversary (who receives λ as input in unary notation) in the corresponding game over group G_λ is negligible as a function of the parameter λ.

ElGamal (a.k.a. plain ElGamal) is IND-CPA secure under the DDH assumption: given a pair (g^x, g^y) of uniformly random and independent group elements it is hard to tell g^{xy} from another independent, uniformly random group element g^z. More precisely, there is a reduction from breaking IND-CPA of ElGamal to solving DDH that succeeds 1/2 of the time. Plain ElGamal is not CCA secure.

2.3 Schnorr Proofs

Over a cryptographic group G with designated generator g and order q, the Schnorr proof scheme is a protocol for a prover to convince a verifier that he knows a secret x such that $y = g^x$, where y may be known to the verifier in advance. The prover picks a random $a \in \mathbb{Z}_q$ and sends y and g^a to the verifier who replies with a challenge c drawn randomly from $\mathbb{Z}_q$. The prover answers with $s = a + cx \pmod q$ and the verifier accepts if and only if $g^s = g^a \cdot y^c$.

The Fiat-Shamir-Schnorr protocol is a non-interactive version of the above. Instead of the verifier picking c, the prover picks it herself as $c = H(y, g^a)$ where H is a cryptographic hash function with codomain $\mathbb{Z}_q$. The prover sends the verifier a single message (y, g^a, s) and the verifier recomputes[3] c and performs the same check as in the interactive protocol. Fiat-Shamir-Schnorr requires the

[2] For ElGamal, the message space is the underlying group G and all group elements have the same length which can be described as "one element".

[3] A variant of the protocol has the prover send (y, c, s) which is often shorter as it consists of one group element and two integers instead of two group elements and one integer. This variant is identical to the protocol presented here for security purposes.

so-called Random Oracle Model (ROM) for its security analysis, which idealises the hash function as an oracle that both prover and verifier can call.

2.4 Signed ElGamal

Signed ElGamal combines plain ElGamal and a Fiat-Shamir-Schnorr proof in a construction that Bernhard et al. call encrypt-then-prove. We define the scheme formally here.

Definition 1. *Signed ElGamal is the following encryption scheme over a cryptographic group G of order q with generator g.*

KeyGen: Pick $x \in \mathbb{Z}_q$ uniformly at random and set $y = g^x$ for your public key. Your keypair is (y, x).

Encrypt: Your message m must be an element of G. Let y be the public key. Pick a random $r \in \mathbb{Z}_q$ and compute an ElGamal ciphertext $(g^r, y^r \cdot m)$. Then make a Fiat-Shamir-Schnorr proof: pick random $a \in \mathbb{Z}_q$, set $c = H(y, g^r, y^r \cdot m, g^a)$ and compute $s = a + cx \pmod q$. Your ciphertext is $(g^r, y^r \cdot m, g^a, s)$.

Decrypt: Given x and a ciphertext (u, v, b, s) compute $c = H(g^x, u, v, b)$ and check that $g^s = b \cdot u^c$. If this check fails, the ciphertext is invalid — return $\perp$. Otherwise decrypt $m = v/u^x$.

2.5 Metareductions

A cryptographic security definition often takes the form of a game: an algorithm with one interface and a notion of winning. Specifically, a scheme is secure if there is no efficient adversary (an algorithm with one interface, compatible with that of the game) such that if we connect the adversary to the game, the adversary wins (with more than a negligible chance).

A reduction from source problem (e.g. IND-CPA of ElGamal) to a target problem (e.g. DDH) is an algorithm with two interfaces, one for a source-problem adversary and one for the target-problem game. The aim of a proof by reduction is to show that for any adversary who could win the source game, the system obtained by composing the adversary and the reduction would win the target game. This system is itself an algorithm with one interface, which is compatible with the target game.

A metareduction is an algorithm with three interfaces. A proof by metareduction shows that there can be no reduction from a source problem S to a target problem T unless another problem U is already easy. The metareduction's first two interfaces are those of an S-adversary and a T-game; the third interface is compatible with the U-game. In a proof by metareduction, we take a hypothetical S-to-T reduction and connect its S and T interfaces to those of the metareduction. In other words, composing a S, T, U metareduction with a S, T reduction gives a system with one free interface of type U, and this whole system can be connected to the U-game.

A metareduction will typically simulate a perfect S-adversary. The accompanying proof will show that if the reduction wins the T-game given a perfect S-adversary, then the metareduction wins the U-game given the reduction. In most cases, security of the U-game should only hold against efficient adversaries, such that the metareduction is typically also required to obey this running time bound.

3 The IES Assumption

The interactive Schnorr proof scheme is known to be a correct, honest-verifier zero-knowledge proof of knowledge of a discrete logarithm. The non-interactive (Fiat-Shamir-Schnorr) version is "full" zero-knowledge in the ROM. We propose an assumption that we call IES (Interactive ElGamal-Schnorr) that looks at an interactive Schnorr proof on an ElGamal ciphertext for a random message. While weaker than assuming such a proof to be zero-knowledge, IES states the assumption that such a proof does not leak the encrypted message.

Suppose you are given an ElGamal public key $y = g^x$ and an encryption $(u, v) = (g^r, my^r)$ for a random group element m. In addition, you receive a Schnorr commitment g^a for a random a and can pick $c \in \mathbb{Z}_q$, in response to which you get $s = a + cr \pmod q$. The IES assumption is then that you cannot recover m.

It turns out that m is actually not required to state IES. Decrypting an ElGamal ciphertext is solving a CDH[4] instance, so we can state IES as a CDH variant directly:

Definition 2. *Given three uniformly random and independent group elements (g^x, g^r, g^a) in a cryptographic group G of order q with generator g, the IES problem is to compute g^{rx} (the CDH problem) where one, after receiving (g^x, g^r, g^a), may pick a single value $c \in \mathbb{Z}_q$ and learns $s = a + cr \pmod q$ as a one-time auxiliary information.*

This definition shows that IES is stronger than CDH since a CDH solver could break IES trivially. The justification that s should not help is the same one as for the interactive Schnorr proof: since a is uniformly random in $\mathbb{Z}_q$ and independent of r, x, if g^a were not provided then s would be uniform and independent of r, x itself and the problem would reduce to CDH. The IES assumption formalises the idea that giving out g^a as well, which is also independent of the CDH problem on r, x, should not help you either.

For IES adversaries who pick c independently of a, the IES assumption reduces to CDH with the help of a rewinding reduction. Given a CDH instance (g^x, g^r) one can pick a random g^a and run the adversary up to the point where she produces c, then pick a random s and set $h = g^s/(g^r)^c$ and rerun the adversary on (g^x, g^r, h). As long as the same c appears in the second run, the simulation is sound (in particular the adversary can verify that she got the correct s). This is of course exactly how one simulates Schnorr proofs to show honest-verifier

[4] Computational Diffie-Hellman: given random group elements g^x, g^y compute g^{xy}.

zero-knowledge of the protocol. Like for Schnorr proofs, the simulation argument breaks down if the adversary chooses c depending on g^a but there is no known attack to exploit this technique.

The difference between breaking IES and extracting a witness from a Schnorr proof is that the former requires only finding a particular group element whereas the latter involves recovering an integer (exponent). An adversary who can recover x from a Schnorr proof $(g^x, g^a, c, s = a + cx)$ can take discrete logarithms.

The result of Bernhard et al. on Fiat-Shamir-Schnorr shows that one cannot build a CCA–to–IND-CPA reduction for Signed ElGamal by extracting the witness (the encryption randomness) from the Schnorr proof in a ciphertext. However, the main task for such a reduction is to answer decryption queries, for which it suffices to recover the encrypted message (a group element).

4 Main Theorem

Our goal is to exclude reductions from CCA security of Signed ElGamal to IND-CPA security of plain ElGamal (equivalently, to DDH). We make three constraints on the class of reductions that we consider. First, we consider only efficient reductions, since an exponential-time reduction could exhaustively search the key-space. Secondly, we consider rewinding black-box reductions: our reductions may invoke any number of copies of the adversary as long as the reduction is efficient overall. Each invocation of the adversary counts as a single operation. All these copies of the adversary run with the same random string. The reduction is in charge of all communication to and from these copies, including random oracle calls. In particular the reduction can employ the usual "special soundness" forking strategies. All computation from the moment the reduction sends a message to a copy of the adversary up to the adversary's reply counts as a single operation as far as the reduction is concerned.

Finally, we consider only key-passing reductions. A reduction to IND-CPA receives a public key from the IND-CPA challenger whereas a CCA adversary expects a public key; keys for plain and Signed ElGamal are of the same form. A key-passing reduction is one that gives all copies of the adversary the same public key which it received from its challenger. Alternatively, one could view the public key as being made available globally to all parties (the reduction and the copies of the adversary) via the IND-CPA challenger.

Why key-passing? We provide some intuition and the technical reasons for the key-passing assumption. Due to the rewinding nature of the Schnorr protocol extractor, we must allow our reduction access to multiple copies of the adversary with full control over the random oracle. Yet we do not want to offer the reduction the option to substitute a key of its own (for which it may know the secret key) for some copies of the adversary, which would lead to the following problem: The reduction may first run multiple copies of the adversary under self-chosen keys and test if the adversary succeeds in predicting the challenge bit with sufficiently high probability, exploiting knowledge of the secret key for answering decryption queries in this part. Only if this test phase is over, it may start the actual reduction to the IND-CPA challenger's public key.

While an actual adversary would pass the test phase of the reduction above, any metareduction most likely will fail to reach the second phase. The reason is simply that it would need to efficiently break CCA security under the reduction's keys. More precisely, if the metareduction treats the reduction as a black box, then one could potentially even mount meta-metareduction techniques (i.e., now playing against the metareduction) as in [11] to base this argument on formal grounds. Still, it seems that this "testing" reduction is somewhat contrived, as it is not known how the test phase helps to break CPA security for the given key.

Technically, the chosen-key problem appears when our metareduction tries to inject an IES challenge into the reduction's view. The reduction, on input the challenger's public key pk, could both substitute a key of its own (for which it knows the secret key, but which is independent of the IES challenger) or the reduction could rerandomise pk by picking random r and returning pk^r. In this case the reduction cannot directly decrypt anything, but there is a dependency on the IES challenger's key. Intuitively, creating further keys of its own should not help the reduction to attack the challenger. But to a metareduction, both these tactics are indistinguishable: the resulting key looks random in both cases. If the metareduction injects an IES challenge into a ciphertext for which the reduction knows the secret key, all bets are off — the metareduction cannot simulate an adversary consistently anymore.

The solution to the above dilemma would be to somewhat grant the metareduction access to the reduction's self-chosen secretkeys. Note that it would not be sufficient to ask that each public key comes with a Schnorr signature of knowledge of its secret key (perhaps signed by the challenger) — the reduction controls the random oracle towards the adversary, so it could easily forge such signatures. But, in principle, other secure means of proofs of knowledge could help. Alternatively, switching to more transparent types of reductions such as algebraic or generic ones could also be a viable path.

It would be interesting to see whether the key-passing requirement could be weakened in future work. A more complicated argument (with looser concrete security bounds) may well succeed, but for now we prefer to work in the key-passing model.

Our main result is the following theorem that excludes a large class of attempts to prove Signed ElGamal CCA-secure. The proof of the following also reveals that showing even CCA1 security is hard.

Theorem 1. *Suppose that DDH and IES hold in a cryptographic group G. Then there is no efficient key-passing black-box reduction from CCA security of Signed ElGamal to IND-CPA security of plain ElGamal in G.*

5 The Proof

We will construct a metareduction to IES from any CCA–to–CPA reduction for Signed ElGamal. We introduce some variants of IES that will make the proof easier to present. We note that this does not introduce additional assumptions for our result: we show that they all reduce to IES.

5.1 Verifiable IES

First, we deal with the issue that decrypted messages are not "verifiable". Proofs of knowledge are usually taken over NP relations (e.g. discrete logarithm). However, the statement that a ciphertext decrypts to a particular message is not immediately verifiable — it would require either the secret key or the encryption randomness to verify.

Our metareduction will have to check the decryptions produced by the reduction with which it interacts. We introduce a new assumption that we call verifiable IES or vIES to give the metareduction this ability; we also show that vIES reduces to IES. The new feature of vIES is that the adversary gets many attempts at guessing the message; formally we introduce a new oracle for the adversary to check messages.

Definition 3 (vIES). *The vIES problem in a cryptographic group* $(\mathbb{G}, q, g)$ *is to solve IES given the extra ability to check candidate solutions. Given* (g^x, g^r, g^a) *one may once submit a value* $c \in \mathbb{Z}_q$ *and learn* $s = a + cr \pmod q$ *in return; in addition, one may query an oracle* $\mathsf{check}(m)$ *many times which returns* 1 *if and only if* $m = g^{rx}$*. One wins the game if one can find* g^{rx}*. A code-based presentation of the game is given in Fig. 2.*

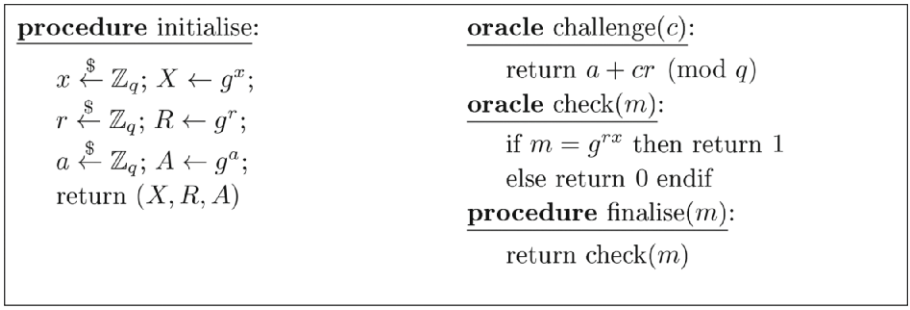


Fig. 2. Verifiable IES. The checking oracle allows the adversary to test candidate solutions before submitting one. Challenge may only be called once.

The vIES assumption reduces to the IES assumption with a loss in soundness of a factor $k+1$ where k is the number of checks made by the adversary. To see this, consider an efficient adversary with probability p of winning the vIES game and let k be a (polynomial) bound on the number of checks the adversary makes. Then with probability p, one of the following $k+1$ events occur: E_i for $1 \le i \le k$ is the event that the adversary makes at least i checking queries and the i-th check contains the correct message; E_0 is the event that the adversary never makes a checking query on the correct message but still calls the finalization oracle with the correct message.

Our reduction to IES guesses $i \xleftarrow{\$} \{0, 1, \ldots, k\}$ uniformly at random and simulates as follows: forward the initial data and the challenge query between

the adversary and IES challenger, for $i > 0$ answer the first $i-1$ checking queries with 0 and pass the result of the i-th checking query to the IES finalization oracle directly, aborting the adversary at this point. For $i = 0$ answer 0 to all the adversary's checking queries and forward the adversary's output to the finalization oracle. If the adversary does not make i checking queries or in case 0 makes no output, abort.

If event E_i occurs then the reduction for case i will break IES. Since we assumed the adversary to succeed with probability p, at least one of the events will occur with probability $p/(k+1)$ as the $k+1$ events are a partition of the event that the adversary succeeds. Since the reduction chooses i uniformly, we conclude that it succeeds against IES with probability $p/(k+1)$.

5.2 One-More Verifiable IES

For our metareduction we use a one-more variation of IES, for the same reason that Bernhard et al.'s proof that Fiat-Shamir-Schnorr is not adaptively secure requires the one-more discrete logarithm assumption. Unlike the cited theorem and assumption, the one-more IES assumption reduces to the basic one. We give the one-more assumption and reduction for verifiable IES; the same reduction holds for the non-verifiable variation.

The one-more assumption works as follows. The adversary may obtain and open a number of IES "instances"; her aim is to solve an unopened instance. The initialization oracle produces a "public key" g^x shared between all instances. The instance oracle creates a fresh pair (g^r, g^a) together with an internal flag $f = 0$ to denote that this instance is fresh. The adversary may issue a challenge c once per instance, to which the challenger replies with $s = a + cr$ and sets the flag to $f = 1$ to denote that the challenge for this instance has been provided. In addition, the adversary may ask for an instance to be opened to which the challenger responds with (r, a) and sets $f = 2$. As in vIES, the adversary may also ask to check a value m against an instance, in which case the challenger reveals if $m = g^{rx}$. Checking does not affect the flag f. The adversary wins by providing the value $m = g^{rx}$ on an instance that has not been opened, i.e. $f \leq 1$. The one-more verifiable IES game is asymmetric in that the adversary must only solve a CDH instance to win but the game must provide a discrete logarithm r on request. It is this asymmetry that makes our metareduction work. Nonetheless, one-more verifiable IES reduces to plain IES. In the code-based presentation of the game in Fig. 3, an index i is used to distinguish different instances.

Definition 4 (OMvIES). *The one-more verifiable IES game is given by the code in Fig. 3.*

The reader may be asking why they should have any confidence that an assumption as complex as OMvIES should be hard. We note that vIES and OMvIES derive their justification solely from the fact that they reduce to IES: they are intermediate steps to make our main proof easier, not assumptions in their own right that we ask anyone to believe in. The justification for basic IES we gave when we introduced it, that a single Schnorr proof should not completely break the security of

procedure initialise:
$j \leftarrow 0$; $T \leftarrow [\,]$;
$x \xleftarrow{\$} \mathbb{Z}_q$; $X \leftarrow g^x$;
return X

procedure instance:
$j \leftarrow j + 1$;
$r \xleftarrow{\$} \mathbb{Z}_q$; $R \leftarrow g^x$;
$a \xleftarrow{\$} \mathbb{Z}_q$; $A \leftarrow g^x$;
$f \leftarrow 0$;
$T[j] \leftarrow (r, a, f)$;
return (j, R, A)

procedure finalise(i, m):
if $i > j$ then return 0 endif;
$(r, a, f) \leftarrow T[i]$;
if $f > 1$ then return 0 endif;
return check(i, m)

oracle challenge(i, c):
if $i > j$ then return $\perp$ endif;
$(r, a, f) \leftarrow T[i]$;
if $f > 0$ then return $\perp$ endif;
$T[i] \leftarrow (r, a, 1)$;
return $a + cr \pmod q$

oracle check(i, m):
if $i > j$ then return 0 endif;
$(r, a, f) \leftarrow T[i]$;
if $m = g^{rx}$ then return 1
else return 0 endif

oracle open(i):
if $i > j$ then return $\perp$ endif;
$(r, a, f) \leftarrow T[i]$;
$T[i] \leftarrow (r, a, 2)$;
return (r, a)

Fig. 3. One-more verifiable IES. All IES instances share a common x but have their own r, a — which can be revealed using the open oracle.

ElGamal encryption. The reason that the one-more version reduces to the simple one is that the instances are independent in the sense that the adversary cannot perform a challenge query that "touches" more than one instance.

Lemma 1. *There is a reduction from OMvIES to IES that loses a factor $O(k^2)$ in soundness where k is a bound on the number of queries made by the adversary.*

Proof. It suffices to reduce OMvIES to vIES with a loss of $O(k)$. Given an upper bound k on the number of instances an adversary can create, pick $n \xleftarrow{\$} \{1, \ldots, k\}$ at random and use the vIES challenger for the n-th instance. Simulate all other instances by picking fresh (r, a). To open a simulated instance, simply reveal (r, a). To check a simulated instance against a candidate m, check if $m = X^r$. If the adversary tries to open the n-th instance, abort. If the adversary succeeds with probability p against OMvIES then she succeeds with probability at least p/k against the n-th instance, in which case she cannot have opened this instance. So the reduction wins the vIES game with at least p/k probability too. □

5.3 A Model Adversary

Let $\mathcal{R}$ be a rewinding, black-box, key-passing reduction from CCA security of Signed ElGamal to IND-CPA security of plain ElGamal. That is, $\mathcal{R}$ may invoke multiple copies of a CCA adversary A which expects to receive a public key, can make one challenge and many decryption queries and will output a guess bit.

$\mathcal{R}$ itself can interact with one IND-CPA challenger who provides a public key and a single challenge query, which returns a plain ElGamal ciphertext.

The aim of $\mathcal{R}$ is to guess its challenger's bit β. We first construct an inefficient adversary A that breaks CCA of Signed ElGamal with advantage 1, that is it guesses correctly all the time. Our adversary A will operate in three phases: phases 1 and 2 are efficient and if a reduction $\mathcal{R}$ advances our adversary to phase 3 then $\mathcal{R}$ must have already broken an assumption (IES or DDH) itself or launched exponentially many copies of the adversary. We also show how to construct an efficient simulation of (multiple copies of) A under these conditions, yielding our metareduction. Thus, using an inefficient adversary in the first place does not cause triviality problems.

Suppose w.l.o.g. that $q > 5$ and consider the inefficient adversary A_n in Fig. 4 where $\Psi : \mathbb{Z}_q[X] \to \mathbb{Z}_q$ is a random function[5] (since efficiency is not an issue, random functions exist). RO is a random oracle call and dlog takes a discrete logarithm (which an inefficient adversary can also do). Decrypt and Challenge are calls to the CCA challenger.

Adversary A_n runs in three phases. In phase 1, it builds up a chain of n Signed ElGamal ciphertexts in such a way that the randomness used in each ciphertext depends on the challenge returned from the random oracle in the previous one. Indeed, A_n only draws one random value to initialise S and uses

```
// input: public key Y
S ←$ Z_q; T ← [ ]

// PHASE 1 //
for i = 1...n do
    r ← Ψ(1, S)
    a ← Ψ(2, S)
    m ← Ψ(3, S)
    M ← g^m
    (C, D) ← (g^r, MY^r)
    A ← g^a
    c ← RO(Y, C, D, A)
    s ← a + cr (mod q)
    S ← (S, c)
    T[i] ← (M, C, D, A, s)
endfor

// PHASE 2 //
for i = n...1 step (−1) do
    (M, C, D, A, s) ← T[i]
    M' ← Decrypt(C, D, A, s)
    if M ≠ M' then abort endif
endfor

// PHASE 3 //
m_0 ← Ψ(4, S)
m_1 ← Ψ(5, S)
(C, D, A, s) ← Challenge(m_0, m_1)
r ← dlog(C)
M ← D/Y^r
if M = g^{m_0} then return 0
else return 1 endif
```

Fig. 4. Adversary A_n against CCA of Signed ElGamal with advantage 1.

[5] By choosing the polynomial ring over $\mathbb{Z}_q$ as the domain, we mean that Ψ takes arbitrary-length finite sequences of integers modulo q as input.

the random function Ψ to update its state afterwards. One can think of S as the current state of a internal pseudorandom number generator.

In phase 2, our adversary asks decryption queries in reverse order, in the manner first proposed by Shoup and Gennaro [16] and used by Bernhard et al. [5]. Crucially, our adversary checks the correctness of each decryption and aborts if the CCA game resp. reduction to which it is connected tries to cheat by returning a false decryption. By the time our adversary reaches phase 3, it is "satisfied" that whoever it is interacting with really can decrypt Signed ElGamal ciphertexts. It picks two random messages, asks a challenge query and takes a discrete logarithm to win the CCA game with overwhelming probability[6].

Our proof strategy will be to give an efficient simulation of phases 1 and 2 (which means dealing with Ψ) and to argue that no copy of A_n will ever reach phase 3 in less than exponential time, unless the reduction solves IES or DDH.

In the proof we will make three case distinctions. Recall that $\mathcal{R}$ is a reduction from CCA of Signed ElGamal to IND-CPA of plain ElGamal.

1. $\mathcal{R}$ answers the IND-CPA challenger's query without any copy of the adversary reaching Phase 3. In this case, we can simulate all copies of the adversary by lazily sampling the random function Ψ to obtain an IND-CPA adversary that wins its game with the same probability as $\mathcal{R}$ given access to a CCA adversary that always guesses correctly.
2. $\mathcal{R}$ answers a decryption query on a ciphertext without using special soundness. We build a metareduction to IES.
3. Neither of the above cases occur. In this case one copy of the adversary we are simulating proceeds to the point where it would have to use its discrete logarithm capability hence it must have got answers to all n decryption queries. In this case we show that the reduction must have launched $\Omega(2^n)$ copies of the adversary.

5.4 Case 1: The Reduction Solves DDH by Itself

If the reduction answers its IND-CPA challenge without getting any copy of the adversary to run to phase 3 then the reduction must be breaking indistinguishability "by itself". In this case we can just simulate the adversary efficiently for as long as needed.

Lemma 2. *Let E_1 be the event that the reduction $\mathcal{R}$ returns a guess to its challenger without any copy of the adversary reaching Phase 3. There is a metareduction M_1 that breaks DDH in $\mathbb{G}$ with advantage $\alpha_M = \Pr[E_1]\alpha_{E_1}/2$ where α_{E_1} is the advantage of $\mathcal{R}$ (with access to our adversary) given that E_1 has occurred.*

Proof. Consider an efficient metareduction M_1 which simulates all the copies of our adversary in phases 1 and 2 and the random function by lazy sampling, once for all copies of the adversary. If an adversary copy reaches phase 3 or $\mathcal{R}$ aborts, M_1 outputs a random guess. Writing $\sigma_{E_1} := \Pr[\mathcal{R}$ guesses correctly $\mid E_1]$

[6] The probability is not exactly 1 because m_0 and m_1 could collide.

and $\alpha_{E_1} := (2\sigma_{E_1} - 1)$ we compute the advantage of M_1 as $\Pr[E_1] \cdot \alpha_{E_1}$. The advantage against the encryption scheme gives an adversary against DDH with advantage $\Pr[E_1]\alpha_{E_1}/2$. □

5.5 Case 2: The Reduction Breaks IES

If the reduction $\mathcal{R}$ does run a copy of the adversary to phase 3, we can hope that it solves IES for us along the way. We define a metareduction M_2 that simulates individual copies of A_n as follows, with joint state between the copies in two global variables U, V. All other variables are local to each simulated copy of the adversary.

```
// COPY OF A_n //

// PHASE 1 //
S ← 0
for i = 1 ... n do
    (R, A, d) ← draw(S)
    c ← R.RO(X, R, g^d, A)
    s ← chal(R, A, c)
    S ← (S, c)
    T[i] ← (R, A, d, c, s)
endfor

// PHASE 2 //
for i = n ... 1 step (−1) do
    (R, A, d, c, s) ← T[i]
    M ← R.decrypt(R, g^d, A, s)
    Z ← g^d / M
    if not check(R, A, Z) then
        abort this copy of A_n
    endif
endfor

oracle check(R, A, Z):
    (j, φ, c', s', r', a') ← V[R, A]
    β ← I.check(j, Z)
    if β = 1 and φ < 2 then
        abort and return (j, Z)
        to OMvIES challenger
    else
        return β
    endif

oracle draw(S):
    if U[S] is defined then
        (R, A, d) ← U[S]
        return (R, A, d)
    else
        (j, R, A) ← I.Instance()
        d ←$ Z_q
        U[S] ← (R, A, d)
        V[R, A] ← (j, 0, 0, 0, 0, 0)
        return (R, A, d)
    endif

oracle chal(R, A, c):
    (j, φ, c', s', r', a') ← V[R, A]
    if φ = 0 then
        s ← I.Challenge(j, c)
        V[R, A] ← (j, 1, c, s, 0, 0)
        return s
    elseif φ = 1 then
        if c = c' then // replay
            return s'
        else // fork
            (r, a) ← I.open(j)
            V[R, A] ← (j, 2, c', s', r, a)
            return a + cr (mod q)
        endif
    else // φ = 2
        return a' + cr' (mod q)
    endif
```

Our metareduction M_2 simulates both the adversary and challenger interfaces towards the reduction $\mathcal{R}$ and interacts with an OMvIES challenger. On the challenger interface M_2 passes the challenger's public key. By assumption, $\mathcal{R}$ is key-passing so although the simulated adversaries formally receive a public key from $\mathcal{R}$ we could equally well have the metareduction provide them with this key directly. If the reduction asks an IND-CPA challenge query, we just simulate this challenge query (picking a random bit b); since we have already dealt with case 1 we can ignore the reduction returning a guess to the challenger for now.

In detail, our metareduction operates as follows. Initially, it obtains a value X from its OMvIES challenger and hands control to the reduction $\mathcal{R}$. When $\mathcal{R}$ asks to invoke a new copy of the adversary A, metareduction M_2 simulates a copy of A using public key X (since $\mathcal{R}$ is key-passing) using the algorithms in the code listing above. The oracles check, draw and chal are shared between all copies of the simulated adversary. R.alg means we call back to $\mathcal{R}$, simulating the adversary calling its challenger's oracle named alg whereas I.alg means call the oracle named alg on the OMvIES challenger.

If the reduction makes a challenge query (to its IND-CPA challenger) the metareduction draws a random bit b and simulates the challenge ciphertext; if the reduction $\mathcal{R}$ makes a guess at b then the metareduction aborts (this is case 1 which we have dealt with above). If $\mathcal{R}$ manages to get a copy of the simulated adversary to phase 3, the metareduction M_2 aborts too — this is case 3 which we will deal with later.

The check, draw and chal oracles help the metareduction M_2 simulate multiple copies of A_n using only one OMvIES challenger. The draw oracle ensures that multiple copies of the adversary who receive identical (random oracle) replies from $\mathcal{R}$ also produce identical ciphertexts. In a table U the metareduction keeps track of whether a particular adversary state S has been encountered before; if so we can simply replay the same ciphertexts.

The chal oracle is responsible for completing the proofs in Signed ElGamal ciphertexts. The table V maps each OMvIES instance (R, A) to the following parameters:

- The integer j is the index required to tell the challenger to operate on this particular instance.
- The potential ϕ is the equivalent of the OMDL potential in Bernhard et al. 's proof [5] and matches the potential f stored internally by the OMvIES challenger.
  - The first time a particular instance (R, A) is used (case $\phi = 0$), chal uses the OMvIES challenge oracle to complete the proof.
  - In case $\phi = 1$, if the current challenge has been used before then the reduction is replaying one adversary copy's responses to a second copy. In this case the metareduction replays the response s that it computed earlier. If c is fresh on the other hand, then the reduction has "forked" two copies of the adversary on the random oracle call in this proof and is about to recover the discrete logarithm r by applying special soundness. In this case our metareduction M_2 opens the instance.

  - In case $\phi = 2$ the instance has already been opened, so M_2 knows the values a, r necessary to make the proof itself.
- The values c', s' in a V-entry store the challenge and response from a previous chal query. These values are used in case $\phi = 1$ and the reduction replays the same c', in which case the metareduction replies with the same s'.
- The values r', a' store the discrete logarithms of R, A when $\phi = 2$. In this case the reduction has forked the adversary on the instance (R, A), forcing the metareduction to open the instance.

The check oracle is responsible for checking both that the reduction does not cheat and whether the reduction has solved OMvIES for us. When the reduction returns a decryption M to a copy of the adversary, the simulated adversary strips out the message to recover what would be the CDH solution g^{rx} for the instance in question. The metareduction M_2 then checks this with the OMvIES challenger. Should the decryption turn out to be false, the copy of the adversary in question aborts. If the decryption is correct and the potential is not yet at 2 then the reduction has given us some information that we do not know already (the instance in question is unopened) and we solve OMvIES.

The above arguments show that whenever $\mathcal{R}$ decrypts an unopened challenge instance, the metareduction M_2 breaks OMvIES. It remains to show that $\mathcal{R}$ cannot distinguish M from multiple, independent copies of A_n running on the same random string. Recall that in Fig. 4 we have the following invariants.

1. Two copies of A_n that receive identical messages from $\mathcal{R}$ also produce identical messages/calls back to $\mathcal{R}$.

   This is because all copies execute on the same random string and do not communicate with anyone except $\mathcal{R}$.
2. $\mathcal{R}$ can influence copies of A_n in exactly three places: answering random oracle queries in phase 1, decryption queries in phase 2 and the challenge query in phase 3.
3. The moment that two copies of A_n get different answers to a random oracle query, the two copies become independent of each other.

   If at some point two copies U, V get different answers $c_U \neq c_V$ to the same random oracle query then their states S_U, S_V will become distinct from then on and never coincide again (since we only ever append to state vectors). Since the randomness used to construct Signed ElGamal ciphertexts is drawn using a random function Ψ from the current state S, the ciphertexts in two copies with different states are independent.
4. The distribution of each individual ElGamal ciphertext and Schnorr commitment produced by A_n is uniform, that is (C, D, A) is a uniformly random element of $\mathbb{G}^3$.

   This follows from r, a, m being drawn by a random function on distinct inputs.
5. In phase 2, a copy of A_n will proceed past a decryption query (and not abort) if and only if the decryption is correct.

These invariants will let us show that the values received by $\mathcal{R}$ when interacting with M_2 or multiple copies of A_n are identically distributed. We use induction over the sequence of all calls made to $\mathcal{R}$ in a particular execution. Before the first call, the distributions of all values sent to $\mathcal{R}$ are certainly equal.

- For a RO call, there are two cases. If this call is made by a copy of the adversary that has received the exact same sequence of inputs and outputs as some other copy has received previously, then it will return the same values (X, C, D, A) as the previous copy.

  This holds for A_n as the state S of the copy that sent the current call will match the state S' of the previous copy at the time it sent the equivalent call, so the values C, D, A will be equal (and X is constant in any case).

  In M_2, the oracle draw ensures that the same state S leads to the same values (R, A, d) being returned.
- For a fresh RO call (that does not match the case above), the value X is constant and the values R, D, A are uniformly random and independent of each other and all values sent to the reduction $\mathcal{R}$ so far.

  In A_n this holds because S is fresh and the values in question are therefore obtained by a random function on distinct, fresh inputs (since m is uniform, so is M and because M is not used elsewhere, so is MY^r). In M_2 a fresh S causes (R, A) to be sampled from the OMvIES challenger so they are uniform and independent of previous values as expected; D is also a fresh, uniform group element.
- The value s in a decryption query is completely determined by the matching C, D, A and c — all of which $\mathcal{R}$ has seen before in the matching random oracle query, or in the case of c the reduction $\mathcal{R}$ has chosen the value itself. This holds in both A_n and M_2.

It follows that up until some adversary copy reaches phase 3, $\mathcal{R}$ cannot tell M_2 from A_n and must therefore have a negligibly close IND-CPA advantages in both experiments.

5.6 Case 3: The Reduction Takes Exponential Time

This case is essentially the same argument as that of Bernhard et al. [5]. If the reduction $\mathcal{R}$ when interacting with M_2 ever gets a copy of the simulated adversary to phase 3 (in which case M_2 aborts) then it must have launched at least 2^n copies of the adversary.

Lemma 3. *Consider an execution of M_2 with any reduction $\mathcal{R}$ that results in one copy of the simulated adversary advancing to phase 3. Then $\mathcal{R}$ must have launched 2^n copies of the adversary.*

If a copy of the adversary simulated by M_2 advances to phase 3, we know that neither has $\mathcal{R}$ returned a guess at β (this would have halted the entire execution as in Case 1) nor has the check oracle aborted because OMvIES has

been solved (Case 2). In particular, $\mathcal{R}$ has never answered a decryption query on a ciphertext linked to an OMvIES instance at potential $\phi \leq 1$.

We build a complete binary tree of depth n representing points in the execution of $\mathcal{R}$ with our metareduction where the adversary must have been "forked" on Schnorr proofs. Our aim is to show that each leaf of the tree must reference a distinct copy of the adversary, hence there must have been at least 2^n copies overall launched by $\mathcal{R}$. We first give the invariants of our tree and prove that these imply distinct adversary copies in the leaves. Then we will construct a tree meeting these invariants from any execution that reaches phase 3.

The nodes in our tree have labels (i, k) where i is an identifier for some copy of the adversary (for example, one can number the copies in the order that they begin phase 1) and $k \leq n$ is an integer referencing a particular decryption query. Our nodes will have the following invariants.

1. Any copy of the adversary referenced in the tree has advanced to at least phase 2 and has obtained all its n challenges. If a node (i, k) is present then copy i has also obtained answers to at least its first k decryption queries,
2. The root of the tree is of the form (i, n). A child of (i, k) is of the form $(j, k-1)$ and a descendant of (i, k) is of the form (j, l) with $0 \leq l < k$.
3. If (j, l) is a descendant of (i, k) then (1) the copy j has got the same first $n - k$ challenges as copy i. (The two could also be identical.) However, (2) the copies $(j, k - 1)$ and $(j', k - 1)$ represented by the two children of (i, k) differ in challenge $n - k + 1$.

Recall that our adversary A_n performs decryption queries in reverse order after it has got all n challenges, so the first decryption query uses the n-th challenge etc. This explains the reversed indexing $n - (k - 1)$ of the referenced challenges for the $(*, k - 1)$ nodes in the last property.

If we can construct such a complete binary tree rooted at (i, n) where i is the copy of the adversary that reached phase 3 then we claim that all 2^n leaves of this tree represent distinct copies of the adversary, proving our exponential lower bound. Suppose for the sake of contradiction that two distinct leaves $L = (j, 0)$ and $M = (j', 0)$ refer to the same copy of the adversary, i.e. $j = j'$. Then consider the unique path from the one leaf to the other in the tree, and the highest (i.e. closest to the root) node $R = (i, k)$ on this path. R will have two children A and B, since R is not itself a leaf by construction. W.l.o.g. L is a descendant or equal to A and M is a descendant or equal to B. The contradiction is that by invariant 3, A and B must differ in their $n - k + 1$st challenge (part 1 of the invariant) whereas all descendants of A, including L, must share their $n-k+1$st challenge with A (by applying part 2 of the invariant to A). Similarly, M must share challenge $n - k + 1$ with B and therefore L, M must differ in challenge $n - k + 1$. It follows that $j \neq j'$ and that there must be 2^n distinct copies of the adversary referenced in the leaves, hence $\mathcal{R}$ must have launched this many copies.

To construct the tree from an execution that reaches phase 3, we pick the copy i of the adversary A_n that reached phase 3 and use (i, n) as the root; this

trivially meets all invariants. We repeatedly give each node (j, l) with $l > 0$ two children as follows. The first child of (j, l) is simply $(j, l-1)$. Invariant 1 carries over as the second component of the node decreases, invariants 2 and 3(1) are trivially satisfied.

The core of the tree construction is in the choice of the second child for each node. For the second child of (j, l) with $l > 0$ we observe that since copy j has got an answer to its l-th decryption query yet M_2 has not solved OMvIES with this answer, the corresponding IES instance must be at $\phi = 2$. Therefore some other copy j' of the adversary must have triggered the opening of this instance, before copy j got its l-th decryption query answered. This other copy j' must therefore have shared challenges 1 up to $n - l$ with j and been "forked" on challenge $n - l + 1$ to open the IES instance in question. And this forking can only have happened after j' had its own $l-1$st decryption query answered, since it must have been the lth decryption query of j' that triggered the opening. It follows that we can pick $(j', l-1)$ as our second child of (j, l) to satisfy all the invariants.

Taken together, our three cases show that a key-passing black-box reduction $\mathcal{R}$ from IND-CPA security of Signed ElGamal to IND-CPA security of plain ElGamal must either solve DDH, or IES, or run in exponential time. This proves our Theorem 1. □

6 Conclusion

CCA security is often presented as the correct notion for public-key encryption and Signed ElGamal is very tempting to use due to its short ciphertexts and fast computation. However, Signed ElGamal has never been proven CCA secure in the plain ROM (without algebraic or generic-group assumptions).

Our results do not disprove CCA security of Signed ElGamal in the plain ROM nor yield an attack against CCA of Signed ElGamal in typical implementations. What they do is further limit the techniques available to anyone wishing to prove CCA security. Where Shoup and Gennaro [16] showed that the obvious proof does not work and Bernhard et al. [5] excluded proofs based on extracting the Schnorr proof's randomness, which seems to us to be overly strong — even the honest decryptor holding the secret key cannot learn the randomness without taking a discrete logarithm — we exclude all proofs by reduction to IND-CPA of plain ElGamal that do not make use of at least one non-standard step, such as treating the adversary in a non-black box manner. We would recommend caution before using Signed ElGamal in a scenario where CCA security is really called for (if NM-CPA is sufficient, so is Signed ElGamal).

At this point, our result works for key-passing reductions only. Our metareduction to IES requires the reduction to launch its adversary copies with the same public key that it got from its (simulated) IND-CPA challenger. In particular, if instead of reducing to IND-CPA of ElGamal, one wishes to reduce to a problem such as DDH directly, there is no notion of a key anymore so such

reductions seem not to be covered[7]. Key-passing seems to be a common way to build a reduction to the CPA security of ElGamal, and has been used in the proofs of previous results on Signed ElGamal [14,17] with some additional knowledge assumption. Alternatively, one may consider the implications of restricting to algebraic or generic reductions — unlike algebraic adversaries, this seems a sound choice to us as there do not seem to be any non-algebraic reductions in discrete-logarithm based schemes. Potentially, this could not only eliminate the key-passing requirement but also show an impossibility of a reduction to any "natural" problem over groups as in Fleischhacker et al. [12]. We will investigate this problem in future research.

Another interesting question is whether IES is hard in the generic group model. Our best answer at the moment is that IES is out of scope of the generic model: as defined by Shoup, the model allows an adversary to start with arbitrary information and perform generic computations on the group elements in the adversary's input, but it does not allow for information relating to the adversary's group inputs to be revealed adaptively during the execution. This is exactly what IES does and it seems to us that one would have to extend the model to capture this, leading to the question how one would validate such a new model.

Acknowledgments. We thank the anonymous reviewers for their comments. Marc Fischlin is supported by the Heisenberg grant Fi 940/3-2 of the German Research Foundation (DFG). This work has been co-funded by the DFG as part of project P2 within the CRC 1119 CROSSING.

References

1. Abdalla, M., Benhamouda F., MacKenzie P.: Security of the J-PAKE password-authenticated key exchange protocol. In: IEEE Symposium on Security and Privacy 2015, pp. 571–587, IEEE Computer Society, May 2015
2. Adida, B.: Helios: web-based open-audit voting. In: 17th USENIX security symposium, pp. 335–348 (2008). Helios website: http://heliosvoting.org paper: http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
4. 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, p. 26. Springer, Heidelberg (1998)
5. Bernhard, D., Fischlin, M., Warinschi, B.: Adaptive proofs of knowledge in the random oracle model. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 629–649. Springer, Heidelberg (2015)

[7] We thank an anonymous reviewer from a previous draft of this paper for commenting on this point. Interestingly, if such a reduction were to exist, the underlying hard problem would either have to not reduce to IND-CPA of plain ElGamal itself, or the combination of reductions (Signed ElGamal to hard problem to IND-CPA of ElGamal) would itself have to be non-key-passing to avoid our impossibility result.

6. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: pitfalls of the Fiat-Shamir Heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer, Heidelberg (2012)
7. 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, p. 13. Springer, Heidelberg (1998)
8. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000)
9. 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)
10. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
11. Fischlin, M., Fleischhacker, N.: Limitations of the meta-reduction technique: the case of Schnorr signatures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 444–460. Springer, Heidelberg (2013)
12. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 512–531. Springer, Heidelberg (2014)
13. Schnorr, C.P.: Efficient signature generation for smart cards. J. Cryptology **4**(3), 161–174 (1991). Springer
14. Schnorr, C.-P., Jakobsson, M.: Security of Signed ElGamal encryption. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, p. 73. Springer, Heidelberg (2000)
15. Seurin, Y., Treger, J.: A robust and plaintext-aware variant of Signed ElGamal encryption. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 68–83. Springer, Heidelberg (2013)
16. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptology **15**(2), 75–96 (2002)
17. Tsiounis, Y., Yung, M.: On the security of ElGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, p. 117. Springer, Heidelberg (1998)
18. Wikström, D.: Simplified submission of inputs to protocols. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 293–308. Springer, Heidelberg (2008)

CCA-Secure Keyed-Fully Homomorphic Encryption

Junzuo Lai[1,2], Robert H. Deng[3], Changshe Ma[4], Kouichi Sakurai[5], and Jian Weng[1(✉)]

[1] Department of Computer Science, Jinan University, Guangzhou, China
{laijunzuo,cryptjweng}@gmail.com
[2] The State Key Laboratory of Integrated Services Networks, Xidian University, Xidian, China
[3] School of Information Systems, Singapore Management University, Singapore, Singapore
robertdeng@smu.edu.sg
[4] School of Computer, South China Normal University, Guangzhou, China
chsma@163.com
[5] Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka, Japan
sakurai@inf.kyushu-u.ac.jp

Abstract. To simultaneously achieve CCA security and homomorphic property for encryption, Emura et al. introduced a new cryptographic primitive named keyed-homomorphic encryption, in which homomorphic ciphertext manipulations can only be performed by someone holding a devoted evaluation key which, by itself, does not enable decryption. A keyed-homomorphic encryption scheme should provide CCA2 security when the evaluation key is unavailable to the adversary and remain CCA1-secure when the evaluation key is exposed. While existing keyed-homomorphic encryption schemes only allow simple computations on encrypted data, our goal is to construct CCA-secure *keyed-fully homomorphic encryption* (keyed-FHE) capable of evaluating any functions on encrypted data with an evaluation key.

In this paper, we first introduce a new primitive called convertible identity-based fully homomorphic encryption (IBFHE), which is an IBFHE with an additional transformation functionality, and define its security notions. Then, we present a generic construction of CCA-secure keyed-FHE from IND-sID-CPA-secure convertible IBFHE and strongly EUF-CMA-secure signature. Finally, we propose a concrete construction of IND-sID-CPA-secure convertible IBFHE, resulting in the first CCA-secure keyed-FHE scheme in the standard model.

Keywords: Chosen ciphertext security · Fully homomorphic encryption · Convertible identity-based fully homomorphic encryption

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 70–98, 2016.
DOI: 10.1007/978-3-662-49384-7_4

1 Introduction

Today's information services are increasingly storing data across many servers shared with other data owners. An example of this is cloud computing which has the great potential of providing various services to the society at significantly reduced cost due to aggregated management of elastic resources. Since software systems are not guaranteed to be bug-free and hardware platforms are not under direct control of data owners in such distributed systems, security risks are abundant. To mitigate users' privacy concern about their data, a common solution is to outsource data in encrypted form so that it will remain private even if data servers are not trusted or compromised. To not nullify the benefits of cloud computing, however, we need homomorphic encryption schemes that allow meaningful computations on encrypted data. Recently, in a breakthrough effort, Gentry [28] constructed a *fully homomorphic encryption* (FHE) scheme enabling anyone to compute *arbitrary* functions on encrypted data. On the other hand, security against chosen-ciphertext attack (CCA) [24,42,47] is now a commonly accepted standard security notion for encryption, and unfortunately, it is well-known that CCA security and the homomorphic property cannot be achieved simultaneously.

The incompatibility of CCA security and homomorphicity cannot be reconciled under the assumption that everyone can "freely" perform homomorphic operations on ciphertexts. Very recently, Emura et al. [25] showed that in the setting where homomorphic operations are performed in a "controlled" fashion, CCA security and homomorphicity can be simultaneously achieved. They suggested a new primitive called keyed-homomorphic encryption [25], where homomorphic ciphertext manipulations are only possible to a party holding a devoted evaluation key EK which, by itself, does not enable decryption. A keyed-homomorphic encryption scheme should provide CCA2 security when the evaluation key is unavailable to the adversary and remain CCA1 secure when EK is exposed. Emura et al. [25] presented a number of keyed-homomorphic encryption schemes through hash proof systems [22], which only allow simple computations on encrypted data (i.e., either adding or multiplying encrypted ciphertexts, *but not both operations at the same time*). This paper is motivated by the goal of constructing CCA-secure *keyed-fully homomorphic encryption* (keyed-FHE)[1] capable of evaluating *any* functions on encrypted data with a devoted evaluation key EK.

Our Contribution. One may hope to obtain CCA-secure keyed-FHE by using the double encryption methodology: a ciphertext of an "inner" CPA-secure FHE scheme is encrypted by an "outer" CCA-secure encryption scheme, and the evaluation key EK is the decryption key of the "outer" CCA-secure encryption scheme. Unfortunately, this naive construction is not secure in the sense of our security definition for keyed-fully homomorphic encryption. An adversary is allowed to

[1] We focus on *leveled* keyed-FHE schemes, and typically omit the term "leveled". In a *leveled* keyed-FHE scheme, the parameters of the scheme may depend on the *depth*, but not the *size*, of the circuits that the scheme can evaluate.

issue decryption queries before the evaluation key EK is exposed to the adversary in our security definition. However, no such decryption query is allowed in the CPA security game of the underlying "inner" FHE scheme[2].

We propose a generic paradigm of constructing CCA-secure keyed-FHE, which follows the line of CHK transformation [18]. It is worth noting that, one cannot achieve CCA-secure keyed-FHE from $\mathsf{IND\text{-}sID\text{-}CPA}$-secure IBFHE by CHK transformation directly, since each IBE ciphertext is under a *fresh* identity and the homomorphic evaluation functionality of IBFHE does not work.

- We define a new primitive named convertible identity-based fully homomorphic encryption (IBFHE) and its $\mathsf{IND\text{-}sID\text{-}CPA}$ security notions. Informally, a convertible IBFHE is an IBFHE with an additional transformation functionality, which may be of independent interest.
- Based on our new primitive, $\mathsf{IND\text{-}sID\text{-}CPA}$-secure convertible IBFHE, and strongly $\mathsf{EUF\text{-}CMA}$-secure signature, we propose a generic paradigm of constructing CCA-secure keyed-FHE by modifying CHK transformation [18] slightly.
- We construct a convertible identity-based (leveled) FHE scheme based on the adaptively-secure IBE scheme proposed by Agrawal et al. [1], and prove that it is $\mathsf{IND\text{-}sID\text{-}CPA}$ secure in the standard model, resulting in the first CCA-secure keyed-FHE scheme in the standard model. Actually, one can use our techniques to construct convertible IBFHE schemes based on the adaptively-secure IBE schemes proposed in [2,19].

Convertible IBFHE. A convertible IBFHE scheme consists of seven algorithms: Setup, $\mathsf{Extract}$, $\mathsf{GenerateTK}$, $\mathsf{Encrypt}$, $\mathsf{Transform}$, $\mathsf{Decrypt}$ and $\mathsf{Evaluate}$. Among these algorithms, (Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, $\mathsf{Evaluate}$) constitute the traditional IBFHE scheme; algorithms $\mathsf{GenerateTK}$ and $\mathsf{Transform}$ provide the following functionality: given a transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}$ for an identity $\widetilde{\mathsf{ID}}$, which is generated by an authority using algorithm $\mathsf{GenerateTK}$, one with the help of algorithm $\mathsf{Transform}$ can transform a ciphertext CT under *any* identity into a ciphertext under identity $\widetilde{\mathsf{ID}}$ without changing the underlying plaintext of CT.

The additional functionality of convertible IBFHE is reminiscent of identity-based proxy re-encryption (IBPRE) [32]. Unlike convertible IBFHE, in an IBPRE scheme, a transformation key (i.e., re-encryption key) $\mathsf{TK}_{\mathsf{ID}_1\to\mathsf{ID}_2}$ associated with two identities ID_1 and ID_2, is generated by the user with identity

[2] Another naive approach to construct CCA-secure keyed-FHE is to utilize Naor-Yung paradigm [42]: a plaintext is encrypted twice (independently) by CPA-secure FHE, and then a non-malleable non-interactive zero-knowledge (NIZK) [51] proof is used in order to prove that both ciphertexts are encryptions to the same plaintext (the CRS needed for the NIZK is part of the public key); the evaluation key EK is the trapdoor associated with the CRS. However, as the construct by using the double encryption methodology, this construction is not secure in the sense of our security definition: the adversary is allowed to use the decryption oracle even after the challenge phase, just before the adversary requests EK.

ID_1, and one with the transformation key can *only* convert an encryption under identity ID_1 into the encryption under identity ID_2.

The adaptive security of convertible IBFHE requires that given a challenge ciphertext CT^* under some identity ID^*, no PPT adversary can distinguish, except with a negligible advantage, whether CT^* is an encryption of 1 under identity ID^* or an encryption of 0 under identity ID^*. We allow an adversary to adaptively issue private key queries on identities ID and transformation key queries on identities $\widetilde{\mathsf{ID}}$, but with the natural constraints that: (1) the adversary cannot issue private key query for the challenge identity ID^*; (2) the adversary cannot issue private key query for an identity $\widetilde{\mathsf{ID}}$ such that the adversary has issued a transformation key query on $\widetilde{\mathsf{ID}}$, and *vice versa*.

For constructing CCA-secure keyed-FHE, we only require that the underlying convertible IBFHE be secure in a weaker security model, denoted as IND-sID-CPA security model. In this weaker security model, the transformation key query can be issued only *once* by the adversary, and the target identity ID^* and the designated identity $\widetilde{\mathsf{ID}}$ which the adversary wants to obtain the corresponding transformation key must be committed by the adversary ahead of the system setup.

CCA-SECURE KEYED-FHE FROM IND-SID-CPA-SECURE CONVERTIBLE IBFHE. We give a high-level description on how to construct a CCA-secure keyed-FHE scheme from an IND-sID-CPA-secure convertible IBFHE scheme characterized by $(\mathsf{GenerateTK}, \mathsf{Transform})$, with the help of a strongly EUF-CMA-secure signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

The public key of our proposed keyed-FHE scheme is the public parameters of the convertible IBFHE scheme, the secret key is the corresponding master key, and the evaluation key is $(\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto \widetilde{vk}})$, where $(\widetilde{vk}, \widetilde{sk})$ is a key-pair for the signature scheme $\mathcal{S}$ and $\mathsf{TK}_{\mapsto \widetilde{vk}}$ which is generated by algorithm $\mathsf{GenerateTK}$ of the convertible IBFHE scheme is the transformation key for "identity" $\widetilde{vk}$.

To encrypt a message bit, the encryption algorithm first runs algorithm $\mathcal{S}.\mathsf{Gen}$ to obtain a key-pair (vk, sk), and then uses the convertible IBFHE scheme to encrypt the message bit with respect to the "identity" vk, with the resulting ciphertext denoted as CT. Next, the signing key sk is used to sign CT to obtain a signature σ. The final ciphertext C consists of the verification key vk, the convertible IBFHE ciphertext CT and the signature σ. Given a ciphertext $C = (vk, \mathsf{CT}, \sigma)$, the decryption algorithm first uses algorithm $\mathcal{S}.\mathsf{Vrfy}$ to verify the signature σ on CT with respect to vk and outputs $\perp$ if the verification fails. Otherwise, the decryption algorithm generates the private key SK_{vk} corresponding to the "identity" vk, and decrypts the ciphertext CT using the underlying convertible IBFHE scheme.

Given a tuple of ciphertexts $\boldsymbol{C} = (C_1, \ldots, C_k)$ where $C_i = (vk_i, \mathsf{CT}_i, \sigma_i)$, and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$, the evaluation algorithm first verifies the signature σ_i on CT_i with respect to vk_i for each $i \in [k]$ and outputs $\perp$ if the verification fails. Otherwise, for each $i \in [k]$, with $\mathsf{TK}_{\mapsto \widetilde{vk}}$, it runs algorithm $\mathsf{Transform}$ of the convertible IBFHE scheme to convert the ciphertext CT_i

under "identity" vk_i into a ciphertext $\widetilde{\mathsf{CT}}_i$ under the "identity" $\widetilde{vk}$. Since now $\widetilde{\mathsf{CT}}_1, \ldots, \widetilde{\mathsf{CT}}_k$ are the ciphertexts under the same "identity" $\widetilde{vk}$, the evaluation algorithm can evaluate the Boolean circuit f on the ciphertexts $\widetilde{\mathsf{CT}}_1, \ldots, \widetilde{\mathsf{CT}}_k$ using the underlying convertible IBFHE scheme. Then the resulting ciphertext $\widetilde{\mathsf{CT}}$ is signed using $\widetilde{sk}$ to obtain a signature $\tilde{\sigma}$, and the evaluation algorithm outputs the ciphertext $C = (\widetilde{vk}, \widetilde{\mathsf{CT}}, \tilde{\sigma})$.

As for the security of our proposed keyed-FHE scheme, we show that if there exists an adversary $\mathcal{A}$ with a non-negligible advantage in the CCA security game, we can create a reduction algorithm $\mathcal{B}$ that breaks the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying convertible IBFHE scheme. The reduction algorithm $\mathcal{B}$ is informally described as follows. $\mathcal{B}$ first runs $\mathcal{S}.\mathsf{Gen}$ to obtain two key-pairs (vk^*, sk^*) and $(\widetilde{vk}, \widetilde{sk})$. Then, $\mathcal{B}$ sets vk^* and $\widetilde{vk}$ as its target "identity" and designated "identity", which are submitted to its challenger in the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game of the convertible IBFHE scheme. $\mathcal{B}$ is given the public parameters of the convertible IBFHE scheme and the transformation key $\mathsf{TK}_{\mapsto \widetilde{vk}}$ for "identity" $\widetilde{vk}$. Now, $\mathcal{B}$ can use $(\widetilde{vk}, \widetilde{sk})$ and $\mathsf{TK}_{\mapsto \widetilde{vk}}$ to answer $\mathcal{A}$'s evaluation queries and the evaluation key query, and the challenge ciphertext C^* given to $\mathcal{A}$ is set as $(vk^*, \mathsf{CT}^*, \sigma^*)$, where CT^* is $\mathcal{B}$'s challenge ciphertext of the convertible IBFHE scheme and $\sigma^* \leftarrow \mathcal{S}.\mathsf{Sign}(sk^*, \mathsf{CT}^*)$. Next, we shall explain how $\mathcal{B}$ answers the decryption queries for ciphertexts $C = (vk, \mathsf{CT}, \sigma)$ issued by adversary $\mathcal{A}$.

We say a ciphertext $C = (vk, \mathsf{CT}, \sigma)$ is valid if σ is a valid signature on CT with respect to vk. For $\mathcal{A}$'s decryption query on a ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that C is a valid ciphertext and $vk \notin \{vk^*, \widetilde{vk}\}$, $\mathcal{B}$ can issue a private key query on the "identity" vk to its challenger to obtain the corresponding private key SK_{vk}, and use the private key SK_{vk} to answer $\mathcal{A}$'s query. The subtlety lies in how $\mathcal{B}$ deals with $\mathcal{A}$'s decryption query on a valid ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk \in \{vk^*, \widetilde{vk}\}$. Recall that $\mathcal{B}$ is not allowed to issue a private key query on the "identity" $vk \in \{vk^*, \widetilde{vk}\}$ to it's own challenger in the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game of the convertible IBFHE scheme. We first note that any valid ciphertext $C = (vk, \mathsf{CT}, \sigma)$ submitted by the adversary during its queries must, except with negligible probability, have $vk \neq vk^*$ by the strong security of the signature scheme $\mathcal{S}$. The crux of the security proof is then to show how $\mathcal{B}$ answers $\mathcal{A}$'s decryption query on a valid ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk = \widetilde{vk}$.

In our security definition of keyed-fully homomorphic encryption, the adversary can issue the decryption and evaluation queries only if it does not request the evaluation key to be exposed. Hence, for any valid ciphertext $C = (vk, \mathsf{CT}, \sigma)$ submitted by the adversary during its decryption queries, if $vk = \widetilde{vk}$, with overwhelming probability, C is one of $\mathcal{B}$'s responses to $\mathcal{A}$'s evaluation queries by the strong $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathcal{S}$. Based on the above observation, $\mathcal{B}$ will resort to a list EList to answer $\mathcal{A}$'s decryption query on a valid ciphertext $C = (vk = \widetilde{vk}, \mathsf{CT}, \sigma)$. The list EList is set as $\emptyset$ initially and is updated while answering $\mathcal{A}$'s evaluations queries. Now, when $\mathcal{A}$ issues an evaluation query on a tuple of ciphertext $\boldsymbol{C} = (C_1, \ldots, C_k)$ and a Boolean circuit

f, after sending the result C of the evaluation algorithm to the adversary, $\mathcal{B}$ additionally proceeds as follows.

1. Check whether there exists an $i \in [k]$ such that $C_i = C^*$. If so, update the list by $\mathsf{EList} \leftarrow \mathsf{Elist} \cup \{(\perp, C)\}$. Note that, to avoid an unachievable security definition, $\mathcal{B}$ answers $\perp$ for "unallowable ciphertext" that are the result of homomorphic evaluation for C^* and any ciphertext of $\mathcal{A}$'s choice.
2. For each valid ciphertext $C_i = (vk_i, \mathsf{CT}_i, \sigma_i)$ where $i \in [k]$, obtain the corresponding plaintext b_i by finding the corresponding record (b_i, C_i) in the list EList if $vk_i = \widetilde{vk}$ or decrypting CT_i with the help of issuing a private key query on the "identity" vk_i to its challenger. Then, compute the message bit $m = f(b_1, \ldots, b_k)$ and update the list by $\mathsf{EList} \leftarrow \mathsf{Elist} \cup \{(m, C)\}$.

Consequently, when $\mathcal{A}$ issues a decryption query on a valid ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk = \widetilde{vk}$, except with negligible probability, $\mathcal{B}$ can find a record (m, C) in the list EList and return m to the adversary as its answer. Hence, by the strong $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathcal{S}$, with overwhelming probability, $\mathcal{B}$ simulates the CCA security game of our proposed keyed-FHE scheme for $\mathcal{A}$ properly. Therefore, if $\mathcal{A}$ has a non-negligible advantage in the CCA security game, $\mathcal{B}$ breaks the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying convertible IBFHE scheme with a non-negligible advantage.

Construction of IND-sID-CPA-secure convertible IBFHE. Based on the standard learning with errors (LWE) problem [49], Agrawal et al. [1] proposed an efficient identity-based encryption scheme and showed that their base construction can be extended to an adaptively-secure IBE using a lattice analog of the Waters IBE [56]. Our $\mathsf{IND\text{-}sID\text{-}CPA}$-secure convertible IBFHE starts from the adaptively-secure IBE scheme in [1].

An encryption of a message bit b for an identity $\mathsf{ID} = (d_1, \ldots, d_\ell) \in \{-1, 1\}^\ell$ in the adaptively-secure IBE scheme [1] takes the form of

$$c_0 = u^\top s + x + b\lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q,\ c_1 = F_{\mathsf{ID}}^\top s + \begin{bmatrix} y \\ R_{\mathsf{ID}}^\top y \end{bmatrix} \in \mathbb{Z}_q^{2m},$$

where $F_{\mathsf{ID}} = A \mid B_0 + \sum_{i=1}^{\ell} d_i B_i$, $R_{\mathsf{ID}} = \sum_{i=1}^{\ell} d_i R_i$, and $A, B_0, B_1, \ldots, B_\ell, u$ are the system's public parameters, a short basis T_A for $\Lambda_q^\perp(A)$ is the master key, $s, x, y, R_1, \ldots, R_\ell$ are noise vectors with short norm used in the encryption algorithm. The private key $\mathsf{SK}_{\mathsf{ID}}$ for identity ID is a short vector e_{ID} in $\Lambda_q^u(F_{\mathsf{ID}})$, hence the message bit b can be recovered from $c_0 - e_{\mathsf{ID}}^\top c_1$.

Agrawal et al. [1] utilized the partitioning strategy to prove the adaptively-secure security of the above IBE scheme. In the security reduction, $B_1, \ldots, B_\ell$ in the public parameters are set as $B_i = AR_i^* + h_i B_0$, where all the matrices R_i^* are random and h_i is a secret coefficient in $\mathbb{Z}_q$. Consequently,

$$F_{\mathsf{ID}} = A \mid B_0 + \sum_{i=1}^{\ell} d_i B_i = A \mid A(\sum_{i=1}^{\ell} d_i R_i^*) + h_{\mathsf{ID}} B_0,$$

where $h_{\mathsf{ID}} = (1 + \sum_{i=1}^{\ell} d_i h_i)$, and the identity space is partitioned into two parts according to whether h_{ID} is equal to 0 or not. If $h_{\mathsf{ID}} \neq 0$, the simulator, without knowing the master key, can use a trapdoor T_{B_0} for $\Lambda_q^{\perp}(B_0)$ to generate the private key for identity ID, i.e., a short vector e_{ID} in $\Lambda_q^u(F_{\mathsf{ID}})$. The simulator cannot produce the corresponding private key for identities ID such that $h_{\mathsf{ID}} = 0$, but will be able to construct a useful challenge to solve the given LWE problem instance. Let ID^* be the challenge identity and let $\mathsf{ID}_1, \ldots, \mathsf{ID}_Q$ be the identities for which the adversary issues private key queries. The security proof will require that for any $\mathsf{ID}^*, \mathsf{ID}_1, \ldots, \mathsf{ID}_Q$, with non-negligible probability,

$$h_{\mathsf{ID}^*} = 0 \wedge h_{\mathsf{ID}_1} \neq 0 \wedge \ldots \wedge h_{\mathsf{ID}_\ell} \neq 0,$$

which can be satisfied by the abort-resistant hash family used in [7,34,56].

The idea of constructing convertible IBFHE is summarized as follows. We first show how to design a convertible IBE scheme (i.e. without the homomorphic evaluation functionality), and then extend it to a convertible IBFHE scheme. To construct convertible IBE, we should provide an approach to converting a ciphertext CT under *any* identity ID into a ciphertext $\widetilde{\mathsf{CT}}$ under the designated identity $\widetilde{\mathsf{ID}}$. For transformation correctness (i.e., decrypting CT and $\widetilde{\mathsf{CT}}$ with the corresponding private key $\mathsf{SK}_{\mathsf{ID}}$ for identity ID and $\mathsf{SK}_{\widetilde{\mathsf{ID}}}$ for identity $\widetilde{\mathsf{ID}}$ respectively, must have the same result), we need be able to check whether a ciphertext is *well-formed*. However, starting from the adaptively-secure IBE scheme proposed in [1], we are thrown into a dilemma. Agrawal et al. [1] proved that in their proposed IBE scheme, encryption of any message bit is indistinguishable from uniform vector over $\mathbb{Z}_q$ under the LWE assumption. That is, any well-formed ciphertext in [1] is pseudorandom; thus it is difficult to design a mechanism to check the well-formedness of a ciphertext. We resort to the recent advances in indistinguishability obfuscation [52] to overcome the obstacle.

Besides $A, B_0, B_1, \ldots, B_\ell, u$, the public parameters of our proposed convertible IBE include an indistinguishability obfuscation of the following program that takes as input an identity $\mathsf{ID} = (d_1, \ldots, d_\ell) \in \{-1, 1\}^\ell$, a message bit $b \in \{0, 1\}$ and randomness r,

1. Set $t = \mathsf{PRG}(r)$ and $(s, x, y, R_1, \ldots, R_\ell) = \mathsf{F}(\mathrm{K}, \mathsf{ID}, t)$;
2. Compute $c_0 = u^\top s + x + b\lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$, $c_1 = F_{\mathsf{ID}}^\top s + \begin{bmatrix} y \\ R_{\mathsf{ID}}^\top y \end{bmatrix} \in \mathbb{Z}_q^{2m}$ and output (t, c_0, c_1).

The system's master key additionally includes the key K to the puncturable pseudorandom function (PRF) [52] F. Let $\mathbf{P}^{\mathsf{Enc}}$ be the above obfuscated program. The encryption algorithm simply runs $\mathbf{P}^{\mathsf{Enc}}(\mathsf{ID}, b, r)$ and outputs the result (t, c_0, c_1). The private key $\mathsf{SK}_{\mathsf{ID}}$ for identity ID is a short vector e_{ID} in $\Lambda_q^u(F_{\mathsf{ID}})$, and given a ciphertext (t, c_0, c_1) under identity ID, the message bit b can be recovered from $c_0 - e_{\mathsf{ID}}^\top c_1$. Observe that, with the knowledge of K, one can retrieve the randomness $s, x, y, R_1, \ldots, R_\ell$ and the message bit b from a ciphertext (t, c_0, c_1) under an identity ID, and thus can check the well-formedness of the ciphertext and re-encrypt b under another identity. Consequently, the transformation key

for an designated identity $\widetilde{\mathsf{ID}} = (\tilde{d}_1, \ldots, \tilde{d}_\ell) \in \{-1,1\}^\ell$ is an indistinguishability obfuscation of the following program that takes as input a ciphertext $\mathsf{CT} = (t, c_0, c_1)$ under an identity $\mathsf{ID} = (d_1, \ldots, d_\ell) \in \{-1,1\}^\ell$,

1. Set $(s, x, y, R_1, \ldots, R_\ell) = \mathsf{F}(\mathrm{K}, \mathsf{ID}, t)$, and check whether there exists $b \in \{0,1\}$ such that $c_0 = u^\top s + x + b\lfloor \frac{q}{2} \rfloor$ and $c_1 = F_{\mathsf{ID}}^\top s + \begin{bmatrix} y \\ R_{\mathsf{ID}}^\top y \end{bmatrix}$. If not, output $\perp$.
2. Set $(\tilde{s}, \tilde{x}, \tilde{y}, \tilde{R}_1, \ldots, \tilde{R}_\ell) = \mathsf{F}(\mathrm{K}, \widetilde{\mathsf{ID}}, t)$, and compute $\tilde{c}_0 = u^\top \tilde{s} + \tilde{x} + b\lfloor \frac{q}{2} \rfloor$, $\tilde{c}_1 = F_{\widetilde{\mathsf{ID}}}^\top \tilde{s} + \begin{bmatrix} \tilde{y} \\ \tilde{R}_{\widetilde{\mathsf{ID}}}^\top \tilde{y} \end{bmatrix}$ and output $(t, \tilde{c}_0, \tilde{c}_1)$.

Let $\mathbf{P}^{\mathsf{Trans}}$ be the above obfuscated program. To convert an encryption CT under identity ID into the encryption under identity $\widetilde{\mathsf{ID}}$, the transformation algorithm now simply runs $\mathbf{P}^{\mathsf{Trans}}(\mathsf{ID}, \mathsf{CT})$ and outputs the result.

As for the IND-sID-CPA security of the convertible IBE scheme, we follow the line of [1], i.e., utilizing the partitioning strategy. Let ID^* be the challenge identity, $\widetilde{\mathsf{ID}}$ be the designated identity which the adversary wants to obtain the corresponding transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}} = (\widetilde{\mathsf{ID}}, \mathbf{P}^{\mathsf{Trans}})$, and $\mathsf{ID}_1, \ldots, \mathsf{ID}_Q$ be the identities for which the adversary issues private key queries. Let $\mathsf{CT}^* = (t^*, c_0^*, c_1^*)$ be the challenge ciphertext for ID^*. In the security reduction, there exist some subtleties:

1. It requires that $h_{\mathsf{ID}^*} = 0$, in order to construct the challenge $\mathsf{CT}^* = (t^*, c_0^*, c_1^*)$ to solve the given LWE problem instance. Like the security reduction in [1], the randomness s^*, x^*, y^* that are used to evaluate c_0^* and c_1^*, come from the given LWE problem instance and is unknown to the simulator. Hence, when the adversary runs $\mathbf{P}^{\mathsf{Trans}}(\mathsf{ID}^*, \mathsf{CT}^*)$, it will get an error symbol $\perp$, which enables it to distinguish the simulated settings and the real settings. We observe that the simulator can prepare CT^* and $\widetilde{\mathsf{CT}}^*$ at the setup phase, where $\widetilde{\mathsf{CT}}^*$ denotes the corresponding result of calling the transformation algorithm on the challenge ciphertext $\mathsf{CT}^* = (t^*, c_0^*, c_1^*)$, since in the IND-sID-CPA security game the adversary must commit ID^* and $\widetilde{\mathsf{ID}}$ ahead of the system setup. Consequently, the simulator can employ the technique of *punctured programs*, introduced by Sahai et al. [52], to simulate the transformation key for $\widetilde{\mathsf{ID}}$ properly.
2. It requires that for *any* $\mathsf{ID}^*, \widetilde{\mathsf{ID}}, \mathsf{ID}_1, \ldots, \mathsf{ID}_Q$, with non-negligible probability, $h_{\mathsf{ID}^*} = 0 \wedge h_{\widetilde{\mathsf{ID}}} = 0 \wedge h_{\mathsf{ID}_1} \neq 0 \wedge \ldots \wedge h_{\mathsf{ID}_\ell} \neq 0$. Unfortunately, it cannot be satisfied by the abort-resistant hash family used in [7,34,56]. On the other hand, we observe that, if ID^* and $\widetilde{\mathsf{ID}}$ are chosen uniformly at random, with non-negligible probability, the requirement of $h_{\mathsf{ID}^*} = 0 \wedge h_{\widetilde{\mathsf{ID}}} = 0 \wedge h_{\mathsf{ID}_1} \neq 0 \wedge \ldots \wedge h_{\mathsf{ID}_\ell} \neq 0$ can be satisfied by the abort-resistant hash family used in [7, 34,56]. Therefore, we use another puncturable PRF to map an identity ID into a random identity $\mathsf{id} \in \{-1,1\}^\ell$, and replace ID with id in all functionalities. Similarly, since ID^* and $\widetilde{\mathsf{ID}}$ must be committed by the adversary ahead of the system setup, the technique of *punctured programs* allows the simulator's simulation be performed properly.

So far, we obtain an IND-sID-CPA-secure convertible IBE scheme. Next, we show that the convertible IBE scheme extends to a convertible IBFHE scheme. Recently, Gentry et al. [31] described a simple "compiler" that transforms any LWE-based IBE scheme (that satisfies certain natural properties) into an identity-based (leveled) FHE scheme. Since our proposed convertible IBE scheme starts from the LWE-based IBE schemes proposed in [1] that have the required properties, we can utilize the "compiler" to transform it into a convertible identity-based (leveled) FHE scheme.

Related Work. Emura et al. [25] showed that CCA security does not rule out homomorphicity when the capability to compute on encrypted data is controlled, by introducing a primitive called keyed-homomorphic encryption. Other approaches to reconcile homomorphism and non-malleability were taken in [9,20,44–46] but they inevitably satisfy weaker security notions than CCA security.

Based on hash proof systems [22], Emura et al. [25] constructed a number of CCA-secure keyed-homomorphic schemes. Recently, Libert et al. [40] applied linearly homomorphic structure-preserving signatures [39] to quasi-adaptive non-interactive zero-knowledge (QA-NIZK) proofs [37], proposed QA-NIZK proofs with unbounded simulation-soundness (USS), and constructed a CCA-secure keyed-homomorphic scheme with threshold decryption by applying USS. These CCA-secure keyed-homomorphic schemes only allow simple computations on encrypted data, i.e., either adding or multiplying encrypted ciphertexts, *but not both operations at the same time.*

Fully Homomorphic Encryption. The notion of fully homomorphic encryption (FHE) capable of performing any computations on encrypted data, was first put forward by Rivest et al. [50]. However, only in the past few years have candidate FHE schemes been proposed. The first such scheme was constructed by Gentry [28]; his work inspired a tremendous amount of research effort on improving the efficiency of his scheme [13,21,29,30,53,54], realizations of FHE based on different assumptions [14,16,17,55], and so on. Until now, fully homomorphic encryption schemes can only be proven secure against chosen-plaintext attack (CPA).

Controlled Homomorphic Encryption. Desmedt et al. [23] put forth the notion of a controllable homomorphic encryption scheme (CHES) that blends together the notion of a fully homomorphic encryption scheme and of a functional encryption scheme [8]. In a CHES, a *designated* homomorphic operation C can be efficiently performed on a *single* ciphertext by a party that has a special token for function C that is released by the owner of the secret key. Compared with CHES, keyed-FHE enables a party holding a devoted evaluation key to compute *arbitrary* functions on *ciphertexts*, and it can provide CCA2 security when the evaluation key is unavailable.

Indistinguishability Obfuscation. Program obfuscation deals with the problem of how to protect a program from reverse engineering while preserving functionality. Unfortunately, Barak et al. [5,6] showed that the most natural simulation-based

formulation of program obfuscation (a.k.a. "black-box obfuscation") is impossible to achieve for *general* programs in a very strong sense. Faced with this impossibility result, Barak et al. [5,6] suggested another notion of program obfuscation named *indistinguishability obfuscation*. Roughly speaking, an indistinguishability obfuscation scheme ensures that the obfuscations of any two functionally equivalent circuits are computationally indistinguishable. Recently, Garg et al. [27] proposed the first candidate construction of an efficient indistinguishability obfuscation ($i\mathcal{O}$) for *general* programs.

Recently, staring with [52] there has been much interest in investigating what can be built from $i\mathcal{O}$, since this model leads to poly-time obfuscation of unrestricted program classes, circumventing the known impossibility results of [5,6]. Subsequently, many papers [11,26,33,35,36,48,52,57] have shown a wide range of cryptographic applications of $i\mathcal{O}$. We utilize $i\mathcal{O}$ to construct an IND-sID-CPA-secure convertible IBFHE scheme.

Organization. The rest of the paper is organized as follows. Some preliminaries are given in Sect. 2. We introduce the notion and security model of convertible IBFHE in Sect. 3. We propose a paradigm of constructing CCA-secure keyed-FHE from IND-sID-CPA-secure convertible IBFHE and strongly EUF-CMA-secure signature in Sect. 4. We present a concrete construction of IND-sID-CPA-secure convertible identity-based (leveled) FHE in Sect. 5. Section 6 concludes the paper.

2 Preliminaries

If S is a set, then $s_1, \ldots, s_t \leftarrow S$ denotes the operation of picking elements $s_1, \ldots, s_t$ uniformly at random from S. If $n \in \mathbb{N}$ then $[n]$ denotes the set $\{1, \ldots, n\}$. For a probabilistic algorithm A, we denote $y \leftarrow A(x; R)$ the process of running A on input x and with randomness R, and assigning y the result. Let $\mathcal{R}_A$ denote the randomness space of A, and we write $y \leftarrow A(x)$ for $y \leftarrow A(x; R)$ with R chosen from $\mathcal{R}_A$ uniformly at random. A function $f(\kappa)$ is *negligible*, if for every $c > 0$ there exists a κ_c such that $f(\kappa) < 1/\kappa^c$ for all $\kappa > \kappa_c$. For a real $x \in \mathbb{R}$, $\lfloor x \rceil$ denotes the nearest integer to x, and $\lfloor x \rfloor$, $\lceil x \rceil$ for $x \geq 0$ to indicate rounding down or up.

2.1 Lattices

A full-rank lattice Λ is the set of all integer linear combinations of n linearly independent basis vectors belonging to some $\mathbb{R}^n$. In this work, we are interested in full-rank integer lattices that are restricted to $\mathbb{Z}^n$.

Definition 1. *Fixing q and given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, define the following m-dimensional Ajtai lattices,*

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} = \boldsymbol{A}^T \boldsymbol{s} \bmod q \text{ for some } \boldsymbol{s} \in \mathbb{Z}^n\},$$
$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{y} = \boldsymbol{0} \bmod q\}.$$

For any $\boldsymbol{u} \in \mathbb{Z}_q^n$ admitting an integral solution to $\boldsymbol{Ax} = \boldsymbol{u} \bmod q$, define the coset (or shifted lattice) $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{Ay} = \boldsymbol{u} \bmod q\} = \Lambda_q^{\perp}(\boldsymbol{A}) + \boldsymbol{x}$.

For a set of vectors $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\} \in \mathbb{Z}_q^{n \times m}$, denote by $\|\boldsymbol{B}\|$ the L_2 length of the longest vector in $\boldsymbol{B}$ and denote by $\widetilde{\boldsymbol{B}} = \{\widetilde{\boldsymbol{b}}_1, \ldots, \widetilde{\boldsymbol{b}}_m\}$ the Gram-Schmidt orthogonalization of $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ taken in that order. We refer to $\|\widetilde{\boldsymbol{B}}\|$ as the Gram-Schmidt norm of $\boldsymbol{B}$.

2.2 Discrete Gaussians

Let $\sigma \in \mathbb{R}^+$ and $\boldsymbol{c} \in \mathbb{R}^m$, the Gaussian function On $\mathbb{R}^m$ with center $\mathbf{c}$ and parameter σ is defined as $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x} - \mathbf{c}\|^2/\sigma^2)$. For a positive integer $m \in \mathbb{N}$, and a lattice $\Lambda \in \mathbb{Z}^m$, define the infinite discrete sum of Gaussian function over the lattice Λ, $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x}\in\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ is the m-dimensional Gaussian distribution centered at $\mathbf{c}$ and restricted to the lattice Λ, defined as $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$ for all the lattice point $\mathbf{x} \in \Lambda$. For ease of notation, we omit the center $\mathbf{c}$ if $\mathbf{c} = \mathbf{0}$, and then abbreviate $\mathcal{D}_{\Lambda,\sigma,\mathbf{0}}$ as $\mathcal{D}_{\Lambda,\sigma}$.

2.3 Sampling Algorithms

How to generate a random matrix $\boldsymbol{A}$ statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ along with a short basis (i.e., trapdoor) $\boldsymbol{T}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ is an important technique in lattice-based cryptography. It has been widely investigated by [3,4,41]. We use the trapdoor sampling algorithm proposed by Alwen and Peikert [4].

Theorem 1. *Let $n \geq 1$ and q be an odd prime, and let $m \geq 6n\log q$. There is an efficient probabilistic polynomial-time algorithm* $\mathsf{TrapGen}(q, n)$ *that outputs $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that the distribution of $\boldsymbol{A}$ is within $negl(n)$ statistical distance of uniform and $\boldsymbol{T}$ is a basis of $\Lambda_q^{\perp}(\boldsymbol{A})$ satisfying $\|\boldsymbol{T}\| \leq O(n\log q)$ and $\|\widetilde{\boldsymbol{T}}\| \leq O(\sqrt{n\log q})$ with all but negligible probability in n.*

In the construction and the simulation of our convertible IBFHE scheme, we employ the sampling algorithms $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ given in [1], which can be used to sample relatively short vectors.

Theorem 2. *Let $\boldsymbol{A}$ be a rank n matrix in $\mathbb{Z}_q^{n\times m}$ and let $\boldsymbol{T_A}$ be a "short" basis of $\Lambda_q^{\perp}(\boldsymbol{A})$. Let $\boldsymbol{M}_1$ be a matrix in $\mathbb{Z}_q^{n\times m_1}$ and let $\boldsymbol{F}_1 = \boldsymbol{A}|\boldsymbol{M}_1$. Let $\boldsymbol{u}$ be a vector in $\mathbb{Z}_q^n$ and $\sigma > \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$. There is a probabilistic polynomial-time algorithm* $\mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{M}_1, \boldsymbol{T_A}, \boldsymbol{u}, \sigma)$ *that outputs a vector $\boldsymbol{e} \in \mathbb{Z}_q^{m+m_1}$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}_1),\sigma}$. In particular, $\boldsymbol{e} \in \Lambda_q^{\boldsymbol{u}}(\mathbf{F}_1)$.*

Theorem 3. *Let $\boldsymbol{B}$ be a rank n matrix in $\mathbb{Z}_q^{n\times m}$ and let $\boldsymbol{T_B}$ be a "short" basis of $\Lambda_q^{\perp}(\boldsymbol{B})$. Let $\boldsymbol{R}$ be a matrix in $\mathbb{Z}_q^{k\times m}$. Let $\boldsymbol{A}$ be a matrix in $\mathbb{Z}_q^{n\times k}$ and let $\boldsymbol{F}_2 = \boldsymbol{A}|\boldsymbol{AR}+\boldsymbol{B}$. Let $\boldsymbol{u}$ be a vector in $\mathbb{Z}_q^n$ and $\sigma > \|\widetilde{\boldsymbol{T_B}}\| \cdot \|\boldsymbol{R}\| \cdot \omega(\sqrt{\log(m)})$. There is a probabilistic polynomial-time algorithm* $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{T_B}, \boldsymbol{u}, \sigma)$ *that outputs a vector $\boldsymbol{e} \in \mathbb{Z}_q^{m+k}$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}_1),\sigma}$. In particular, $\boldsymbol{e} \in \Lambda_q^{\boldsymbol{u}}(\mathbf{F}_2)$.*

2.4 The LWE Hardness Assumption

Let n be a positive integer dimension, let $q \geq 2$ be a prime, and let χ be a probability distribution over $\mathbb{Z}_q$. For $\boldsymbol{s} \in \mathbb{Z}_q^n$, let $A_{\boldsymbol{s},\chi}$ and $U_\$$ be two distributions defined as follows:

- $A_{\boldsymbol{s},\chi}$: the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a random vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly, choosing an error term $e \in \mathbb{Z}_q$ according to χ, and outputting $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e)$.
- $U_\$$: the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

For uniformly random $\boldsymbol{s} \in \mathbb{Z}_q^n$, an $(\mathbb{Z}_q, n, \chi)$-LWE problem instance consists of access to a challenge oracle $\mathcal{O}$ that outputs samples $(\boldsymbol{a}, b)$ from $\mathbb{Z}_q^n \times \mathbb{Z}_q$ according to, either the probability distribution $A_{\boldsymbol{s},\chi}$, or the uniform distribution $U_\$$. The $(\mathbb{Z}_q, n, \chi)$-LWE problem allows repeated queries to the challenge oracle $\mathcal{O}$. We say that an algorithm $\mathcal{A}$ decides the $\mathrm{LWE}_{\mathbb{Z}_q,n,\chi}$ problem if

$$\mathrm{Adv}_{\mathcal{A}}^{(\mathbb{Z}_q,n,\chi)\text{-LWE}} = |\Pr[\mathcal{A}^{\mathcal{O}_s} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_\$} = 1]|$$

is non-negligible for a random $\boldsymbol{s} \in \mathbb{Z}_q^n$, where $\mathcal{O}_s$ and $\mathcal{O}_\$$ represent that the oracle $\mathcal{O}$ outputs samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$ according to $A_{\boldsymbol{s},\chi}$ and $U_\$$ respectively.

Regev [49] and Perkert [43] showed that for certain noise distributions χ, denoted $\bar{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction. Brakerski et al. [15] provided the first classical hardness reduction of LWE with polynomial modulus.

Definition 2. *Consider a real parameter $\alpha \in (0,1)$ and a prime q. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ denote the group of reals $[0,1)$ with addition modulo 1. Let Ψ_α be the distribution on $\mathbb{T}$ obtained by sampling a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and reducing the result modulo 1. Let $\bar{\Psi}_\alpha$ denote the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rceil$ where the random variable $X \in \mathbb{T}$ has distribution Ψ_α.*

The following lemma about the distribution $\bar{\Psi}_\alpha$ taken from [1] will be needed to show that decryption works correctly.

Lemma 1. *Let $\boldsymbol{e}$ be some vector in $\mathbb{Z}^m$ and let $\boldsymbol{y} \leftarrow_R \bar{\Psi}_\alpha$. Then the quantity $|\boldsymbol{e}^\top \boldsymbol{y}|$ treated as an integer in $[0, q-1]$ satisfies $|\boldsymbol{e}^\top \boldsymbol{y}| \leq \|e\| q\alpha\omega(\sqrt{\log m}) + \|e\|\sqrt{m}/2$ with all but negligible probability in m.*

2.5 Vector Decomposition

Let k be an integer dimension, let $l = \lfloor \log_2 q \rfloor + 1$ and $N = k \cdot l$. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^k$. We show a way of decomposing vectors that preserves the inner product [31]. We often break vectors into their bit representations as defined below:

$\mathsf{BitDecomp}(\boldsymbol{a})$: For $\boldsymbol{a} \in \mathbb{Z}_q^k$, let $a_{i,j}$ be the j-th bit in a_i's binary representation, bits ordered least significant to most significant. Output the N-dimensional vector $(a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{k,0}, \ldots, a_{k,l-1})$. $\mathsf{BitDecomp}^{-1}(\boldsymbol{a}')$: It is

the inverse of $\mathsf{BitDecomp}$. For $\boldsymbol{a}' = (a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{k,0}, \ldots, a_{k,l-1})$, output $(\sum 2^j \cdot a_{1,j}, \ldots, \sum 2^j \cdot a_{k,j})$. Note that, it is well-defined even if $\boldsymbol{a}'$ is not a 0/1 vector.

$\mathsf{Flatten}(\boldsymbol{a}')$: For N-dimensional vector $\boldsymbol{a}'$, output $\mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\boldsymbol{a}'))$, a N-dimensional vector with 0/1 coefficients.

$\mathsf{Powerof2}(\boldsymbol{b})$: For $\boldsymbol{b} = (b_1, \ldots, b_k) \in \mathbb{Z}_q^k$, output the N-dimensional vector $(b_1, 2b_1, \ldots, 2^{l-1}b_1, \ldots, b_k, 2b_k, \ldots, 2^{l-1}b_k)$.

Claim 1. *Let $\boldsymbol{a}, \boldsymbol{b}$ be vectors of some dimension k over $\mathbb{Z}_q$, let $\boldsymbol{a}'$ be any N-dimensional vector. We have*

- $\langle \mathsf{BitDecomp}(\boldsymbol{a}), \mathsf{Powerof2}(\boldsymbol{b}) \rangle = \langle \boldsymbol{a}, \boldsymbol{b} \rangle$.
- $\langle \boldsymbol{a}', \mathsf{Powerof2}(\boldsymbol{b}) \rangle = \langle \mathsf{BitDecomp}^{-1}(\boldsymbol{a}'), \boldsymbol{b} \rangle = \langle \mathsf{Flatten}(\boldsymbol{a}'), \mathsf{Powerof2}(\boldsymbol{b}) \rangle$.

When $\boldsymbol{A}$ is a matrix, let $\mathsf{BitDecomp}(\boldsymbol{A})$, $\mathsf{BitDecomp}^{-1}(\boldsymbol{A})$ or $\mathsf{Flatten}(\boldsymbol{A})$ be the matrix formed by applying the operation to each row of $\boldsymbol{A}$ separately.

2.6 Indistinguishability Obfuscation

Roughly speaking, an indistinguishability obfuscation ($i\mathcal{O}$) scheme ensures that the obfuscations of any two functionally equivalent circuits are computationally indistinguishable. Indistinguishability obfuscation was originally proposed by Barak et al. [5,6] as a potential weakening of virtual-black-box obfuscation. We recall the definition from [27]. A uniform probabilistic polynomial time (PPT) machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if the following conditions are satisfied:

- Correctness. For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, and for all input x, we have that $\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$.
- Security. For any (not necessarily uniform) PPT distinguisher D, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $C_0(x) = C_1(x)$ on all inputs x the following distinguishing advantage is negligible:

$$\mathsf{Adv}^D_{i\mathcal{O},C_0,C_1}(\lambda) := |\Pr[D(i\mathcal{O}(\lambda, C_0)) = 1] - \Pr[D(i\mathcal{O}(\lambda, C_1)) = 1]|.$$

2.7 Puncturable PRFs

A pseudorandom function (PRF) is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that the function $F(K, \cdot)$ is indistinguishable from random when $K \leftarrow \mathcal{K}$. Puncturable PRFs were defined by Sahai and Waters [52] as a simple type of constrained PRFs [10,12,38]. They defined a puncturable PRF as a PRF for which a key can be given out that allows evaluation of the PRF on all inputs, except for a designated polynomial-size set of inputs. Formally, a *puncturable* PRF $F(K, \cdot)$ is equipped with additional PPT algorithms $(\mathsf{Eval}_F, \mathsf{Puncture}_F)$ such that the following properties hold:

– Correctness. For every PPT algorithm which on input a security parameter λ outputs a set $S \subseteq \{0,1\}^n$, for all $x \in \{0,1\}^n \backslash S$, we have that

$$\Pr[\mathsf{Eval}_F(K\{S\}, x) = F(K, x) : K \leftarrow \mathcal{K}, K\{S\} \leftarrow \mathsf{Puncture}_F(K, S)] = 1.$$

– Security. For any PPT algorithm $\mathcal{A}$, the following distinguishing advantage is negligible:

$$|\Pr[\mathcal{A}(S, K\{S\}, F(K, S)) = 1 : S \leftarrow \mathcal{A}(\lambda), K\{S\} \leftarrow \mathsf{Puncture}_F(K, S)] - \Pr[\mathcal{A}(S, K\{S\}, U_{\bar{\ell}\cdot|S|}) = 1 : S \leftarrow \mathcal{A}(\lambda), K\{S\} \leftarrow \mathsf{Puncture}_F(K, S)]|,$$

where $F(K, S)$ denotes the concatenation of $F(K, x_1), \cdots, F(K, x_k)$, $S = \{x_1, \cdots, x_k\}$ is the enumeration of the elements of S in lexicographic order, $\bar{\ell}$ denotes the bit-length of the output $F(K, x)$, and U_ℓ denotes the uniform distribution over ℓ bits.

2.8 Keyed-Fully Homomorphic Encryption

A keyed-fully homomorphic encryption scheme consists of the following four algorithms:

$\mathsf{Setup}(1^\kappa)$ takes as input a security parameter κ. It outputs a public key PK, a decryption key DK and an evaluation key EK.

$\mathsf{Enc}(\mathsf{PK}, b)$ takes as input a public key PK and a message bit $b \in \{0,1\}$. It outputs a ciphertext C.

$\mathsf{Dec}(\mathsf{PK}, \mathsf{DK}, C)$ takes as input a public key PK, a decryption key DK and a ciphertext C. It outputs a message bit b.

$\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$ takes as input a public key PK, an evaluation key EK, a tuple of ciphertexts $\boldsymbol{C} = (C_1, \ldots, C_k)$ and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$. It outputs a ciphertext C.

Correctness. We require that for each $(\mathsf{PK}, \mathsf{DK}, \mathsf{EK})$ output by $\mathsf{Setup}(1^\kappa)$, the following hold:

Encryption correctness: with overwhelming probability, for all message bit $b \in \{0,1\}$, we have $\mathsf{Dec}(\mathsf{PK}, \mathsf{DK}, \mathsf{Enc}(\mathsf{PK}, b)) = b$.

Evaluation correctness: for any k-ciphertexts $(C_1, \ldots, C_k)$ such that $\mathsf{Dec}(\mathsf{PK}, \mathsf{DK}, C_i) = b_i \in \{0,1\}$, and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$, with overwhelming probability, we have

$$\mathsf{Dec}(\mathsf{PK}, \mathsf{DK}, \mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C} = (C_1, \ldots, C_k), f)) = f(b_1, \ldots, b_k).$$

Security. The CCA security of keyed-FHE scheme is defined using the following game between a PPT adversary $\mathcal{A}$ and a challenger. The adversary is only allowed to issue the decryption queries before it requests the evaluation key EK to be exposed in our security definition; thus it is slightly different from the definition given in [25]. That is, in our model, a keyed-FHE scheme should provide CCA security when the evaluation key is unavailable to the adversary and remain CPA-secure when the evaluation key is exposed.

Setup. The challenger runs $\mathsf{Setup}(1^\lambda)$ to obtain a public key PK, a decryption key DK and an evaluation key EK. It sends the public key PK to the adversary $\mathcal{A}$. In addition, the challenger maintains a list DList, which is set as $\emptyset$ initially.

Query phase 1. The adversary $\mathcal{A}$ adaptively issues the following queries:

- **DecCT**$\langle C \rangle$: The challenger uses the decryption key DK to decrypt C with algorithm Dec. The result is sent back to $\mathcal{A}$. *This query is not allowed to issue if $\mathcal{A}$ has queried to* ***RevEK***.
- **EvalOnCT**$\langle \boldsymbol{C} = (C_1, \ldots, C_k), f \rangle$: The challenger runs $\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$ to obtain a ciphertext C, which is returned to $\mathcal{A}$. *This query is not allowed to issue if $\mathcal{A}$ has queried to* ***RevEK***.
- **RevEK**: The challenger sends the evaluation key EK to $\mathcal{A}$.

Challenge. The challenger first selects a message bit $b^* \in \{0, 1\}$ uniformly at random. Then, it computes $C^* \leftarrow \mathsf{Enc}(\mathsf{PK}, b^*)$, and sends the challenge ciphertext C^* to the adversary. Finally, the challenger updates the list by $\mathsf{DList} \leftarrow \mathsf{DList} \cup \{C^*\}$.

Query phase 2. The adversary $\mathcal{A}$ continues to adaptively issue the following queries:

- **DecCT**$\langle C \rangle$: If $C \in \mathsf{DList}$, the challenger returns $\perp$. Otherwise, the challenger uses the decryption key DK to decrypt C with algorithm Dec, and the result is sent back to $\mathcal{A}$. *This query is not allowed to issue if $\mathcal{A}$ has queried to* ***RevEK***.
- **EvalOnCT**$\langle \boldsymbol{C} = (C_1, \ldots, C_k), f \rangle$: The challenger runs $\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$ to obtain a ciphertext C, which is returned to $\mathcal{A}$. In addition, if there exists $i \in [k]$ such that $C_i \in \mathsf{DList}$, then the challenger updates the list by $\mathsf{DList} \leftarrow \mathsf{DList} \cup \{C\}$. *This query is not allowed to issue if $\mathcal{A}$ has queried to* ***RevEK***.
- **RevEK**: The challenger sends the evaluation key EK to $\mathcal{A}$.

Guess. The adversary $\mathcal{A}$ outputs its guess $b \in \{0, 1\}$ for b^* and wins the game if $b = b^*$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[b = b^*] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 3. *A keyed-FHE scheme is CCA-secure if all probabilistic polynomial time adversaries have at most a negligible advantage in the above security game.*

3 Convertible Identity-Based Fully Homomorphic Encryption

Informally, a convertible IBFHE is an IBFHE with an additional transformation functionality: given a transformation key $\mathsf{TK}_{\mapsto \widetilde{\mathsf{ID}}}$ for an identity $\widetilde{\mathsf{ID}}$, which is generated by the authority, one can transform a ciphertext CT under *any* identity into a ciphertext under identity $\widetilde{\mathsf{ID}}$ without changing the underlying plaintext of CT. Concretely, a convertible IBFHE scheme consists of the following seven algorithms:

$\mathsf{Setup}(1^\kappa)$ takes as input a security parameter κ. It generates a public parameters PP and a master key MK.

$\mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID})$ takes as input the public parameters PP, the master key MK and an identity ID. It produces a private key $\mathsf{SK}_{\mathsf{ID}}$ for identity ID.

$\mathsf{GenerateTK}(\mathsf{PP}, \mathsf{MK}, \widetilde{\mathsf{ID}})$ takes as input the public parameters PP, the master key MK and an identity $\widetilde{\mathsf{ID}}$. It generates a transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}$ for identity $\widetilde{\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, b)$ takes as input the public parameters PP, an identity ID and a message bit $b \in \{0,1\}$. It outputs a ciphertext CT.

$\mathsf{Transform}(\mathsf{PP}, \mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}, \mathsf{ID}, \mathsf{CT})$ takes as input the public parameters PP, a transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}$, and a ciphertext CT for an identity ID. It outputs a ciphertext $\widetilde{\mathsf{CT}}$ under identity $\widetilde{\mathsf{ID}}$.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{CT})$ takes as input the public parameters PP, a private key $\mathsf{SK}_{\mathsf{ID}}$ and a ciphertext CT. It outputs a message bit $b \in \{0,1\}$.

$\mathsf{Evaluate}(\mathsf{PP}, \mathsf{ID}, \boldsymbol{CT}, f)$ takes as input the public parameters PP, a tuple of ciphertexts $\boldsymbol{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k)$ under an identity ID and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$. It outputs a ciphertext CT under identity ID.

Correctness. We require that for each $(\mathsf{PP}, \mathsf{MK})$ output by $\mathsf{Setup}(1^\kappa)$, the following hold:

Encryption correctness: with overwhelming probability, for all identity ID and message bit $b \in \{0,1\}$, we have $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID}), \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, b)) = b$.

Transformation correctness: with overwhelming probability, for all identity $\mathsf{ID}, \widetilde{\mathsf{ID}}$ and message bit $b \in \{0,1\}$, let $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, b)$, $\mathsf{SK}_{\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID})$, $\mathsf{SK}_{\widetilde{\mathsf{ID}}} \leftarrow \mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \widetilde{\mathsf{ID}})$, $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}} \leftarrow \mathsf{GenerateTK}(\mathsf{PP}, \mathsf{MK}, \widetilde{\mathsf{ID}})$, and $\widetilde{\mathsf{CT}} \leftarrow \mathsf{Transform}(\mathsf{PP}, \mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}, \mathsf{ID}, \mathsf{CT})$, we have

$$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{CT}) = \mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_{\widetilde{\mathsf{ID}}}, \widetilde{\mathsf{CT}}).$$

Evaluation correctness: for any k-ciphertexts $(\mathsf{CT}_1, \ldots, \mathsf{CT}_k)$ under an identity ID such that $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID}), \mathsf{CT}_i) = b_i \in \{0,1\}$, and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$, with overwhelming probability, we have $\mathsf{Decrypt}(\mathsf{PP}, \mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID}), \mathsf{Evaluate}(\mathsf{PP}, \mathsf{ID}, \boldsymbol{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k), f)) = f(b_1, \ldots, b_k)$.

Security. The $\mathsf{IND\text{-}sID\text{-}CPA}$ security of convertible IBFHE scheme is defined using the following game between a PPT adversary $\mathcal{A}$ and a challenger.

Init. The adversary submits a target identity ID^* and a designated identity $\widetilde{\mathsf{ID}}$.

Setup. The challenger first runs $\mathsf{Setup}(1^\kappa)$ to obtain a public parameters PP and a master key MK. Then, it runs $\mathsf{GenerateTK}(\mathsf{PP}, \mathsf{MK}, \widetilde{\mathsf{ID}})$ to get the transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}$ for identity $\widetilde{\mathsf{ID}}$, and sends the public parameters PP and the transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}$ to the adversary $\mathcal{A}$.

Query phase 1. The adversary $\mathcal{A}$ adaptively issues the following queries:

- **GetSK**$\langle \mathsf{ID} \rangle$: The challenger runs $\mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID})$ to generate the corresponding private key $\mathsf{SK}_{\mathsf{ID}}$, which is returned to $\mathcal{A}$. We require that $\mathsf{ID} \notin \{\mathsf{ID}^*, \widetilde{\mathsf{ID}}\}$.

Challenge. The challenger first selects a message bit $b^* \in \{0,1\}$ uniformly at random. Then, it computes $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}^*, b^*)$, and sends the challenge ciphertext CT^* to the adversary.

Query phase 2. This is same as Query phase 1.

Guess. The adversary $\mathcal{A}$ outputs its guess $b \in \{0,1\}$ for b^* and wins the game if $b = b^*$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[b = b^*] - \frac{1}{2}|$, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 4. *A convertible IBFHE scheme is IND-sID-CPA secure, if the advantage in the above security game is negligible for all PPT adversaries.*

4 Proposed CCA Secure Keyed-FHE Scheme

Given a convertible IBFHE scheme $\mathsf{cIBE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{GenerateTK}, \mathsf{Encrypt}, \mathsf{Transform}, \mathsf{Decrypt}, \mathsf{Evaluate})$ for identities of length ℓ which is IND-sID-CPA secure, we construct a CCA-secure keyed-FHE scheme. In the construction, we use a strongly EUF-CMA secure signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ in which the verification key output by Gen has length ℓ. The construction of our CCA-secure keyed-FHE scheme is described as follows.

$\mathsf{Setup}(1^\kappa)$: The setup algorithm first runs $\mathsf{cIBE.Setup}(1^\kappa)$ to obtain $(\mathsf{PP}, \mathsf{MK})$, and calls $\mathcal{S}.\mathsf{Gen}(1^\kappa)$ to obtain a key pair $(\widetilde{vk}, \widetilde{sk})$. Then, it computes

$$\mathsf{TK}_{\mapsto \widetilde{vk}} \leftarrow \mathsf{cIBE.GenerateTK}(\mathsf{PP}, \mathsf{MK}, \widetilde{vk}).$$

Finally, it sets the public key $\mathsf{PK} = \mathsf{PP}$, the decryption key $\mathsf{DK} = \mathsf{MK}$ and the evaluation key $\mathsf{EK} = (\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto \widetilde{vk}})$.

$\mathsf{Enc}(\mathsf{PK}, b \in \{0,1\})$: The encryption algorithm takes as input the public key $\mathsf{PK} = \mathsf{PP}$, and a message bit $b \in \{0,1\}$. It proceeds as follows.

1. Run $\mathcal{S}.\mathsf{Gen}(1^\kappa)$ to obtain a key pair (vk, sk).
2. Compute $\mathsf{CT} \leftarrow \mathsf{cIBE.Encrypt}(\mathsf{PP}, vk, b)$ and $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(sk, \mathsf{CT})$.
3. Output the ciphertext $C = (vk, \mathsf{CT}, \sigma)$.

$\mathsf{Dec}(\mathsf{PK}, \mathsf{DK}, C)$: The decryption algorithm takes as input the public key $\mathsf{PK} = \mathsf{PP}$, the decryption key $\mathsf{DK} = \mathsf{MK}$ and a ciphertext $C = (vk, \mathsf{CT}, \sigma)$. This algorithm first checks whether $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$. If not, it outputs $\perp$. Otherwise, it computes $\mathsf{SK}_{vk} \leftarrow \mathsf{cIBE.Extract}(\mathsf{PP}, \mathsf{MK}, vk)$ and sets $b \leftarrow \mathsf{cIBE.Decrypt}(\mathsf{PP}, \mathsf{SK}_{vk}, \mathsf{CT})$. Then, it outputs the message bit b.

$\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$: This algorithm takes as input the public key $\mathsf{PK} = \mathsf{PP}$, the evaluation key $\mathsf{EK} = (\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto \widetilde{vk}})$, a tuple of ciphertexts $\boldsymbol{C} = (C_1 = (vk_1, \mathsf{CT}_1, \sigma_1), \ldots, C_k = (vk_k, \mathsf{CT}_k, \sigma_k))$ and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$. For $i = 1, \ldots, k$, it proceeds as follows.

1. Check whether $\mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$. If not, it outputs $\perp$.
2. Compute $\widetilde{\mathsf{CT}}_i \leftarrow \mathsf{cIBE.Transform}(\mathsf{PP}, \mathsf{TK}_{\mapsto \widetilde{vk}}, vk_i, \mathsf{CT}_i)$.

Next, it calls $\mathsf{cIBE.Evaluate}$ to obtain $\widetilde{\mathsf{CT}} \leftarrow \mathsf{cIBE.Evaluate}(\mathsf{PP}, \widetilde{vk}, (\widetilde{\mathsf{CT}}_1, \ldots, \widetilde{\mathsf{CT}}_k), f)$. Then, it computes $\tilde{\sigma} \leftarrow \mathcal{S}.\mathsf{Sign}(\widetilde{sk}, \mathsf{CT})$, and outputs the ciphertext $C = (\widetilde{vk}, \widetilde{\mathsf{CT}}, \tilde{\sigma})$.

Correctness. If the underlying convertible IBFHE scheme cIBE satisfies encryption correctness, transformation correctness and evaluation correctness, it is obvious that the above construction satisfies the correctness requirements of keyed-FHE.

Theorem 4. *If the underlying convertible IBFHE scheme is IND-sID-CPA secure, and the signature scheme $\mathcal{S}$ is strongly EUF-CMA secure, then our proposed keyed-FHE scheme is CCA-secure.*

Proof. To prove the CCA security of our proposed keyed-FHE scheme, we consider the following games which is described by its modification from the previous game.

Game 0. This is the original CCA security game between an adversary $\mathcal{A}$ against our scheme and a CCA challenger.

Game 1. In this game, we slightly change the way that the challenger answers the adversary's **DecCT** and **EvalOnCT** queries. Let $C^* = (vk^*, \mathsf{CT}^*, \sigma^*)$ be the challenge ciphertext.

When the adversary $\mathcal{A}$ issues a **DecCT** query on ciphertext $C = (vk, \mathsf{CT}, \sigma)$, the challenger checks whether $vk = vk^*, C \neq C^*$ and $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$. If so, the challenger returns $\perp$; otherwise, it responds as in Game 0.

When the adversary $\mathcal{A}$ issues an **EvalOnCT** query on $\langle \boldsymbol{C} = (C_1, \ldots, C_k), f \rangle$, the challenger first parses C_i as $(vk_i, \mathsf{CT}_i, \sigma_i)$ for each $i \in [k]$. Then, the challenger checks whether there exists $i \in [k]$ such that $vk_i = vk^*, C_i \neq C^*$ and $\mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$. If so, the challenger returns $\perp$; otherwise, it responds as in Game 0.

Game 2. In this game, at the setup phase, except for the list DList, the challenger also maintains another list EList, which is set as $\emptyset$ initially. We also modify the way how the adversary $\mathcal{A}$'s **DecCT** and **EvalOnCT** queries are answered. Let $\mathsf{PK}, \mathsf{DK}, \mathsf{EK} = (\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto \widetilde{vk}})$ be the public key, decryption key and evaluation key respectively, generated by the challenger at the setup phase.

When the adversary $\mathcal{A}$ issues a **DecCT** query on ciphertext $C = (vk, \mathsf{CT}, \sigma)$, the challenger checks whether $vk = vk^*$ or $vk \neq \widetilde{vk}$. If so, the challenger responds as in Game 1; otherwise (i.e., $vk = \widetilde{vk}$), it proceeds as follows:

1. Check whether $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$. If not, return $\perp$.
2. Search the list EList for a record (m, C). If such record does not exist, return $\perp$; otherwise, send m to $\mathcal{A}$.

When the adversary $\mathcal{A}$ issues an **EvalOnCT** query on $\langle \boldsymbol{C} = (C_1, \ldots, C_k), f \rangle$, the challenger first parses C_i as $(vk_i, \mathsf{CT}_i, \sigma_i)$ for each $i \in [k]$. Then, it checks whether there exists $i \in [k]$ such that one of the following conditions holds: (1) $vk_i = vk^*, \mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$ and $C_i \neq C^*$; (2) $vk_i = \widetilde{vk}, \mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$ and the list EList does not contain a record (m_i, C_i). If so, the challenger returns $\perp$ to $\mathcal{A}$; otherwise, the challenger runs $\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$ to obtain a ciphertext C, which is returned to $\mathcal{A}$. In addition, when the ciphertext $C \neq \perp$, the challenger checks whether there exists $i \in [k]$ such that $C_i \in \mathsf{DList}$. If so, the challenger updates the list by $\mathsf{DList} \leftarrow \mathsf{DList} \cup \{C\}$; otherwise, it proceeds as follows.

1. For each $i \in [k]$, if $vk_i = \widetilde{vk}$, the challenger finds the record (m_i, C_i) in the list EList; otherwise (i.e., $vk_i \neq \widetilde{vk}$), the challenger uses the decryption key DK to decrypt C_i with algorithm Dec and obtain a message bit m_i.
2. The challenger computes $m = f(m_1, \ldots, m_k)$ and updates the list by $\mathsf{EList} \leftarrow \mathsf{EList} \cup \{(m, C)\}$.

By the following lemmas, we prove these games are computationally indistinguishable, and in Game 2, the advantage of the adversary is negligible. Therefore, we conclude that the advantage of the adversary in Game 0 (i.e., the original CCA security game) is negligible. This completes the proof of Theorem 4.

Lemma 2. *Suppose that the signature scheme $\mathcal{S}$ is strongly EUF-CMA-secure. Then Game 0 and Game 1 are computationally indistinguishable.*

Proof. Let $C^* = (vk^*, \mathsf{CT}^*, \sigma^*)$ be the challenge ciphertext. Define event E: the adversary $\mathcal{A}$ submits a ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk = vk^*, C \neq C^*$ and $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$ during its **DecCT** or **EvalOnCT** queries. If E does not happen, Game 0 is identical to Game 1. All we have to do is to prove that E happens with negligible probability.

Suppose that E happens with non-negligible probability. Then we can build an algorithm $\mathcal{B}$ that breaks strong $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathcal{S}$ with non-negligible probability. Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the strong $\mathsf{EUF\text{-}CMA}$ security game of the signature scheme $\mathcal{S}$. $\mathcal{B}$ is given the verification key vk^* of the signature scheme $\mathcal{S}$, and simulates Game 1 to the adversary $\mathcal{A}$ as follows.

$\mathcal{B}$ runs Setup to obtain $(\mathsf{PK}, \mathsf{DK}, \mathsf{EK})$, and sends the public key PK to $\mathcal{A}$. Since $\mathcal{B}$ knows the decryption key DK and the evaluation key EK associated with PK, thus it is able to answer all queries made by the adversary. At some point, $\mathcal{A}$ asks for the challenge ciphertext. $\mathcal{B}$ proceeds as follows.

1. Choose a message bit $b^* \in \{0, 1\}$ uniformly at random.
2. Compute $\mathsf{CT}^* \leftarrow \mathsf{cIBE.Encrypt}(\mathsf{PK}, vk^*, b^*)$.
3. Issue the signing query on CT^* to its challenger $\mathcal{C}$ to obtain the corresponding signature σ^*.

4. Set the challenge ciphertext $C^* = (vk^*, \mathsf{CT}^*, \sigma^*)$ and send it to the adversary $\mathcal{A}$.

Suppose E happens during the simulation (i.e., the adversary submits a ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk = vk^*, C \neq C^*$ and $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$ during its **DecCT** or **EvalOnCT** queries), $\mathcal{B}$ outputs (CT, σ) which is not equal to $(\mathsf{CT}^*, \sigma^*)$, as its forgery of the signature scheme $\mathcal{S}$. Thus, if E happens with non-negligible probability, then $\mathcal{B}$ can break strong $\mathsf{EUF\text{-}CMA}$ security of the signature scheme $\mathcal{S}$ with non-negligible probability.

Lemma 3. *Suppose that the signature scheme $\mathcal{S}$ is strongly EUF-CMA-secure. Then Game 1 and Game 2 are computationally indistinguishable.*

Proof. Let $\mathsf{EK} = (\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto\widetilde{vk}})$ be the evaluation key. Game 2 is the same as Game 1 except for the way of answering the adversary $\mathcal{A}$'s **DecCT** and **EvalOnCT** queries when $\mathcal{A}$ submits a ciphertext $C = (vk, \mathsf{CT}, \sigma)$ such that $vk = \widetilde{vk}$ and $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$. Recall that in our security definition of keyed-FHE, the adversary cannot issue the decryption or evaluation queries if it requests the evaluation key to be exposed. Since our proposed scheme satisfies the requirement of evaluation correctness, it is easy to observe that when $\mathcal{A}$ submits a ciphertext $C = (vk = \widetilde{vk}, \mathsf{CT}, \sigma)$ during its **DecCT** or **EvalOnCT** queries such that C is the return of $\mathcal{A}$'s some **EvalOnCT** query, the challenger's response is the same in Game 1 and Game 2.

Define event E: the adversary $\mathcal{A}$ submits a ciphertext $C = (vk = \widetilde{vk}, \mathsf{CT}, \sigma)$ during its **DecCT** or **EvalOnCT** queries such that $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$ and C is not the response to $\mathcal{A}$'s some **EvalOnCT** query. If E does not happen, Game 1 is identical to Game 2. All we have to do is to prove that E happens with negligible probability.

One can prove that if the signature scheme $\mathcal{S}$ is strongly $\mathsf{EUF\text{-}CMA}$-secure, then event E happens with negligible probability. We omit the details due to its similarity of Lemma 2.

Lemma 4. *If the underlying convertible IBFHE scheme is IND-sID-CPA-secure, then in Game 2, the advantage of the adversary is negligible.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that achieves a non-negligible advantage in Game 2. Then we can build an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to attack the underlying convertible IBFHE scheme cIBE in the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game with a non-negligible advantage. Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game of the convertible IBFHE scheme cIBE. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

Setup. $\mathcal{B}$ first runs $\mathcal{S}.\mathsf{Gen}$ twice to obtain two key pairs (vk^*, sk^*) and $(\widetilde{vk}, \widetilde{sk})$. Then, it submits $(vk^*, \widetilde{vk})$ to $\mathcal{C}$ as its target identity and designated identity, and $\mathcal{C}$ returns the public parameters PP of the convertible IBFHE scheme cIBE and the transformation key $\mathsf{TK}_{\mapsto\widetilde{vk}}$ for identity $\widetilde{vk}$ to $\mathcal{B}$. Next, $\mathcal{B}$ sets the public key $\mathsf{PK} = \mathsf{PP}$, the evaluation key $\mathsf{EK} = (\widetilde{vk}, \widetilde{sk}, \mathsf{TK}_{\mapsto\widetilde{vk}})$, and

sends the public key PK to the adversary $\mathcal{A}$. In addition, $\mathcal{B}$ maintains two lists DList and EList, which are set as $\emptyset$ initially.

Query phase 1. The adversary $\mathcal{A}$ adaptively issues the following queries:

- **DecCT**$\langle C \rangle$: $\mathcal{B}$ first parses the ciphertext C as $(vk, \mathsf{CT}, \sigma)$. Then, it checks whether $\mathcal{S}.\mathsf{Vrfy}(vk, \mathsf{CT}, \sigma) = 1$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ proceeds as follows.
  1. If $vk = vk^*$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
  2. If $vk \neq \widetilde{vk}$, $\mathcal{B}$ issues **GetSK** query on $\langle vk \rangle$ to its challenger $\mathcal{C}$ to obtain a private key SK_{vk} for identity vk, and uses the private key SK_{vk} to decrypt CT with algorithm $\mathsf{cIBE.Decrypt}$. The result is sent back to $\mathcal{A}$.
  3. If $vk = \widetilde{vk}$ and $C \in \mathsf{DList}$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
  4. Otherwise (i.e., $vk = \widetilde{vk}$ and $C \notin \mathsf{DList}$), $\mathcal{B}$ search the list EList for a record (m, C). If such record does not exist, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ sends the message bit m to the adversary $\mathcal{A}$.
- **EvalOnCT**$\langle \boldsymbol{C} = (C_1, \ldots, C_k), f \rangle$: For each $i \in [k]$, $\mathcal{B}$ parses C_i as $(vk_i, \mathsf{CT}_i, \sigma_i)$. Then, $\mathcal{B}$ checks whether there exists $i \in [k]$ such that one of the following conditions holds: (1) $vk_i = vk^*, \mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$ and $C_i \neq C^*$; (2) $vk_i = \widetilde{vk}, \mathcal{S}.\mathsf{Vrfy}(vk_i, \mathsf{CT}_i, \sigma_i) = 1$ and the list EList does not contain a record (m_i, C_i). If so, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$; otherwise, $\mathcal{B}$ runs $\mathsf{Eval}(\mathsf{PK}, \mathsf{EK}, \boldsymbol{C}, f)$ to obtain a ciphertext C, which is returned to $\mathcal{A}$. In addition, when the ciphertext $C \neq \perp$, $\mathcal{B}$ checks whether there exists $i \in [k]$ such that $C_i \in \mathsf{DList}$. If so, $\mathcal{B}$ updates the list by $\mathsf{DList} \leftarrow \mathsf{DList} \cup \{C\}$; otherwise, $\mathcal{B}$ proceeds as follows.
  1. For each $i \in [k]$, if $vk_i = \widetilde{vk}$, $\mathcal{B}$ finds the record (m_i, C_i) in the list EList; otherwise (i.e., $vk_i \neq \widetilde{vk}$), $\mathcal{B}$ issues **GetSK** query on $\langle vk_i \rangle$ to its challenger $\mathcal{C}$ to obtain a private key SK_{vk_i} for identity vk_i, and uses the private key SK_{vk_i} to decrypt CT_i with algorithm $\mathsf{cIBE.Decrypt}$ and obtain a message bit m_i.
  2. $\mathcal{B}$ computes $m = f(m_1, \ldots, m_k)$ and updates the list by $\mathsf{EList} \leftarrow \mathsf{EList} \cup \{(m, C)\}$.
- **RevEK**: The challenger sends the evaluation key EK to $\mathcal{A}$.

Challenge. Firstly, $\mathcal{B}$ asks $\mathcal{C}$ for its challenge ciphertext of the convertible IBFHE scheme cIBE, and receives the ciphertext CT^*. Then, $\mathcal{B}$ computes $\sigma^* \leftarrow \mathcal{S}.\mathsf{Sign}(sk^*, \mathsf{CT}^*)$, and sets the challenge ciphertext $C^* = (vk^*, \mathsf{CT}^*, \sigma^*)$. Finally, $\mathcal{B}$ sends C^* to the adversary $\mathcal{A}$. In addition, $\mathcal{B}$ updates the list by $\mathsf{DList} \leftarrow \mathsf{DList} \cup \{C^*\}$.

Query phase 2. The adversary $\mathcal{A}$ continues to adaptively issue **DecCT**, **EvalOnCT** and **RevEK** queries. $\mathcal{B}$ responds as Query phase 1.

Guess. The adversary $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$. $\mathcal{B}$ also takes b as its output.

It is easy to observe that, $\mathcal{B}$'s simulation is perfect. Thus, if $\mathcal{A}$ has a non-negligible advantage in Game 2, then $\mathcal{B}$ attacks the underlying convertible IBFHE scheme cIBE in the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game with a non-negligible advantage.

5 Proposed Convertible IBFHE Scheme

We denote $\mathsf{SampleUS}(\mathbb{Z}_q^n; r_S)$ as a sample algorithm that chooses an element in $\mathbb{Z}_q^n$ uniformly at random with the randomness r_S, $\mathsf{SampleGX}(\mathbb{Z}_q, \bar{\Psi}_\alpha; r_X)$ as a sample algorithm that chooses an element in $\mathbb{Z}_q$ from the distribution $\bar{\Psi}_\alpha$ with the randomness r_X, $\mathsf{SampleGY}(\mathbb{Z}_q^m, \bar{\Psi}_\alpha^m; r_Y)$ as a sample algorithm that chooses an element in $\mathbb{Z}_q^m$ from the distribution $\bar{\Psi}_\alpha^m$ with the randomness r_Y, and $\mathsf{SampleURs}(\{-1,1\}^{m\times m}; r_R)$ as a sample algorithm that chooses ℓ-elements in domain $\{-1,1\}^{m\times m}$ uniformly at random with the randomness r_R. Let $\mathsf{F}_D : \mathcal{K}_D \times \{0,1\}^* \to \{-1,1\}^\ell$, $\mathsf{F}_S : \mathcal{K}_S \times \{0,1\}^* \times \{0,1\}^{2\kappa} \times [N] \to \mathcal{R}_{\mathsf{SampleUS}}$, $\mathsf{F}_X : \mathcal{K}_X \times \{0,1\}^* \times \{0,1\}^{2\kappa} \times [N] \to \mathcal{R}_{\mathsf{SampleGX}}$, $\mathsf{F}_Y : \mathcal{K}_Y \times \{0,1\}^* \times \{0,1\}^{2\kappa} \times [N] \to \mathcal{R}_{\mathsf{SampleGY}}$, $\mathsf{F}_R : \mathcal{K}_R \times \{0,1\}^* \times \{0,1\}^{2\kappa} \times [N] \to \mathcal{R}_{\mathsf{SampleURs}}$ be puncturable PRFs, and let $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ be a pseudorandom generator. Let $i\mathcal{O}$ be a program indistinguishability obfuscator. Our proposed convertible identity-based (leveled) FHE scheme consists of the following algorithms:

$\mathsf{Setup}(1^\kappa, 1^L)$: On input a security parameter κ and a number of levels L (maximum circuit depth to support), this algorithm first chooses the parameters $(q, n, m, \sigma, \alpha)$ as specified in Sect. 5.1 below. Let $N = (2m+1)\cdot(\lfloor \log q \rfloor + 1)$. It then invokes $\mathsf{TrapGen}(q, n)$ to generate a uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$ and a short basis $T_A \in \mathbb{Z}^{m\times m}$ for $\Lambda_q^\perp(A)$. It also chooses uniformly random matrices $B_0, B_1, \ldots, B_\ell \in \mathbb{Z}_q^{n\times m}$, and a uniformly random vector $u \in \mathbb{Z}_q^n$. Next, it chooses puncturable PRF keys $\mathrm{K}_D \leftarrow \mathcal{K}_D, \mathrm{K}_S \leftarrow \mathcal{K}_S, \mathrm{K}_X \leftarrow \mathcal{K}_X, \mathrm{K}_Y \leftarrow \mathcal{K}_Y, \mathrm{K}_R \leftarrow \mathcal{K}_R$, and creates an obfuscation of the program ProduceCT Fig. 1 as $\mathbf{P}^{\mathsf{Enc}} \leftarrow i\mathcal{O}(\kappa, \mathsf{ProduceCT})$,

ProduceCT:
Input: Identity $\mathsf{ID} \in \{0,1\}^*$, a message $b \in \{0,1\}$, and randomness $r \in \{0,1\}^\kappa$.
Constants: PRF keys $\mathrm{K}_D, \mathrm{K}_S, \mathrm{K}_X, \mathrm{K}_Y$ and K_R.

1. Compute $\mathsf{id} = \mathsf{F}_D(\mathrm{K}_D, \mathsf{ID}) \in \{-1,1\}^\ell$.
2. Compute $t = \mathsf{PRG}(r)$.
3. For each $i \in [N]$, do the following:
   (a) compute $r_{S,i} = \mathsf{F}_S(\mathrm{K}_S, \mathsf{ID}, t, i), r_{X,i} = \mathsf{F}_X(\mathrm{K}_X, \mathsf{ID}, t, i), r_{Y,i} = \mathsf{F}_Y(\mathrm{K}_Y, \mathsf{ID}, t, i)$ and $r_{R,i} = \mathsf{F}_R(\mathrm{K}_R, \mathsf{ID}, t, i)$;
   (b) evaluate $(c_{i,0}, c_{i,1}) = \mathsf{ABBEnc0}(\mathsf{PP}_{\mathsf{ABB}}, \mathsf{id}, r_{S,i}, r_{X,i}, r_{Y,i}, r_{R,i})$.
4. Set $c_{\mathsf{id}} = \begin{bmatrix} c_{1,0} | c_{1,1}^\top \\ \vdots \\ c_{N,0} | c_{N,1}^\top \end{bmatrix} \in \mathbb{Z}_q^{N\times(2m+1)}$ and compute $C_{\mathsf{ID}} = \mathsf{Flatten}(b \cdot I_N + \mathsf{BitDecomp}(c_{\mathsf{id}}))$.
5. Output: (t, C_{ID}).

Fig. 1. Program ProduceCT

Finally, it outputs the public parameters

$$\mathsf{PP} = \Big(\mathsf{PP}_{\mathsf{ABB}} = (A, B_0, B_1, \ldots, B_\ell, u), \mathbf{P}^{\mathsf{Enc}}, \mathsf{PRG}, \mathsf{F}_D, \mathsf{F}_S, \mathsf{F}_X, \mathsf{F}_Y, \mathsf{F}_R\Big),$$

and master key $\mathsf{MK} = \Big(T_A, \mathrm{K}_D, \mathrm{K}_S, \mathrm{K}_X, \mathrm{K}_Y, \mathrm{K}_R\Big)$.

$\mathsf{Extract}(\mathsf{PP}, \mathsf{MK}, \mathsf{ID})$: On input public parameters PP, a master key MK, and an identity $\mathsf{ID} \in \{0,1\}^*$, this algorithm first sets $\mathsf{id} = \mathsf{F}_D(\mathrm{K}_D, \mathsf{ID}) = (d_1, \ldots, d_\ell) \in \{-1,1\}^\ell$ and evaluates $\boldsymbol{e}_{\mathsf{ID}} \leftarrow \mathsf{SampleLeft}(A, B_0 + \sum_{i=1}^{\ell} d_i B_i, T_A, u, \sigma)$ to obtain a short vector in $\Lambda_q^u(F_{\mathsf{id}})$, where $F_{\mathsf{id}} = A \mid B_0 + \sum_{i=1}^{\ell} d_i B_i$ and $\boldsymbol{e}_{\mathsf{ID}}$ is distributed as $D_{\Lambda_q^u(F_{\mathsf{id}}),\sigma}$. Then, it sets $\boldsymbol{s}_{\mathsf{ID}} = (1, -\boldsymbol{e}_{\mathsf{ID}})$ and $\boldsymbol{v}_{\mathsf{ID}} = \mathsf{Powersof2}(\boldsymbol{s}_{\mathsf{ID}})$, and outputs the private key $\mathsf{SK}_{\mathsf{ID}} = \boldsymbol{v}_{\mathsf{ID}}$ for identity ID.

$\mathsf{GenerateTK}(\mathsf{PP}, \mathsf{MK}, \widetilde{\mathsf{ID}})$: On input public parameter PP, a master key MK, and an identity $\widetilde{\mathsf{ID}} \in \{0,1\}^*$, this algorithm creates an obfuscation of the program $\mathsf{ConvertTAID}$ Fig. 2 as $\mathbf{P}^{\mathsf{Trans}} \leftarrow i\mathcal{O}(\kappa, \mathsf{ConvertTAID})$, and outputs the transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}} = \Big(\widetilde{\mathsf{ID}}, \mathbf{P}^{\mathsf{Trans}}\Big)$.

$\mathsf{Encrypt}(\mathsf{PP}, \mathsf{ID}, b)$: On input public parameters PP, an identity $\mathsf{ID} \in \{0,1\}^*$, and a message $b \in \{0,1\}$, the encryption algorithm first chooses $r \in \{0,1\}^\kappa$ uniformly at random. Then, it computes $(t, C_{\mathsf{ID}}) = \mathbf{P}^{\mathsf{Enc}}(\mathsf{ID}, b, r)$, and outputs the ciphertext $\mathsf{CT} = (t, C_{\mathsf{ID}})$.

$\mathsf{Transform}(\mathsf{PP}, \mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}}, \mathsf{ID}, \mathsf{CT})$: On input public parameters PP, a transformation key $\mathsf{TK}_{\mapsto\widetilde{\mathsf{ID}}} = (\widetilde{\mathsf{ID}}, \mathbf{P}^{\mathsf{Trans}})$, a ciphertext $\mathsf{CT} = (t, C_{\mathsf{ID}})$ for an identity ID, this algorithm computes $(\tilde{t}, C_{\widetilde{\mathsf{ID}}}) = \mathbf{P}^{\mathsf{Trans}}(\mathsf{ID}, \mathsf{CT})$, and outputs the transformed ciphertext $\widetilde{\mathsf{CT}} = (\tilde{t}, C_{\widetilde{\mathsf{ID}}})$.

$\mathsf{Decrypt}(\mathsf{PP}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{CT})$: The decryption algorithm takes as input public parameters PP, a private key $\mathsf{SK}_{\mathsf{ID}} = \boldsymbol{v}_{\mathsf{ID}}$, and a ciphertext $\mathsf{CT} = (t, C_{\mathsf{ID}})$. Observe that the first $\lfloor \log q \rfloor + 1$ coefficients of $\boldsymbol{v}_{\mathsf{ID}}$ are $1, 2, \ldots, 2^{\lfloor \log q \rfloor}$. Among these coefficients, let $v_{\mathsf{ID},i} = 2^i$ be in $(q/4, q/2]$. Let $C_{\mathsf{ID},i}$ be the i-th row of C_{ID}. This algorithm computes $x_i \leftarrow \langle C_{\mathsf{ID},i}, \boldsymbol{v}_{\mathsf{ID}} \rangle$ and outputs $\mu' = \lfloor x_i / v_{\mathsf{ID},i} \rceil$.

$\mathsf{Evaluate}(\mathsf{PP}, \mathsf{ID}, \boldsymbol{CT}, f)$: The evaluation algorithm takes as input public parameters PP, a tuple of ciphertext $\boldsymbol{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k)$ under an identity ID and a Boolean circuit $f : \{0,1\}^k \to \{0,1\}$. It is a remarkable fact that, Boolean circuits computed over encryptions of binary values can be converted to use only NAND gates [31]. Let $\mathsf{CT}_i = (t_i, ct_i)$ be an encryption of $b_i \in \{0,1\}$ for all $i \in [k]$, the NAND homomorphic operation is described below:
$\mathsf{NAND}(ct_1, ct_2)$: To NAND ciphertexts $ct_1, ct_2 \in \mathbb{Z}_q^{N \times N}$, output $\mathsf{Flatten}(I_N - ct_1 \cdot ct_2)$.

Let ct be the result of $f(ct_1, \ldots, ct_k)$ through appropriate leveled application of the NAND homomorphic operation. The algorithm chooses a random value $t \in \{0,1\}^{2\kappa}$ and outputs the resulting ciphertext $\mathsf{CT} = (t, ct)$.

$\mathsf{ABBEnc0}(\mathsf{PP}_{\mathsf{ABB}}, \mathsf{id}, r_S, r_X, r_Y, r_R)$: On input public parameters $\mathsf{PP}_{\mathsf{ABB}}$, an identity $\mathsf{id} = (d_1, \ldots, d_\ell) \in \{-1,1\}^\ell$ and randomness $r_S \in \mathcal{R}_{\mathsf{SampleUS}}, r_X \in$

ConvertTAID:
Input: Identity $\mathsf{ID} \in \{0,1\}^*$, and ciphertext $\mathsf{CT} = (t, C_{\mathsf{ID}}) \in \{0,1\}^{2\kappa} \times \mathbb{Z}_q^{N \times N}$.
Constants: PRF keys $\mathrm{K}_D, \mathrm{K}_S, \mathrm{K}_X, \mathrm{K}_Y$ and K_R.

1. If ID is equal to $\widetilde{\mathsf{ID}}$, output CT.
2. Compute $\mathsf{id} = \mathsf{F}_D(\mathrm{K}_D, \mathsf{ID}) \in \{-1,1\}^\ell$ and $\widetilde{\mathsf{id}} = \mathsf{F}_D(\mathrm{K}_D, \widetilde{\mathsf{ID}}) \in \{-1,1\}^\ell$.
3. For each $i \in [N]$, do the following:
   (a) compute $r_{S,i} = \mathsf{F}_S(\mathrm{K}_S, \mathsf{ID}, t, i), r_{X,i} = \mathsf{F}_X(\mathrm{K}_X, \mathsf{ID}, t, i), r_{Y,i} = \mathsf{F}_Y(\mathrm{K}_Y, \mathsf{ID}, t, i)$ and $r_{R,i} = \mathsf{F}_R(\mathrm{K}_R, \mathsf{ID}, t, i)$;
   (b) evaluate $(c_{i,0}, c_{i,1}) = \mathsf{ABBEnc0}(\mathsf{PP}_{\mathsf{ABB}}, \mathsf{id}, r_{S,i}, r_{X,i}, r_{Y,i}, r_{R,i})$.
4. Set $c_{\mathsf{id}} = \begin{bmatrix} c_{1,0} | c_{1,1}^\top \\ \vdots \\ c_{N,0} | c_{N,1}^\top \end{bmatrix} \in \mathbb{Z}_q^{N \times (2m+1)}$.
5. Check whether there exists $b \in \{0,1\}$ such that $C_{\mathsf{ID}} = \mathsf{Flatten}(b \cdot I_N + \mathsf{BitDecomp}(c_{\mathsf{id}}))$.
   If not, output $\perp$.
6. For each $i \in [N]$, do the following:
   (a) compute $\tilde{r}_{S,i} = \mathsf{F}_S(\mathrm{K}_S, \widetilde{\mathsf{ID}}, t, i), \tilde{r}_{X,i} = \mathsf{F}_X(\mathrm{K}_X, \widetilde{\mathsf{ID}}, t, i), \tilde{r}_{Y,i} = \mathsf{F}_Y(\mathrm{K}_Y, \widetilde{\mathsf{ID}}, t, i)$ and $\tilde{r}_{R,i} = \mathsf{F}_R(\mathrm{K}_R, \widetilde{\mathsf{ID}}, t, i)$;
   (b) evaluate $(\tilde{c}_{i,0}, \tilde{c}_{i,1}) = \mathsf{ABBEnc0}(\mathsf{PP}_{\mathsf{ABB}}, \widetilde{\mathsf{id}}, \tilde{r}_{S,i}, \tilde{r}_{X,i}, \tilde{r}_{Y,i}, \tilde{r}_{R,i})$.
7. Set $c_{\widetilde{\mathsf{id}}} = \begin{bmatrix} \tilde{c}_{1,0} | \tilde{c}_{1,1}^\top \\ \vdots \\ \tilde{c}_{N,0} | \tilde{c}_{N,1}^\top \end{bmatrix} \in \mathbb{Z}_q^{N \times (2m+1)}$ and compute $C_{\widetilde{\mathsf{ID}}} = \mathsf{Flatten}(b \cdot I_N + \mathsf{BitDecomp}(c_{\widetilde{\mathsf{id}}}))$.
8. Output: $(t, C_{\widetilde{\mathsf{ID}}})$.

Fig. 2. Program ConvertTAID

$\mathcal{R}_{\mathsf{SampleGX}}, r_Y \in \mathcal{R}_{\mathsf{SampleGY}}, r_R \in \mathcal{R}_{\mathsf{SampleURs}}$, this algorithm proceeds as follows.

1. Let $F_{\mathsf{id}} = A \mid B_0 + \sum_{i=1}^{\ell} d_i B_i \in \mathbb{Z}_q^{n \times 2m}$.
2. Evaluate $s = \mathsf{SampleUS}(\mathbb{Z}_q^n; r_S), x = \mathsf{SampleGX}(\mathbb{Z}_q, \bar{\Psi}_\alpha; r_X)$ and $y = \mathsf{SampleGY}(\mathbb{Z}_q^m, \bar{\Psi}_\alpha^m; r_Y)$.
3. Evaluate $(R_1, \ldots, R_\ell) = \mathsf{SampleURs}(\{-1,1\}^{m \times m}; r_R)$.
4. Set $R_{\mathsf{id}} = \sum_{i=1}^{\ell} d_i R_i$ and compute $z = R_{\mathsf{id}}^\top y$.
5. Compute $c_0 = u^\top s + x \in \mathbb{Z}_q$, $c_1 = F_{\mathsf{id}}^\top s + \begin{bmatrix} y \\ z \end{bmatrix} \in \mathbb{Z}_q^{2m}$, and return (c_0, c_1).

5.1 Parameters and Correctness

Let $\mathsf{CT}_1 = (t_1, ct_1 = \mathsf{Flatten}(b_1 \cdot I_N + \mathsf{BitDecomp}(c_1)))$ be an encryption of $b_1 \in \{0,1\}$ under an identity ID. Recall that c_1 is a N-row matrix whose rows

are encryptions of 0 generated by using $\mathsf{ABBEnc0}$, and the private key $\boldsymbol{v}_{\mathsf{ID}} = \mathsf{Powersof2}(\boldsymbol{s}_{\mathsf{ID}})$. By Claim 1, we have

$$ct_1 \cdot \boldsymbol{v}_{\mathsf{ID}} = (b_1 \cdot I_N + \mathsf{BitDecomp}(c_1)) \cdot \boldsymbol{v}_{\mathsf{ID}} = b_1 \cdot \boldsymbol{v}_{\mathsf{ID}} + c_1 \cdot \boldsymbol{s}_{\mathsf{ID}}.$$

Let $c_{1,i}$ be the i-th row of the matrix c_1, and let $v_{\mathsf{ID},i}$ be the i-th coefficient of $\boldsymbol{v}_{\mathsf{ID}}$. Algorithm $\mathsf{Decrypt}$ only uses the i-th coefficient of $ct_1 \cdot \boldsymbol{v}_{\mathsf{ID}}$, which is $x_i = b_1 \cdot v_{\mathsf{ID},i} + c_{1,i} \cdot \boldsymbol{s}_{\mathsf{ID}}$. If $c_{1,i}$ is properly generated using $\mathsf{ABBEnc0}$, then the norm of the inner product $c_{1,i} \cdot \boldsymbol{s}_{\mathsf{ID}}$ is bounded $w.h.p$ by $B = q\sigma\ell m\alpha\omega(\sqrt{\log m}) + O(\sigma m^{3/2})$ by Lemma 24 of [1]. Similarly as in [31], if $B < q/8$ and $v_{\mathsf{ID},i} \in (q/4, q/2]$, then $x_i/v_{\mathsf{ID},i}$ differs from b_1 by at most $(q/8)/v_{\mathsf{ID},i} < 1/2$, and algorithm $\mathsf{Decrypt}$ uses rounding to output the correct value of b_1.

It is clear that our system satisfies transformation correctness if encryption correctness holds. Regarding evaluation correctness, let $\mathsf{CT}_2 = (t_2, ct_2 = \mathsf{Flatten}(b_2 \cdot I_N + \mathsf{BitDecomp}(c_2)))$ be an encryption of another bit $b_2 \in \{0,1\}$ under the same identity ID, where c_2 is also a N-row matrix whose rows are encryptions of 0 generated by using $\mathsf{ABBEnc0}$. We have

$$\mathsf{NAND}(ct_1, ct_2) \cdot \boldsymbol{v}_{\mathsf{ID}} = (I_N - ct_1 \cdot ct_2) \cdot \boldsymbol{v}_{\mathsf{ID}} = (1 - b_1 b_2) \cdot \boldsymbol{v}_{\mathsf{ID}} - b_2 \cdot (c_1 \cdot \boldsymbol{s}_{\mathsf{ID}}) - ct_1 \cdot (c_2 \cdot \boldsymbol{s}_{\mathsf{ID}}).$$

Note that NAND maintains the invariant that if ct_1 and ct_2 are encryptions of messages in $\{0,1\}$, then the output ciphertext is also encryption of message in $\{0,1\}$. After an NAND homomorphic operation, the error is increased by a factor of at most $N + 1$.

Recall that we represent the homomorphic function f over encryptions of binary values as a Boolean circuit that can be converted to use only NAND gates. Through appropriate leveled application of the NAND homomorphic operation, the final ciphertext's error will be bounded by $(N+1)^L \cdot B$, where L is the NAND-depth of the circuit. As long as $(N+1)^L \cdot B < q/8$, the decryption algorithm will decrypt correctly.

Hence, for the system to work correctly and evaluate a circuit of depth L, we set the parameters $(q, n, m, \sigma, \alpha)$ that satisfy the following requirements, taking n to be the security parameter κ:

- the error term has magnitude at most $q/8$ $w.h.p$ (i.e. $B \cdot (N+1)^L < q/8$),
- algorithm $\mathsf{TrapGen}$ can operate (i.e. $m > 6n \log q$),
- σ is sufficiently large for $\mathsf{SampleLeft}$ and $\mathsf{SampleRight}$ (i.e. $\sigma > \ell m\omega(\sqrt{\log m})$),
- Regev's reduction applies [49] (i.e. $\alpha q > 2\sqrt{n}$),
- our security reduction applies (i.e. $q > 2Q + 4$, where Q is the number of private key queries from the adversary).

5.2 Security

We now state the security theorem of our proposed scheme.

Theorem 5. *If the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-LWE assumptions holds, the proposed convertible IBFHE scheme is IND-sID-CPA secure.*

Proof. See the full version of this paper.

6 Conclusion

We introduced a new primitive called convertible IBFHE which is an IBFHE with an additional transformation functionality. We showed that CCA-secure keyed-FHE can be constructed from IND-sID-CPA-secure convertible IBFHE and strongly EUF-CMA-secure signature. Utilizing the recent advances in indistinguishability obfuscation, we presented a concrete construction of IND-sID-CPA-secure convertible IBFHE without random oracles, and yielded the first CCA-secure keyed-FHE scheme in the standard model. Since indistinguishability obfuscation is a slightly cumbersome primitive currently, thus it would be interesting to construct an efficient IND-sID-CPA-secure convertible IBFHE without using indistinguishability obfuscation, even in the random oracle model.

Acknowledgement. We are grateful to the anonymous reviewers for their helpful comments. The work of Junzuo Lai was supported by National Natural Science Foundation of China (Nos. 61572235, 61300226), Research Fund for the Doctoral Program of Higher Education of China (No. 20134401120017), Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2015A030306045), Natural Science Foundation of Guangdong Province (No. 2014A030310156), Pearl River S&T Nova Program of Guangzhou and Fundamental Research Funds for the Central Universities (No. 21615445). The work of Jian Weng was supported by National Natural Science Foundation of China (Nos. 61272413, 61472165, 61133014).

References

1. 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)
2. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)
3. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, p. 1. Springer, Heidelberg (1999)
4. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: Proceedings of 26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, Freiburg, Germany, pp. 75–86, 26–28 February 2009
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 1. Springer, Heidelberg (2001)
6. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012)
7. Bellare, M., Ristenpart, T.: Simulation without the artificial abort: simplified proof and improved concrete security for waters' IBE scheme. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 407–424. Springer, Heidelberg (2009)
8. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)

9. Boneh, D., Segev, G., Waters, B., Targeted malleability: homomorphic encryption for restricted computations. In: Innovations in Theoretical Computer Science, Cambridge, MA, USA, pp. 350–366, 8–10 January 2012
10. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
11. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014)
12. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
13. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in LWE-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 1–13. Springer, Heidelberg (2013)
14. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325 (2012)
15. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, CA, USA, pp. 575–584, 1–4 June 2013
16. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
17. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
18. 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)
19. 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)
20. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 281–300. Springer, Heidelberg (2012)
21. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
22. 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, p. 45. Springer, Heidelberg (2002)
23. Desmedt, Y., Iovino, V., Persiano, G., Visconti, I.: Controlled homomorphic encryption: definition and construction. IACR Cryptology ePrint Arch. **2014**, 989 (2014)
24. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000)
25. Emura, K., Hanaoka, G., Ohtake, G., Matsuda, T., Yamada, S.: Chosen ciphertext secure keyed-homomorphic public-key encryption. In: Hanaoka, G., Kurosawa, K. (eds.) PKC 2013. LNCS, vol. 7778, pp. 32–50. Springer, Heidelberg (2013)

26. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014)
27. Garg, S., Gentry, C., Halevi S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, Berkeley, CA, USA, pp. 40–49, 26–29 October 2013
28. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
29. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer, Heidelberg (2012)
30. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
31. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
32. 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)
33. Hofheinz, D., Kamath, A., Koppula, V., Waters, B.: Adaptively secure constrained pseudorandom functions. IACR Cryptology ePrint Arch. **2014**, 720 (2014)
34. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
35. Hohenberger, S., Koppula, V., Waters, B.: Adaptively secure puncturable pseudorandom functions in the standard model. IACR Cryptology ePrint Arch. **2014**, 521 (2014)
36. Hohenberger, S., Sahai, A., Waters, B.: Replacing a random Oracle: full domain hash from indistinguishability obfuscation. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 201–220. Springer, Heidelberg (2014)
37. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013)
38. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, Berlin, Germany, pp. 669–684, 4–8 November (2013)
39. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307. Springer, Heidelberg (2013)
40. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer, Heidelberg (2014)
41. 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)
42. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437 (1990)

43. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem,extended abstract. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, pp. 333–342, 31 May - 2 June (2009)
44. Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 517–534. Springer, Heidelberg (2007)
45. M. Prabhakaran, M. Rosulek. Homomorphic encryption with CCA security. In Automata, Languages, Programming, 35th International Colloquium, ICALP: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations, pp. 667–678, (2008)
46. M. Prabhakaran and M. Rosulek. Towards robust computation on encrypted data. In Advances in Cryptology - ASIACRYpPT , 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7–11, Proceedings, pp. 216–233, (2008)
47. Rackoff, Charles, Simon, Daniel R.: Non-interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, Joan (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
48. Ramchen, K., Waters, B.: Fully secure and fast signing from obfuscation. IACR Cryptology ePrint Archive **2014**, 523 (2014)
49. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22–24, pp. 84–93, (2005)
50. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **32**(4), 169–178 (1978)
51. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17–18 , New York, NY, USA, pp. 543–553, 1999., October 1999
52. Sahai, A., Waters, B., How to use indistinguishability obfuscation, : deniable encryption, and more. In Symposium on Theory of Computing, STOC , New York, NY, USA, May 31 - June 03, pp. 475–484, (2014)
53. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Public Key Cryptography, pp. 420–443, (2010)
54. Stehlé, D., Steinfeld, R.: Faster fully homomorphic encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 377–394. Springer, Heidelberg (2010)
55. 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)
56. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
57. Waters, B.: A punctured programming approach to adaptively secure functional encryption. IACR Cryptology ePrint Arch. **2014**, 588 (2014)

On the Key Dependent Message Security of the Fujisaki-Okamoto Constructions

Fuyuki Kitagawa[1,2(✉)], Takahiro Matsuda[2], Goichiro Hanaoka[2], and Keisuke Tanaka[1]

[1] Tokyo Institute of Technology, Tokyo, Japan
{kitagaw1,keisuke}@is.titech.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
{t-matsuda,hanaoka-goichiro}@aist.go.jp

Abstract. In PKC 1999, Fujisaki and Okamoto showed how to convert any public key encryption (PKE) scheme secure against chosen plaintext attacks (CPA) to a PKE scheme which is secure against chosen ciphertext attacks (CCA) in the random oracle model. Surprisingly, the resulting CCA secure scheme has almost the same efficiency as the underlying CPA secure scheme. Moreover, in J. Cryptology 2013, they proposed more efficient conversion by using the hybrid encryption framework.

In this work, we clarify whether these two constructions are also secure in the sense of *key dependent message security* against chosen ciphertext attacks (KDM-CCA security), under exactly the same assumptions on the building blocks as those used by Fujisaki and Okamoto. Specifically, we show two results: Firstly, we show that the construction proposed in PKC 1999 does not satisfy KDM-CCA security generally. Secondly, on the other hand, we show that the construction proposed in J. Cryptology 2013 satisfies KDM-CCA security.

Keywords: Public key encryption · Key dependent message security · Chosen ciphertext security

1 Introduction

1.1 Background and Motivation

Security against chosen ciphertext attacks (CCA) has been considered as a desirable security notion for public key encryption (PKE) schemes. In order to take adversaries who mount active attacks into consideration, it is desirable that PKE schemes satisfy CCA security. Moreover, since CCA security implies non-malleability [7,18], it is considered that CCA security is strong enough for many applications. Therefore, many standardization bodies for public key cryptography judge whether they include a PKE scheme or not mainly based on whether the scheme satisfies CCA security [28].

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 99–129, 2016.
DOI: 10.1007/978-3-662-49384-7_5

On these backgrounds, it has been widely studied how to construct a practical CCA secure PKE scheme [10,19–21,26]. Among them, the constructions proposed by Fujisaki and Okamoto [19,21] are one of the most famous constructions. In [19], Fujisaki and Okamoto showed how to convert any PKE scheme secure against chosen plaintext attacks (CPA) to a PKE scheme which is CCA secure in the random oracle model. Surprisingly, the resulting CCA secure scheme has almost the same efficiency as the underlying CPA secure scheme. Moreover, in [21], they proposed more efficient conversion by using the hybrid encryption framework. EPOC (Efficient PrObabilistiC public-key encryption) that is one of the concrete instantiations of [19,21] has been included by IEEE p1363a [1], as it has high practicality, and moreover, its security can be strictly analyzed.

CCA security has been considered as a standard security notion, but it has recently come to light that there are many situations where even CCA security may not guarantee confidentiality of communication. One typical example is situations where secret keys are encrypted in the system. It is known that there is an encryption scheme which is totally insecure when an adversary can get an encryption of secret keys, even though the scheme satisfies CCA security [16].

Black, Rogaway, and Shrimpton [11] introduced a security notion called *key dependent message (KDM) security* which guarantees confidentiality even in the situation of encrypting secret keys. (Around the same time, Camenisch and Lysyanskaya [15] independently formalized a similar notion called circular security.) It is widely known that when an encryption scheme is used as a building block of complicated systems, encrypting secret keys can often occur. Hard disk encryption systems (e.g., BitLocker [11]) and anonymous credential systems [15] are known as such examples. In addition, from the perspective of symbolic cryptography, KDM security is also important [2,3]. From these facts, we consider that KDM security against chosen ciphertext attacks, that is, KDM-CCA security is one of the desirable security notions for practical encryption schemes.

Since CCA security is regarded as a desirable security notion, the security of standardized PKE schemes has been analyzed only in the sense of CCA security. Therefore, it is not clear whether these schemes remain secure even when an adversary can get an encryption of secret keys. In modern society where encryption schemes can be used as a building block of complicated systems, and can encrypt secret keys, it is very important to clarify whether standardized schemes are secure also in the sense of KDM-CCA security.

1.2 Our Results

Based on this motivation, in this paper, we clarify whether the constructions proposed by Fujisaki and Okamoto [19,21] satisfy KDM-CCA security[1] under exactly the same assumptions on the building blocks as those used in [19,21],

[1] When we refer to "KDM security", unless stated otherwise, we mean KDM security with respect to any polynomial time computable functions. For the details, see Remark after Definition 4 in Sect. 2.2.

and show two results.[2] Firstly, we show that the construction of [19] (which we call FO_1) does not satisfy KDM-CCA security generally. Secondly, on the other hand, we show that the construction of [21] (which we call FO_2) satisfies KDM-CCA security. More specifically, we prove the following two theorems.

Theorem 1 (Informal). *Assume that there exists an IND-CPA secure and smooth PKE scheme. Then, there exists an IND-CPA secure and smooth PKE scheme Π such that the PKE scheme FO_1 does not satisfy KDM-CPA security in the random oracle model, where FO_1 is constructed by applying the conversion of [19] to Π.*

Theorem 2 (Informal). *Let Π be a OW-CPA secure and smooth PKE scheme, Σ be a OT-CPA secure symmetric key encryption scheme, and FO_2 be the PKE scheme which is constructed by applying the conversion of [21] to Π and Σ. Then, FO_2 satisfies KDM-CCA security in the random oracle model.*

We note that smoothness is a security notion for PKE schemes introduced by Bellare et al. [9], and essentially equivalent to γ-uniformity which is used in [19,21]. We review the definition of smoothness in Sect. 2.

We think it is theoretically very interesting that the construction of [19] does not necessarily satisfy KDM-CCA security, and on the other hand, that of [21] satisfies KDM-CCA security, even though these two constructions are closely related. In addition, due to Theorem 2, we can construct various practical KDM-CCA secure PKE schemes in the random oracle model, by applying the construction of [21] to existing OW-CPA secure PKE schemes and OT-CPA secure symmetric key encryption (SKE) schemes.

The standardized PKE schemes EPOC-1 and EPOC-2 are respectively instantiated by applying the conversion of [19,21] to the PKE scheme proposed by Okamoto and Uchiyama [27]. We note that the counter-example we show in the proof of Theorem 1 does not capture the PKE scheme of [27]. Therefore, it is not the case that Theorem 1 states that EPOC-1 is insecure in the sense of KDM security. On the other hand, due to Theorem 2, we can immediately see that EPOC-2 is KDM-CCA secure in the random oracle model.

1.3 Related Work

Backes et al. [6] showed that RSA-OAEP is secure in the sense of KDM security in the random oracle model. More specifically, they defined a security notion called *adKDM* security which takes adaptive corruptions and arbitrary active attacks into consideration, and showed that OAEP is adKDM secure in the random oracle model if the underlying trapdoor permutation satisfies partial domain one-wayness. Recently, Davies and Stam [17] studied KDM security of hybrid encryption in the random oracle model. (Since the construction treated

[2] Actually, the construction of [21] is based on that of [20]. In this work, we concentrate on the construction of [21].

in [17] is associated with the construction of FO_2, we later refer to their work in detail in Sect. 5).

Boneh et al. [12] constructed the first KDM secure PKE scheme in the standard model under the decisional Diffie-Hellman (DDH) assumption. Their scheme is KDM secure relative to the family of affine functions (*affine-KDM secure*, for short) which is a comparatively simple function family. Informally, a PKE scheme is said to be KDM secure relative to a function family $\mathcal{F}$ if the scheme remains secure even when an adversary can get an encryption of $f(sk)$, where sk is the secret key and f is an arbitrary function belonging to $\mathcal{F}$. Also, affine-KDM secure schemes were later constructed under the learning with errors (LWE) [5], quadratic residuosity (QR) [13], decisional composite residuosity (DCR) [13,25], and learning parity with noise (LPN) [5] assumptions.

Boneh et al.'s scheme is KDM secure only in the CPA setting, and thus how to construct a KDM-CCA secure scheme remained open. Camenisch et al. [14] later showed how to construct a KDM-CCA secure scheme using a KDM-CPA secure scheme and a non-interactive zero-knowledge (NIZK) proof system for NP languages as building blocks. Recently, Hofheinz [22] showed the first construction of a circular-CCA secure scheme whose security can be directly proved based on number theoretic assumptions.

Applebaum [4] showed how to construct a PKE scheme which is KDM secure relative to functions computable by a-priori bounded polynomial time, based on a PKE scheme which is KDM secure relative to a simple function family called projection functions. We note that the result of Applebaum works in both of the CPA and the CCA settings. Bellare et al. [8] showed a similar result that works only in the CPA setting but is more efficient than Applebaum's. Recently, Kitagawa et al. [23] also showed a more efficient result than Applebaum's, which works in the CCA setting. In addition, Kitagawa et al. [24] showed how to expand the plaintext space of a PKE scheme which is KDM secure relative to projection functions, without using any other assumption.

1.4 Outline of the Paper

In Sect. 2, we review the definitions of the primitives and the security notions that we use in this paper. Then, in Sect. 3, we prove Theorem 1. In the subsequent sections, we tackle Theorem 2. Our idea for proving Theorem 2 is simple, but the proof of Theorem 2 might look somewhat complicated. Thus, after reviewing the construction of FO_2 in Sect. 4, in Sect. 5, we first explain the difficulty which we encounter when trying to prove the KDM security of a hybrid encryption scheme whose key derivation function is regarded as a random oracle. Then, in Sect. 6, we prove Theorem 2.

In order to help the reader understand the proof of Theorem 2, in the full version of this paper, we also show that the hybrid encryption scheme whose key derivation function is a random oracle satisfies KDM-CPA security in the random oracle model, if the underlying PKE scheme and SKE scheme satisfy OW-CPA security and OT-CPA security, respectively. Since the construction can roughly be seen as a simplification of FO_2, we believe the proof is relatively easy to understand than that of Theorem 2.

2 Preliminaries

In this section we define some notations and cryptographic primitives.

2.1 Notations

$x \xleftarrow{r} X$ denotes choosing an element from a finite set X uniformly at random, and $y \leftarrow \mathsf{A}(x; r)$ 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 $y \leftarrow \mathsf{A}(x)$. For strings x and y, $x \| y$ denotes the concatenation of x and y. λ denotes a security parameter. 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 $f(\lambda) = \mathsf{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function. PPT stands for probabilistic polynomial time. $[\ell]$ denotes the set of integers $\{1, \cdots, \ell\}$. $\mathsf{MSB}_n(x)$ denotes the first n bits of x. $\emptyset$ denotes the empty set.

2.2 Public Key Encryption

In this subsection we define public key encryption (PKE).

Definition 1 (Public key encryption). *A PKE scheme Π is a three tuple $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms.*

- *The key generation algorithm KG, given a security parameter 1^λ, outputs a public key pk and a secret key sk.*
- *The encryption algorithm Enc, given a public key pk and a message $m \in \mathcal{M}$, outputs a ciphertext c, where $\mathcal{M}$ is the plaintext space of Π.*
- *The decryption algorithm Dec, given a secret key sk and a ciphertext c, outputs a message $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. This algorithm is deterministic.*

Correctness. *We require $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$ for every $m \in \mathcal{M}$ and $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$.*

Next, we define one-wayness against chosen plaintext attacks (OW-CPA security) for PKE schemes. KDM security, which we define in this subsection, considers situations where there are many users, and thus KDM security is defined via a security game where there are many keys and an adversary can make many challenge queries. Therefore, for our purpose, it is useful to consider the following one-wayness in the multi-user setting. Specifically, we use a security notion which we call List-OW-CPA *security*. In the security game of List-OW-CPA security, there are many keys, and an adversary can make multiple encryption queries and outputs a list of candidate plaintexts in the final phase.

Definition 2 (List-OW-CPA security). *Let Π be a PKE scheme whose message space is $\mathcal{M}$, and ℓ be the number of keys. We define the List-OW-CPA game between a challenger and an adversary $\mathcal{A}$ as follows.*

Initialization. *First, the challenger generates ℓ key pairs $(pk_j, sk_j) \leftarrow \mathsf{KG}(1^\lambda)(j = 1, \cdots, \ell)$. Then, the challenger sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$. Finally, the challenger sets $L_{enc} = \emptyset$.*
$\mathcal{A}$ may make polynomially many encryption queries.

Encryption queries. *$j \in [\ell]$ is an index of a key. The challenger generates $m \xleftarrow{r} \mathcal{M}$ and computes $c \leftarrow \mathsf{Enc}(pk_j, m)$. Then, the challenger adds m to L_{enc} and returns c to $\mathcal{A}$.*

Final phase. *$\mathcal{A}$ outputs L_{ans} which is a set of plaintexts. (We require the size of L_{ans} to be bounded by some polynomial of λ.)*

In this game, we define the advantage of the adversary $\mathcal{A}$ as follows.

$$\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{A},\ell}(\lambda) = \Pr[L_{enc} \cap L_{ans} \neq \emptyset]$$

We say that Π is List-OW-CPA secure if for any PPT adversary $\mathcal{A}$ and polynomial $\ell = \ell(\lambda)$, we have $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{A},\ell}(\lambda) = \mathsf{negl}(\lambda)$.

A OW-CPA secure PKE scheme Π is also List-OW-CPA secure. Formally, the following lemma holds. We provide the definition of OW-CPA security and the proof of Lemma 1 in Appendix A.

Lemma 1. *Let Π be a OW-CPA secure PKE scheme. Then, Π is also List-OW-CPA secure.*

Next, we define KDM-CPA security and KDM-CCA security for PKE schemes.

Definition 3 (KDM-CPA security). *Let Π be a PKE scheme and ℓ be the number of keys. We define the KDM-CPA game between a challenger and an adversary $\mathcal{A}$ as follows. In the following,* **sk** *denotes $(sk_1, \cdots, sk_\ell)$.*

Initialization. *First, the challenger chooses a challenge bit $b \xleftarrow{r} \{0,1\}$. Next, the challenger generates ℓ key pairs $(pk_j, sk_j) \leftarrow \mathsf{KG}(1^\lambda)(j = 1, \cdots, \ell)$ and sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$.*
$\mathcal{A}$ may adaptively make polynomially many KDM queries.

KDM queries. *(j, f), where j is a key index and f is a function. Here, f needs to be efficiently computable. If $b = 1$ then the challenger returns $c \leftarrow \mathsf{Enc}(pk_j, f(\mathbf{sk}))$; If $b = 0$ then the challenger returns $c \leftarrow \mathsf{Enc}(pk_j, 0^{|f(\cdot)|})$.*

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

In this game, we define the advantage of the adversary $\mathcal{A}$ as follows.

$$\mathsf{Adv}^{kdmcpa}_{\Pi,\mathcal{A},\ell}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$$

We say that Π is KDM-CPA secure if for any PPT adversary $\mathcal{A}$ and polynomial $\ell = \ell(\lambda)$, we have $\mathsf{Adv}^{kdmcpa}_{\Pi,\mathcal{A},\ell}(\lambda) = \mathsf{negl}(\lambda)$.

By permitting the adversary to make decryption queries, we can analogously define KDM-CCA security.

Definition 4 (KDM-CCA security). *Let Π be a PKE scheme and ℓ be the number of keys. We define the KDM-CCA game between a challenger and an adversary $\mathcal{A}$ in the same way as the KDM-CPA game except that $\mathcal{A}$ is allowed to adaptively make decryption queries. In the initialization step of the KDM-CCA game, the challenger first runs in the same way as the KDM-CPA game, and then, prepares the KDM query list L_{kdm} into which pairs of the form (j, c) will be stored, where j is an index and c is a ciphertext, and which is initially empty. When $\mathcal{A}$ makes a KDM query (j, f), the challenger computes the answer c and adds (j, c) to L_{kdm}. $\mathcal{A}$ is not allowed to make a decryption query (j, c) which is contained in L_{kdm}.*

Decryption queries. $(j, c) \notin L_{kdm}$, *where j is a key index and c is a ciphertext. For this query, the challenger returns $m \leftarrow \mathsf{Dec}(sk_j, c)$.*

In this game, we define the advantage $\mathsf{Adv}^{kdmcca}_{\Pi,\mathcal{A},\ell}(\lambda)$ of the adversary $\mathcal{A}$ analogously to that in the KDM-CPA game. Then, Π is said to be KDM-CCA secure if for any PPT adversary $\mathcal{A}$ and polynomial $\ell = \ell(\lambda)$, we have $\mathsf{Adv}^{kdmcca}_{\Pi,\mathcal{A},\ell}(\lambda) = \mathsf{negl}(\lambda)$.

Remarks. lack et al. [11] first defined KDM security. In their paper, they made an assumption that functions which the adversary queries in the security game are length-regular. A function f is said to be length-regular if the output length of $f(\mathbf{sk})$ does not depend on the value of $\mathbf{sk}$, and thus we can uniquely determine the length of $f(\mathbf{sk})$ only from f. In this paper, we also impose the length-regularity of functions which the adversary queries in the security game.

In the KDM-CPA game and KDM-CCA game in the random oracle model, the adversary is allowed to make hash queries to the random oracle. Moreover, it is more appropriate to permit a function which the adversary queries as a KDM query (KDM function) to access to the random oracle, in order to capture more various situations. Actually, Black et al. used the definition which allows KDM functions to access to the random oracle. Therefore, similarly to the definition of Black et al., we allow a KDM function to access to the random oracle.

IND-CPA security is a special case of KDM-CPA security. More specifically, we can define IND-CPA security by restricting functions an adversary can query as a KDM query in the KDM-CPA game to any constant functions. Similarly, IND-CCA security is a special case of KDM-CCA security.

Usually, KDM security is defined with respect to a function family $\mathcal{F}$. $\mathcal{F}$-KDM security is defined by restricting KDM functions used by an adversary to functions belonging to $\mathcal{F}$. In this paper, unless stated otherwise, we allow an adversary to query arbitrary function computable in polynomial-time in the security game, and we omit to write a function family.

Next, we review a security notion for PKE schemes called *smoothness* [9]. Informally, a PKE scheme is said to be smooth if the number of possible ciphertexts is super-polynomially large for any message. We note that many known PKE schemes secure in the sense of indistinguishability have smoothness unconditionally, but it is not the case that any IND-CPA or IND-CCA secure PKE

scheme is smooth. However, we can easily transform any non-smooth PKE scheme to a smooth one. Fujisaki and Okamoto [19,21] proved the security of their scheme via a property called γ-uniformity. γ-uniformity is a slightly stronger security notion than smoothness in the sense that it considers maximum also over all public keys, but these two notions are essentially the same.

Definition 5 (Smoothness [9]). Let Π be a PKE scheme. For $\lambda \in \mathbb{N}$, we define **Smth** as follows.

$$\mathbf{Smth}(\lambda) = \mathrm{E}_{(pk,sk)\leftarrow \mathsf{KG}(1^\lambda)}\left[\max_{m,c'} \Pr_{c\leftarrow \mathsf{Enc}(pk,m)}[c = c']\right]$$

We say that Π is smooth if we have $\mathbf{Smth}(\lambda) = \mathsf{negl}(\lambda)$.

We note that Definition 5 is essentially equivalent to the security notion defined via the following game played by a challenger and an adversary $\mathcal{A}$.

Initialization. The challenger generates ℓ key pairs $(pk_j, sk_j) \leftarrow \mathsf{KG}(1^\lambda)(j = 1, \cdots, \ell)$ and sends $((pk_1, sk_1), \cdots, (pk_\ell, sk_\ell))$ to $\mathcal{A}$.
Final phase. $\mathcal{A}$ outputs (j, m, c'), and the challenger computes $c \leftarrow \mathsf{Enc}(pk_j, m)$.

In this game, we define the advantage of the adversary $\mathcal{A}$ as follows.

$$\mathsf{Adv}^{smth}_{\Pi,\mathcal{A},\ell}(\lambda) = \Pr[c = c']$$

Then, it is straightforward to see that for any computationally unbounded adversary $\mathcal{A}$ and polynomial $\ell = \ell(\lambda)$, we have $\mathsf{Adv}^{smth}_{\Pi,\mathcal{A},\ell}(\lambda) \leq \ell \cdot \mathbf{Smth}(\lambda)$. Therefore, if $\mathbf{Smth}(\lambda)$ is negligible, so is $\mathsf{Adv}^{smth}_{\Pi,\mathcal{A},\ell}(\lambda)$ for any computationally unbounded adversary $\mathcal{A}$ and polynomial $\ell = \ell(\lambda)$.

2.3 Symmetric Key Encryption

In this subsection we define symmetric key encryption (SKE).

Definition 6 (Symmetric key encryption). *SKE scheme Σ is a two tuple (E, D) of PPT algorithms.*

- *The encryption algorithm E, given a key $K \in \{0,1\}^\lambda$ and a message $m \in \mathcal{M}$, outputs a ciphertext c, where $\mathcal{M}$ is the plaintext space of Σ.*
- *The decryption algorithm D, given a key K and a ciphertext c, outputs a message $\tilde{m} \in \{\bot\} \cup \mathcal{M}$. This algorithm is deterministic.*

Correctness. *We require $\mathsf{D}(K, \mathsf{E}(K, m)) = m$ for every $m \in \mathcal{M}$ and $K \in \{0,1\}^\lambda$.*

Next, we review the definition of indistinguishability against one-time chosen plaintext attacks (OT-CPA security) for SKE schemes.

$\mathsf{KG}_{\mathsf{FO1}}(1^\lambda)$:	$\mathsf{Enc}_{\mathsf{FO1}}(pk, m)$:	$\mathsf{Dec}_{\mathsf{FO1}}(sk, c)$:
$(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$	$r \leftarrow \{0,1\}^n$	$m\|r \leftarrow \mathsf{Dec}(sk, c)$
return (pk, sk)	$R \leftarrow \mathsf{H}(m\|r)$	if $m\|r = \perp$
	$c \leftarrow \mathsf{Enc}(pk, m\|r; R)$	return $\perp$
	return c	else
		$R \leftarrow \mathsf{H}(m\|r)$
		if $c \neq \mathsf{Enc}(pk, m\|r; R)$
		return $\perp$
		else
		return m

Fig. 1. The construction [19] of an IND-CCA secure PKE scheme $\mathsf{FO}_1 = (\mathsf{KG}_{\mathsf{FO1}}, \mathsf{Enc}_{\mathsf{FO1}}, \mathsf{Dec}_{\mathsf{FO1}})$ from a PKE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ which is IND-CPA secure and smooth, and a hash function H.

Definition 7 (OT-CPA security). *Let Σ be a SKE scheme whose message space is $\mathcal{M}$. We define the OT-CPA game between a challenger and an adversary $\mathcal{A}$ as follows.*

Initialization. *First the challenger chooses a challenge bit $b \xleftarrow{r} \{0,1\}$. Next the challenger generates a key $K \xleftarrow{r} \{0,1\}^\lambda$ and sends 1^λ to $\mathcal{A}$.*
Challenge. *$\mathcal{A}$ selects two messages m_0 and m_1 of equal length, and sends them to the challenger. Then the challenger returns $c \leftarrow \mathsf{E}(K, m_b)$.*
Final phase. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as follows.

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$$

We say that Σ is OT-CPA secure if for any PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

3 Fujisaki-Okamoto Construction (PKC'99) Does Not Satisfy KDM Security in General

Fujisaki and Okamoto [19] showed how to transform any IND-CPA secure (and smooth) PKE scheme to an IND-CCA secure one by using a random oracle. The resulting scheme has almost the same efficiency as the underlying scheme. In this section, although their construction satisfies IND-CCA security, we show that their construction generally does not satisfy KDM security. In the following, we first review the construction of [19], and then we show our negative result.

Let $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme, and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ be a hash function, where $n = n(\lambda)$ is a polynomial. Then, we construct a PKE scheme $\mathsf{FO}_1 = (\mathsf{KG}_{\mathsf{FO1}}, \mathsf{Enc}_{\mathsf{FO1}}, \mathsf{Dec}_{\mathsf{FO1}})$ as described in Fig. 1. Here, we assume

$\mathsf{KG}(1^\lambda)$:	$\mathsf{Enc}(pk, m)$:	$\mathsf{Dec}(sk, p\|\|c)$:
$sk \xleftarrow{r} \{0,1\}^s$	$c \leftarrow \widehat{\mathsf{Enc}}(pk, m)$	$(\widehat{pk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KG}}(1^\lambda; sk)$
$(\widehat{pk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KG}}(1^\lambda; sk)$	$(pk', sk') \leftarrow \widehat{\mathsf{KG}}(1^\lambda; \mathsf{MSB}_s(m))$	$m \leftarrow \widehat{\mathsf{Dec}}(\widehat{sk}, c)$
$pk \leftarrow \widehat{pk}$	if $pk = pk'$	return m
return (pk, sk)	return $1\|\|c$	
	else	
	return $0\|\|c$	

Fig. 2. The construction of a PKE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ which is IND-CPA secure and smooth but not KDM-CPA secure from an IND-CPA secure and smooth PKE scheme $\widehat{\Pi} = (\widehat{\mathsf{KG}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Dec}})$.

that the plaintext space of Π is $\{0,1\}^*$, and thus that of FO_1 is also $\{0,1\}^*$. In addition, let the randomness spaces of Enc and $\mathsf{Enc}_{\mathsf{FO1}}$ be both $\{0,1\}^n$.

In the above construction, Fujisaki and Okamoto showed that if Π is IND-CPA secure and smooth, and H is a random oracle, then FO_1 is IND-CCA secure in the random oracle model. However, as mentioned above, we show that FO_1 does not satisfy KDM-CPA security generally under the same assumptions. Formally, we show the following theorem.

Theorem 3. *Assume that there exists an IND-CPA secure and smooth PKE scheme. Then, there exists an IND-CPA secure and smooth PKE scheme Π such that FO_1 does not satisfy KDM-CPA security in the random oracle model.*

Proof of Theorem 3. This proof consists of two steps. In the first step, using any IND-CPA secure and smooth PKE scheme, we construct a PKE scheme which is still IND-CPA secure and smooth, but insecure in the sense of KDM security. Then, in the second step, we show that the PKE scheme which is constructed by applying the conversion of [19] to the PKE scheme we construct in the first step, also does not satisfy KDM-CPA security. In the following, we start with the first step.

Let $\widehat{\Pi} = (\widehat{\mathsf{KG}}, \widehat{\mathsf{Enc}}, \widehat{\mathsf{Dec}})$ be any IND-CPA secure and smooth PKE scheme. Without loss of generality, we assume that the plaintext space of $\widehat{\Pi}$ is $\{0,1\}^*$, and the randomness space of $\widehat{\mathsf{KG}}$ is $\{0,1\}^s$ for some polynomial $s = s(\lambda)$. Then, using $\widehat{\Pi}$, we construct a PKE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ as described in Fig. 2.

It is clear that if $\widehat{\Pi}$ is IND-CPA secure and smooth, then Π satisfies the same security notions. The reason is as follows. In the IND-CPA game regarding Π, since a PPT adversary can find the randomness that was used to run $\widehat{\mathsf{KG}}$ with negligible probability, when the challenger generates the challenge ciphertext, Enc outputs a ciphertext whose first bit is 1 with negligible probability. Thus, if $\widehat{\Pi}$ satisfies IND-CPA, then so is Π. Moreover, regardless of the plaintext, a ciphertext output by Enc includes a ciphertext output by $\widehat{\mathsf{Enc}}$ itself, and thus Π is smooth if so is $\widehat{\Pi}$. On the other hand, Π does not satisfy KDM-CPA security. In order to show it, we consider the following adversary $\mathcal{A}$ which attacks the KDM-CPA security of Π. For simplicity, we consider the case where only one

key pair exists. On input pk, $\mathcal{A}$ queries the identity function id, gets the answer $p\|c$, and outputs $b' = p$. Let b be the challenge bit in the KDM-CPA game between the challenger and $\mathcal{A}$. Then, we can estimate the advantage of $\mathcal{A}$ as follows.

$$\begin{aligned}\mathsf{Adv}^{kdmcpa}_{\Pi,\mathcal{A},1}(\lambda) &= \frac{1}{2}|\Pr[b'=1|b=1] - \Pr[b'=1|b=0]| \\ &= \frac{1}{2}|\Pr[p=1|b=1] - \Pr[p=1|b=0]|\end{aligned}$$

Here, let sk be the secret key corresponding to pk. Then, sk is chosen from $\{0,1\}^s$ at random and $pk = \widehat{pk}$, where $(\widehat{pk}, \widehat{sk}) = \widehat{\mathsf{KG}}(1^\lambda; sk)$. In addition, let $m_1 = sk$ and $m_0 = 0^s$. Then, we note that for any $b \in \{0,1\}$, the probability that p equals 1 is the same as the probability that $pk = pk'$ holds, where $(pk', sk') = \widehat{\mathsf{KG}}(1^\lambda; m_b)$. We note that these probabilities are taken over the choice of sk. When $b = 1$, it is straightforward that $pk = pk'$ always holds, and thus we have $\Pr[p = 1|b = 1] = 1$. On the other hand, when $b = 0$, $pk = pk'$ occurs only with negligible probability. The reason is as follows. If $pk = pk'$ holds, by the correctness of $\widehat{\Pi}$, $\widehat{\mathsf{Dec}}(sk', \widehat{\mathsf{Enc}}(pk, m)) = m$ holds for any $m \in \{0,1\}^*$. Therefore, if $pk = pk'$ holds with non-negligible probability, an adversary can break the IND-CPA security of $\widehat{\Pi}$ by generating $(pk', sk') \leftarrow \widehat{\mathsf{KG}}(1^\lambda; 0^s)$ and decrypting the challenge ciphertext using sk'. This is a contradiction, and thus we have $\Pr[p = 1|b = 0] = \mathsf{negl}(\lambda)$. From these, we have $\mathsf{Adv}^{kdmcpa}_{\Pi,\mathcal{A},1}(\lambda) = \frac{1}{2}(1 - \mathsf{negl}(\lambda))$, and we see that Π does not satisfy KDM-CPA security.

Next, we construct a PKE scheme $\mathsf{FO}_1 = (\mathsf{KG}_{\mathsf{FO1}}, \mathsf{Enc}_{\mathsf{FO1}}, \mathsf{Dec}_{\mathsf{FO1}})$ by applying the conversion in Fig. 1 to the above Π, and show that FO_1 also does not satisfy KDM-CPA security. Let (pk, sk) be a key pair output by $\mathsf{KG}_{\mathsf{FO1}}$. Here, a key pair of FO_1 is a key pair of Π itself. Namely, sk is randomly chosen from $\{0,1\}^s$ and $pk = \widehat{pk}$, where $(\widehat{pk}, \widehat{sk}) \leftarrow \widehat{\mathsf{KG}}(1^\lambda; sk)$. Then, for any $m \in \{0,1\}^s$, the probability that the first bit of the result of $\mathsf{Enc}_{\mathsf{FO1}}(pk, m)$ equals 1 is the same as the probability that $pk = pk'$ holds, where $(pk', sk') = \widehat{\mathsf{KG}}(1^\lambda; \mathsf{MSB}_s(m\|r)) = \widehat{\mathsf{KG}}(1^\lambda; m)$ and r is a randomness generated in $\mathsf{Enc}_{\mathsf{FO1}}$. These probabilities are over the choice of the random oracle H, (pk, sk), and $r \in \{0,1\}^n$. Therefore, if $m = sk$, the first bit of $\mathsf{Enc}_{\mathsf{FO1}}(pk, m; r)$ always equals 1. One the other hand, if m does not depend on sk, similarly to the first step, the first bit of $\mathsf{Enc}_{\mathsf{FO1}}(pk, m; r)$ equals 1 only with negligible probability. From these, FO_1 does not satisfy KDM-CPA security. □ (**Theorem** 3)

4 KDM-CCA Security of Fujisaki-Okamoto Construction (J. Cryptology'13)

Fujisaki and Okamoto [21] showed how to construct an IND-CCA secure PKE scheme in the random oracle model (which we call FO_2) using a OW-CPA secure PKE scheme and a OT-CPA secure SKE scheme. In this section, we show that

$\mathsf{KG}_{\mathsf{FO2}}(1^\lambda)$:	$\mathsf{Enc}_{\mathsf{FO2}}(pk, m)$:	$\mathsf{Dec}_{\mathsf{FO2}}(sk, (c, d))$:
$(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$	$r \xleftarrow{r} \{0,1\}^\lambda$	$r \leftarrow \mathsf{Dec}(sk, c)$
return (pk, sk)	$K \xleftarrow{r} \mathsf{G}(r)$	if $r = \bot$ or $c \neq \mathsf{Enc}(pk, r; \mathsf{H}(r, d))$
	$d \leftarrow \mathsf{E}(K, m)$	return $\bot$
	$R \leftarrow \mathsf{H}(r, d)$	else
	$c \leftarrow \mathsf{Enc}(pk, r; R)$	$K \leftarrow \mathsf{G}(r)$
	return (c, d)	$m \leftarrow \mathsf{D}(K, d)$
		return m

Fig. 3. The construction [21] of a PKE scheme $\mathsf{FO}_2 = (\mathsf{KG}_{\mathsf{FO2}}, \mathsf{Enc}_{\mathsf{FO2}}, \mathsf{Dec}_{\mathsf{FO2}})$ from a PKE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a SKE scheme $\Sigma = (\mathsf{E}, \mathsf{D})$.

FO_2 also satisfies KDM-CCA security in the random oracle model, under exactly the same assumptions on the building blocks as those used in [21]. First, we review the construction of FO_2.

Let $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme and $\Sigma = (\mathsf{E}, \mathsf{D})$ be a SKE scheme. Here, we assume that the message space and the randomness space of Π are $\{0,1\}^\lambda$ and $\{0,1\}^n$, respectively, where $n = n(\lambda)$ is a polynomial. Moreover, we also assume that the message space and the key space of Σ are $\{0,1\}^*$ and $\{0,1\}^\lambda$, respectively. In addition, let $\mathsf{H} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^n$ and $\mathsf{G} : \{0,1\}^* \to \{0,1\}^\lambda$ be hash functions. Then, we construct a PKE scheme $\mathsf{FO}_2 = (\mathsf{KG}_{\mathsf{FO2}}, \mathsf{Enc}_{\mathsf{FO2}}, \mathsf{Dec}_{\mathsf{FO2}})$ as described in Fig. 3. Here, we note that the message space of FO_2 is $\{0,1\}^*$.

[21] showed that, by regarding H and G as random oracles, if Π is OW-CPA secure and smooth, and Σ is OT-CPA secure, then FO_2 satisfies IND-CCA security in the random oracle model. As mentioned earlier, we show that FO_2 satisfies KDM-CCA security, even though we require exactly the same assumptions for building blocks as those used in [21]. Formally, we show the following theorem.

Theorem 4. *Let Π be a PKE scheme which is OW-CPA secure and smooth, Σ be a OT-CPA secure SKE scheme, and H and G be random oracles. Then, FO_2 is a PKE scheme which is KDM-CCA secure in the random oracle model.*

5 Overview of Our Techniques

Our idea for proving the KDM-CCA security of FO_2 is conceptually simple. However, unfortunately, our security proof might look somewhat complicated. Thus, in this section, we first explain where the difficulty lies and how we overcome it when showing the KDM-CCA security of FO_2.

The difficulty. FO_2 has a somewhat complicated structure at first glance. However, it can roughly be seen as a hybrid encryption scheme Π_{hyb} which has the following encryption algorithm $\mathsf{Enc}_{\mathsf{hyb}}$.

$$\mathsf{Enc}_{\mathsf{hyb}}(pk, m; r) = (\mathsf{Enc}(pk, r), \mathsf{E}(\mathsf{G}(r), m))$$

Here, similarly to FO_2, $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and $\Sigma = (\mathsf{E}, \mathsf{D})$ are a OW-CPA secure PKE scheme and a OT-CPA secure SKE scheme, respectively, and G is

	Game $[b = 1]$	Game [hybrid]	Game $[b = 0]$
1st	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_1^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_1(\cdot)\rvert})$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_1(\cdot)\rvert})$
2nd	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_2^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_2^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_2(\cdot)\rvert})$

Fig. 4. The ordinary sequence of games. "1st" and "2nd" indicate the answers to the first and second KDM queries from $\mathcal{A}$, respectively.

a random oracle. The difficulty that lies in the security proof of the KDM-CCA security of FO_2, is almost the same as that of the KDM-CPA security of Π_{hyb}. Therefore, for simplicity, we explain the difficulty we encounter when showing the KDM-CPA security of Π_{hyb} in the case where only one key pair (pk, sk) exists. In the following, we call a function an adversary queries as a KDM query in the security game a KDM function. In addition, we call an answer to a KDM query a challenge ciphertext, and randomness r encapsulated by Enc a proto-key.

We first consider a simple case, that is, the case where KDM functions cannot access to the random oracle G. In this case, an adversary who does not query a proto-key r to G cannot distinguish $(\mathsf{Enc}(pk, r), \mathsf{E}(\mathsf{G}(r), f(sk)))$ and $(\mathsf{Enc}(pk, r), \mathsf{E}(\mathsf{G}(r), 0^{|f(\cdot)|}))$ due to the randomness of outputs of G and the OT-CPA security of Σ, where f is a KDM function. Thus, all we have to consider is whether the adversary can query the proto-key r to G, but it is unlikely because of the OW-CPA security of Π. From these, in this case, we can easily see that Π_{hyb} is KDM-CPA secure.

However, in the case where KDM functions can access to the random oracle G, there is a problem. The problem is that an adversary who makes multiple KDM queries can get an encryption of a proto-key r which was used to compute a past challenge ciphertext. In order to take a closer look at this problem, we consider the reduction from the KDM-CPA security of Π_{hyb} to the OT-CPA security of Σ. For simplicity, we consider an adversary $\mathcal{A}$ who makes only two KDM queries in the KDM-CPA game of Π_{hyb}. Let f_1 and f_2 be the KDM functions that $\mathcal{A}$ sends, and b denote the challenge bit between the challenger and $\mathcal{A}$. Then, consider the sequence of games as described in Fig. 4.

Game $[b = 1]$ and Game $[b = 0]$ correspond to the KDM-CPA game when $b = 1$ and $b = 0$, respectively. If the behavior of $\mathcal{A}$ does not change non-negligibly between Game $[b = 1]$ and Game $[b = 0]$, then we can conclude that Π_{hyb} is KDM-CPA secure. In order to show this by using the OT-CPA security of Σ, we typically consider a hybrid game Game [hybrid], and we show that the behavior of $\mathcal{A}$ does not change between Game$[b = 1]$ and Game [hybrid], and between Game [hybrid] and Game $[b = 0]$.

Then, we try to construct an adversary $\mathcal{B}$ who simulates Game $[b = 1]$ or Game [hybrid] for $\mathcal{A}$ according to the value of the challenge bit between $\mathcal{B}$ and the challenger of the OT-CPA game regarding Σ. $\mathcal{B}$ first generates (pk, sk) using KG and sends pk to $\mathcal{A}$. $\mathcal{B}$ simulates G by lazy sampling. For the first KDM query f_1 from $\mathcal{A}$, $\mathcal{B}$ first makes the challenge query $(f_1^{\mathsf{G}}(sk), 0^{|f_1(\cdot)|})$ and gets the answer d_1. Then, $\mathcal{B}$ generates a proto-key r_1, computes $c_1 \leftarrow \mathsf{Enc}(pk, r_1)$, and returns (c_1, d_1) to $\mathcal{A}$. In addition, $\mathcal{B}$ let the value of $\mathsf{G}(r_1)$ be the value of the key of

Σ that the challenger used to compute d_1. We note that $\mathcal{B}$ does not know the actual value of the key of Σ. Here, suppose that $\mathcal{A}$ sends the following KDM function f_2 as the second KDM query. $f_2^{\mathsf{G}}(sk)$ computes $r_1 = \mathsf{Dec}(sk, c_1)$, and then computes $\mathsf{G}(r_1)$ and returns the value. Then, in order to compute the value of $f_2^{\mathsf{G}}(r_1)$, $\mathcal{B}$ needs the value of $\mathsf{G}(r_1)$. However, $\mathcal{B}$ does not know the actual value of $\mathsf{G}(r_1)$, and thus cannot compute $f_2^{\mathsf{G}}(sk)$ correctly. Therefore, $\mathcal{B}$ fails a simulation of Games for $\mathcal{A}$ if $\mathcal{A}$ queries such a second KDM query.

The approach of Davies and Stam [17]. Davies and Stam [17] studied KDM security for hybrid encryption where the key derivation function (KDF) is regarded as a random oracle, and pointed out the above problem.[3] Then, they overcame the problem and showed that if a PKE scheme satisfies OW-CCA security and a SKE scheme satisfies OT-CCA security, the hybrid encryption scheme satisfies KDM-CCA security in the random oracle model.

They approached the above problem by introducing a new security notion for SKE schemes that they call *prior key dependent message security (*PKDM security*)*. Informally, PKDM security guarantees that an encryption scheme can securely encrypt a message which depends only on keys of the scheme used to generate past ciphertexts. In other words, confidentiality of a ciphertext of a PKDM secure scheme under a key K_i holds even if an adversary can get an encryption of the form $\mathsf{E}(K_i, f(K_1, \cdots, K_{i-1}))$, where $K_1, \cdots, K_{i-1}$ are keys used so far and f is an arbitrary function. Davies and Stam showed that PKDM-CCA security is equivalent to OT-CCA security, and they overcame the above problem by reducing the KDM-CCA security of the hybrid encryption scheme to the PKDM-CCA security of the SKE scheme.

To accomplish this task, their reduction algorithm has to convert a KDM function of the secret keys of the PKE scheme to that of the keys of the SKE scheme. Here, in the KDM-CCA game which the reduction algorithm simulates, there exists a random oracle. On the other hand, in the PKDM-CCA game which the reduction algorithm actually plays, there does not exist a random oracle. Therefore, Davies and Stam used the technique of replacing the random oracle with a pseudorandom functions (PRF) when conducting the above conversion of KDM functions. Therefore, their security bound has a PRF term even though the construction does not include a PRF. In addition, they stated that it is difficult to prove its KDM-CCA security without using PKDM security or a PRF.

Our approach. Both our work and the work of [17] study the KDM-CCA security of the hybrid encryption scheme whose KDF is regarded as a random oracle. However, there is a big difference between our work and [17]. The difference is that the building blocks of [17] already satisfy CCA security. On the other hand, the building blocks of FO_2 that we treat satisfy only CPA security. In order to prove the KDM-CCA security of FO_2 even though the building blocks satisfy only CPA security, similarly to Fujisaki and Okamoto [21], we have to use smoothness [9]. (As mentioned in Sect. 2, smoothness is essentially the same notion as

[3] Davies and Stam actually treated a hybrid encryption scheme constructed from a SKE scheme and a key encapsulation mechanism (KEM).

	Game $[b = 1]$	Game [reverse]	Game $[b = 0]$
1st	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_1^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_1^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_1(\cdot)\rvert})$
2nd	$\mathsf{Enc}_{\mathsf{hyb}}(pk, f_2^{\mathsf{G}}(sk))$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_2(\cdot)\rvert})$	$\mathsf{Enc}_{\mathsf{hyb}}(pk, 0^{\lvert f_2(\cdot)\rvert})$

Fig. 5. The sequence of games which replace the challenge ciphertexts in the "reverse order".

γ-uniformity.) In addition, the construction of FO_2 contains two random oracles, and one of them is used to generate a randomness for the encryption algorithm of the PKE scheme. Thus, it looks difficult to replace both of two random oracles with a PRF, and thus, to directly use the proof technique used in [17]. Therefore, we try to prove the KDM-CCA security of FO_2 by a proof technique which is different from that of [17], especially a technique without using PKDM security or a PRF. In the following, we give our main idea using Π_{hyb}.

As earlier, we consider an adversary $\mathcal{A}$ for the KDM-CPA security of Π_{hyb} who makes a KDM query only twice. Our idea is to replace the challenge ciphertexts in "reverse order". Namely, we consider the sequence of games as described in Fig. 5.

Then, we can avoid the problem that we explained above. We try to construct an adversary $\mathcal{B}$ who simulates Game $[b = 1]$ or Game [reverse] for $\mathcal{A}$ according to the value of the challenge bit between $\mathcal{B}$ and the challenger of the OT-CPA game regarding Σ. $\mathcal{B}$ first generates (pk, sk) using KG and sends pk to $\mathcal{A}$. For the first KDM query f_1 from $\mathcal{A}$, $\mathcal{B}$ generates a proto-key r_1 and a key K_1 of Σ, and returns $(\mathsf{Enc}(pk, r_1), \mathsf{E}(K_1, f_1^{\mathsf{G}}(sk)))$. In addition, $\mathcal{B}$ defines the value of $\mathsf{G}(r_1)$ as K_1 by itself. For the second KDM query f_2 from $\mathcal{A}$, $\mathcal{B}$ makes the challenge query $(f_2^{\mathsf{G}}(sk), 0^{|f_2(\cdot)|})$ and gets the answer d_2. Then, $\mathcal{B}$ generates a proto-key r_2, computes $c_2 \leftarrow \mathsf{Enc}(pk, r_2)$, and returns (c_2, d_2) to $\mathcal{A}$. Since $\mathcal{B}$ defines the value of $\mathsf{G}(r_1)$ by itself, $\mathcal{B}$ can simulate G for f_2 and compute $f_2^{\mathsf{G}}(sk)$ correctly even if f_2 calls G.

Then, we in turn try to construct an adversary $\mathcal{B}'$ who simulates Game [reverse] or Game $[b = 0]$ for $\mathcal{A}$ according to the value of the challenge bit between $\mathcal{B}'$ and the challenger of the OT-CPA game regarding Σ. $\mathcal{B}'$ also generates (pk, sk) using KG and sends pk to $\mathcal{A}$. For the first KDM query f_1 from $\mathcal{A}$, $\mathcal{B}'$ first makes the challenge query $(f_1^{\mathsf{G}}(sk), 0^{|f_1(\cdot)|})$ and gets the answer d_1. Then, $\mathcal{B}'$ generates a proto-key r_1, computes $c_1 \leftarrow \mathsf{Enc}(pk, r_1)$, and returns (c_1, d_1) to $\mathcal{A}$. Here, $\mathcal{B}'$ let the value of $\mathsf{G}(r_1)$ be the value of the key of Σ that the challenger used to compute d_1, and thus $\mathcal{B}$ does not know the value of $\mathsf{G}(r_1)$. However, in this case, $\mathcal{B}'$ does not need the value of $\mathsf{G}(r_1)$ to respond to the second KDM query because the answer to this query is an encryption of $0^{|f_2(\cdot)|}$, and thus $\mathcal{B}'$ does not have to compute $f_2^{\mathsf{G}}(sk)$ actually. We note that since KDM functions are length regular, $\mathcal{B}'$ can know $|f_2(\cdot)|$ without computing $f_2^{\mathsf{G}}(sk)$. Therefore, we can overcome the problem that KDM functions may refer to past proto-keys by replacing the challenge ciphertexts in the reverse order.

When we prove the KDM-CCA security of FO_2, we have to take the OW-CPA security and smoothness of the building block PKE scheme into consideration,

and then we use the identical-until-bad technique and the deferred analysis technique. Hence, in this case, whether a KDM function is computed actually or not is very sensitive, but by dividing the bad events into smaller pieces than Davies and Stam, we are able to complete the proof.

6 Proof of Theorem 4

In this section, we show the formal proof of Theorem 4.

Let $\mathcal{A}$ be an adversary that attacks the KDM-CCA security of FO_2 in the random oracle model, and makes at most q_e KDM queries and q_d decryption queries, where q_e and q_d are polynomials of λ. Let ℓ be a polynomial of λ and denote the number of keys. As mentioned just after Definition 4, similarly to Black et al. [11], we assume that a function which $\mathcal{A}$ queries as a KDM query can access to the random oracles, and is length-regular. We note that since KDM functions can access to the random oracle, it makes security proof simple to clearly distinguish the entries of the hash list used to compute KDM functions and that used to make challenge ciphertexts. Thus, we divide the random oracle into multiple random oracles, which is a technique used by Davies and Stam [17]. Namely, in our sequence of games, there are six random oracles even though the construction of FO_2 contains only two random oracles. Now, consider the following sequence of games.

Game 0. This is the KDM-CCA game in the random oracle model regarding FO_2. See Fig. 6 for how KDM queries and decryption queries are answered, and how random oracles behave in Game 0. In the original KDM-CCA game regarding FO_2, there exist only two random oracles H and G. However, to define the subsequent games, we consider six random oracles H, H^*, HH^*, G, G^*, and GG^*. Moreover, random oracles are implemented by lazy sampling. More specifically, random oracles run as follows.

- H maintains the list L_{H} which stores query/answer pairs so far, and runs as follows. If some value (r, d) is queried to H, H first checks whether there is an entry of the form $((r, d), R)$ in L_{H}. If so, H returns R. Otherwise, H returns a fresh random value R and adds $((r, d), R)$ to L_{H}.
- H^*, G, and G^* also maintain the query/answer pairs list L_{H^*}, L_{G}, and L_{G^*}, respectively, and run in the same way as H.
- Similarly to the other random oracles, HH^* and GG^* are implemented by lazy sampling. However, HH^* and GG^* do not have their own list. When HH^* samples a fresh random value, HH^* adds a new entry to L_{H}, and when GG^* samples a fresh random value, GG^* adds a new entry to L_{G}. In addition, HH^* runs by referring to both lists L_{H} and L_{H^*}, and GG^* runs by referring to both lists L_{G} and L_{G^*}.

Moreover, in this game, H and H^* are synchronized. Namely, H and H^* refer to not only their own list but also the list of the other one. Similarly, G and G^* are synchronized. In this game, random oracles are called at the following four cases.

[KDM] (j, f)	$\mathsf{H}(r, d)$:	$\mathsf{H}^*(r, d)$:
$m_1 \leftarrow f^{\mathsf{HH}^*, \mathsf{GG}^*}(\mathbf{sk})$	if $((r, d), R) \in L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$	if $((r, d), R) \in L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$
$m_0 \leftarrow 0^{\lvert f(\cdot) \rvert}$	return R	return R
$r \xleftarrow{r} \{0,1\}^\lambda$, $K \leftarrow \mathsf{G}^*(r)$	else	else
$d \leftarrow E(K, m_b)$	$R \xleftarrow{r} \{0,1\}^n$	$R \xleftarrow{r} \{0,1\}^n$
$R \leftarrow \mathsf{H}^*(r, d)$	add $((r, d), R)$ to L_{H}	add $((r, d), R)$ to L_{H^*}
$c \leftarrow \mathsf{Enc}(pk_j, r; R)$	return R	return R
add (j, c, d) to L_{kdm}	$\mathsf{G}(r)$:	$\mathsf{G}^*(r)$:
return (c, d)	if $(r, K) \in L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$	if $(r, K) \in L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$
[Decryption] $(j, c, d) \notin L_{kdm}$	return K	return K
$r \leftarrow \mathsf{Dec}(sk_j, c)$	else	else
if $r = \perp$ or	$K \xleftarrow{r} \{0,1\}^\lambda$	$K \xleftarrow{r} \{0,1\}^\lambda$
$c \neq \mathsf{Enc}(pk_j, r; \mathsf{HH}^*(r, d))$	add (r, K) to L_{G}	add (r, K) to L_{G^*}
return $\perp$	return K	return K
else		
$K \leftarrow \mathsf{GG}^*(r)$		
return $m \leftarrow \mathsf{D}(K, c)$		

Fig. 6. The manner the challenger responds to a KDM query and a decryption query, and the behavior of H, H^*, G, and G^* in Game 0. We note that in Game 0, HH^* and GG^* run in exactly the same way as H and G, respectively.

- **(1)** When $\mathcal{A}$ makes a hash query.
- **(2)** When the challenger computes a hash value to respond to a KDM query from $\mathcal{A}$.
- **(3)** When a function which $\mathcal{A}$ sends to the challenger as a KDM query accesses to the random oracles.
- **(4)** When the challenger computes a hash value to respond to a decryption query from $\mathcal{A}$.

H and G are used when **(1)**, H^* and G^* are used when **(2)**, and HH^* and GG^* are used when **(3)** and **(4)**. Then, we note that the difference between this game and the original KDM-CCA game is only conceptual.

Game 1. Same as Game 0, except for the behaviors of H^* and G^*. In this game, H^* runs without referring to L_{H}. Moreover, every time H^* is given an input $(r, d) \in \{0,1\}^\lambda \times \{0,1\}^*$, H^* generates a uniformly random value R over $\{0,1\}^n$. Then, H^* outputs R after adding $((r, d), R)$ to L_{H^*} even if there already exists an entry whose first component is (r, d) in L_{H^*}. We note that H and HH^* still refer to L_{H^*} in this game. When H and HH^* refer to L_{H^*}, if there are multiple entries whose first components are identical, then H and HH^* adopt the entry which was added first. G, G^*, and GG^* run analogously to H, H^*, and HH^*, respectively. See Fig. 7 for how H^* and G^* behave in Game 1.

Game 2. Same as Game 1, except that H runs without referring to L_{H^*}, and G runs without referring to L_{G^*}. Here, we note that HH^* still refers to both L_{H}

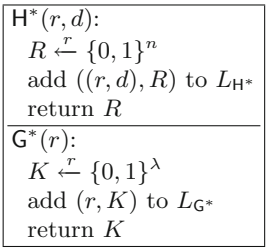


Fig. 7. The behavior of H^* and G^* in Game 1.

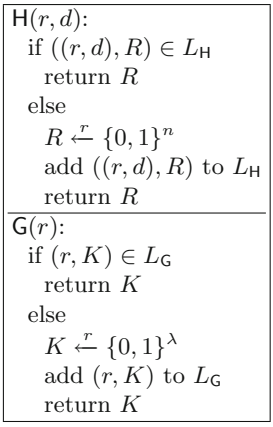


Fig. 8. The behavior of H and G in Game 2.

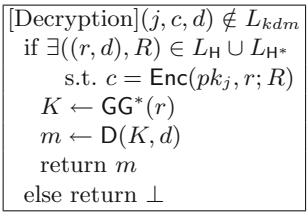


Fig. 9. The manner the challenger responds to a decryption query in Game 3.

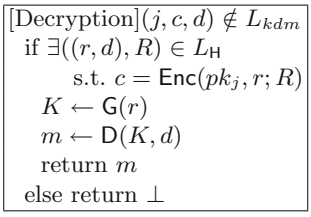


Fig. 10. The manner the challenger responds to a decryption query in Game 4.

and L_{H^*}, and GG^* also refers to both L_{G} and L_{G^*}. See Fig. 8 for how H and G behave in Game 2.

Game 3. Same as Game 2, except that if $\mathcal{A}$ makes a decryption query, then the challenger responds as described in Fig. 9. We note that, due to this change, the challenger can respond to a decryption query without using the secret keys in this and subsequent games.

Game 4. Same as Game 3, except that if $\mathcal{A}$ makes a decryption query, the challenger refers to only L_{H} instead of $L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$ when checking the validity of the ciphertext from $\mathcal{A}$, and uses G instead of GG^* to compute the answer. See Fig. 10.

Game 5. Same as Game 4, except that if $\mathcal{A}$ makes a KDM query (j,f), the challenger always returns a ciphertext whose plaintext is $0^{|f(\cdot)|}$. See Fig. 11.

The above completes the description of the games.

We define the following events in Game i $(i = 0, \cdots, 5)$.

$\mathtt{SUC}_i$**:** $\mathcal{A}$ succeeds in guessing the challenge bit, that is, $b = b'$ occurs.

$[\mathrm{KDM}](j, f)$
$r \xleftarrow{r} \{0,1\}^\lambda$
$K \leftarrow \mathsf{G}^*(r)$
$d \leftarrow \mathsf{E}(K, 0^{|f(\cdot)|})$
$R \leftarrow \mathsf{H}^*(r, d)$
$c \leftarrow \mathsf{Enc}(pk_j, r; R)$
add (j, c, d) to L_{kdm}
return (c, d)

Fig. 11. The manner the challenger responds to a KDM query in Game 5.

$\mathtt{COL}_i$**:** When the challenger generates $r \leftarrow \{0,1\}^\lambda$ to respond to a KDM query from $\mathcal{A}$, there exists an entry of the form $(r, \cdot)$ in $L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$, or there exists an entry of the form $((r, \cdot), \cdot)$ in L_{H}.

$\mathtt{BHQ}_i$**:** When $\mathcal{A}$ queries r to G or queries (r, d) to H, there exists an entry of the form $(r, \cdot)$ in L_{G^*}. We call such a hash query a "**bad hash query**".

In addition, we define the following two events related to decryption queries.

$\mathtt{SMTH}_i$**:** $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ which satisfies the following two conditions, where $\mathsf{Dec}(sk_j, c) = r$: There does not exist an entry of the form $((r, d), \cdot)$ in $L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$, and $c = \mathsf{Enc}(pk, r; \mathsf{HH}^*(r, d))$ holds.

$\mathtt{BDQ}_i$**:** $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ which satisfies the following condition: There exists an entry $((r, d), R) \in L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$, and for such r, $(r, \cdot) \in L_{\mathsf{G}^*}$ holds. Here, $(r, \cdot) \in L_{\mathsf{G}^*}$ indicates that there exists an entry in L_{G^*} whose first component is r. We call such a decryption query a "**bad decryption query**".

Using the above events, we can estimate $\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda)$ as Lemma 2 stated below.

Lemma 2. *We can estimate* $\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda)$ *as follows:*

$$\begin{aligned}\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda) \leq\ & \Pr[\mathtt{COL}_1] + 2\Pr[\mathtt{SMTH}_3] + |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| \\ & + |\Pr[\mathtt{BHQ}_4] - \Pr[\mathtt{BHQ}_5]| + 2|\Pr[\mathtt{BDQ}_4] - \Pr[\mathtt{BDQ}_5]| \\ & + \Pr[\mathtt{BHQ}_5] + 2\Pr[\mathtt{BDQ}_5] \end{aligned} \tag{1}$$

Proof of Lemma 2. As mentioned above, the difference between Game 0 and the original KDM-CCA game is only conceptual, and thus we have $\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda) = |\Pr[\mathtt{SUC}_0] - \frac{1}{2}|$. By using the triangle inequality, we get the following inequality.

$$|\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \leq \sum_{k=0}^{4} |\Pr[\mathtt{SUC}_k] - \Pr[\mathtt{SUC}_{k+1}]| + |\Pr[\mathtt{SUC}_5] - \frac{1}{2}|$$

We note that, in Game 5, the challenger always responds to a KDM query (j, f) from $\mathcal{A}$ by returning an encryption of $0^{|f(\cdot)|}$ regardless of the value of the challenge bit. Therefore, in Game 5, the choice of the challenge bit and the behavior of $\mathcal{A}$ are independent, and thus $|\Pr[\mathtt{SUC}_5] - \frac{1}{2}| = 0$. Below, we estimate $|\Pr[\mathtt{SUC}_k] - \Pr[\mathtt{SUC}_{k+1}]|(k = 0, 1, 2, 3, 4)$.

Game 0 and Game 1 are identical games unless when the challenger generates $r \xleftarrow{r} \{0,1\}^\lambda$, there already exists an entry whose first component is r in $L_{\mathsf{H}} \cup L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$. We note that if L_{G^*} does not have such an entry, the same is true for L_{H^*}. Therefore, we can see that Game 0 and Game 1 are identical unless the event $\mathtt{COL}_0$ (resp. $\mathtt{COL}_1$) occurs in Game 0 (resp. Game 1), and thus we have $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \Pr[\mathtt{COL}_1]$.

Next, the only difference between Game 1 and Game 2 is how the challenger responds to a bad hash query from $\mathcal{A}$. In other words, Game 1 and Game 2 are identical unless the event $\mathtt{BHQ}_1$ (resp. $\mathtt{BHQ}_2$) occurs in Game 1 (resp. Game 2), and thus we have $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \Pr[\mathtt{BHQ}_2]$.

Moreover, Game 2 and Game 3, and Game 3 and Game 4 are identical games except for how the challenger responds to a decryption query which satisfies the condition we stated in the definition of events $\mathtt{SMTH}_i$ and $\mathtt{BDQ}_i$, respectively. In other words, Game 2 and Game 3 are identical unless the event $\mathtt{SMTH}_2$ (resp. $\mathtt{SMTH}_3$) occurs in Game 2 (resp. Game 3), and Game 3 and Game 4 are identical unless the event $\mathtt{BDQ}_3$ (resp. $\mathtt{BDQ}_4$) occurs in Game 3 (resp. Game 4). $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| \leq \Pr[\mathtt{SMTH}_3]$ and $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| \leq \Pr[\mathtt{BDQ}_4]$. Then, we get the following inequality.

$$\begin{aligned}\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda) \leq{}& \Pr[\mathtt{COL}_1] + \Pr[\mathtt{BHQ}_2] + \Pr[\mathtt{SMTH}_3] \\ &+ \Pr[\mathtt{BDQ}_4] + |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| \end{aligned} \tag{2}$$

In addition, $\Pr[\mathtt{BHQ}_2] \leq \sum_{k=2}^{4} |\Pr[\mathtt{BHQ}_k] - \Pr[\mathtt{BHQ}_{k+1}]| + \Pr[\mathtt{BHQ}_5]$ holds. By considering analogously to the above argument, we get $|\Pr[\mathtt{BHQ}_2] - \Pr[\mathtt{BHQ}_3]| \leq \Pr[\mathtt{SMTH}_3]$ and $|\Pr[\mathtt{BHQ}_3] - \Pr[\mathtt{BHQ}_4] \leq \Pr[\mathtt{BDQ}_4]$. Therefore, the following inequality holds.

$$\Pr[\mathtt{BHQ}_2] \leq \Pr[\mathtt{SMTH}_3] + \Pr[\mathtt{BDQ}_4] + |\Pr[\mathtt{BHQ}_4] - \Pr[\mathtt{BHQ}_5]| + \Pr[\mathtt{BHQ}_5]$$

Moreover, we have $\Pr[\mathtt{BDQ}_4] \leq |\Pr[\mathtt{BDQ}_4] - \Pr[\mathtt{BDQ}_5]| + \Pr[\mathtt{BDQ}_5]$. By using these inequalities in the inequality (2), we get the inequality (1). □ (**Lemma** 2)

Below, we show the following lemmas that state each term of the right side of the inequality (1) is negligible.

Lemma 3. $\Pr[\mathrm{COL}_1] = \mathsf{negl}(\lambda)$.

Lemma 4. *Let Π be smooth. Then* $\Pr[\mathrm{SMTH}_3] = \mathsf{negl}(\lambda)$.

Lemma 5. *Let Σ be OT-CPA secure. Then* $|\Pr[\mathrm{SUC}_4] - \Pr[\mathrm{SUC}_5]| = \mathsf{negl}(\lambda)$, $|\Pr[\mathrm{BHQ}_4] - \Pr[\mathrm{BHQ}_5]| = \mathsf{negl}(\lambda)$, *and* $|\Pr[\mathrm{BDQ}_4] - \Pr[\mathrm{BDQ}_5]| = \mathsf{negl}(\lambda)$.

Lemma 6. *Let Π be OW-CPA secure. Then* $\Pr[\mathrm{BHQ}_5] = \mathsf{negl}(\lambda)$.

Lemma 7. *Let Π be OW-CPA secure. Then* $\Pr[\mathrm{BDQ}_5] = \mathsf{negl}(\lambda)$.

Proof of Lemma 3. Since $\mathcal{A}$ is a PPT algorithm, there is a polynomial of λ which is the upper bound of the number of total entries in $L_{\mathsf{H}} \cup L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$. Let $Q = Q(\lambda)$ denote this upper bound. Then, the probability that when the challenger generates $r \xleftarrow{r} \{0,1\}^\lambda$ to respond to a KDM query form $\mathcal{A}$, there is an entry of the form $(r, \cdot)$ in $L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$, or there is an entry of the form $((r, \cdot), \cdot)$ in L_{H} is at most $\frac{Q}{2^\lambda}$. $\mathcal{A}$ makes a KDM query at most q_e times, and q_e is a polynomial of λ. Therefore, we have $\Pr[\mathtt{COL}_1] \leq \frac{Q \cdot q_e}{2^\lambda} = \mathsf{negl}(\lambda)$. □ (**Lemma 3**)

Proof of Lemma 4. We first define the following event for every $i \in [q_d]$.

$\mathtt{SMTH}_3^i$: In Game 3, the i-th decryption query (j, c, d) made by $\mathcal{A}$ satisfies the following two conditions, where $\mathsf{Dec}(sk_j, c) = r$: **(1)** There does not exist an entry of the form $((r, d), \cdot)$ in $L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$, and **(2)** $c = \mathsf{Enc}(pk_j, r; \mathsf{HH}^*(r, d))$.

When the condition **(1)** is satisfied, the value of $\mathsf{HH}^*(r, d)$ is defined with a newly generated uniformly random value. Then, the above condition **(2)** means that $c = \mathsf{Enc}(pk_j, r; R)$ holds, where $R \xleftarrow{r} \{0,1\}^n$. In addition, $\Pr[\mathtt{SMTH}_3] \leq \sum_{i \in [q_d]} \Pr[\mathtt{SMTH}_3^i]$ holds. Then, using the adversary $\mathcal{A}$ that attacks FO_2, we construct the following adversary $\mathcal{B}$ that attacks the smoothness of Π.

Initialization. On input $((pk_1, sk_1), \cdots, (pk_\ell, sk_\ell))$, $\mathcal{B}$ first chooses $b \xleftarrow{r} \{0,1\}$ and $t \xleftarrow{r} [q_d]$. Then, $\mathcal{B}$ sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$. Finally, $\mathcal{B}$ sets $\mathbf{sk} = (sk_1, \cdots, sk_\ell)$ and $L_{kdm} = L_{\mathsf{H}} = L_{\mathsf{H}^*} = L_{\mathsf{G}} = L_{\mathsf{G}^*} = \emptyset$.

Hash queries. If $\mathcal{A}$ queries (r, d) to H, $\mathcal{B}$ simulates H. Namely, $\mathcal{B}$ first checks whether there is an entry of the form $((r, d), R)$ in L_{H}. If so, $\mathcal{B}$ returns R to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates $R \xleftarrow{r} \{0,1\}^n$, adds $((r, d), R)$ to L_{H}, and returns R to $\mathcal{A}$. If $\mathcal{A}$ queries r to G, $\mathcal{B}$ simulates G analogously.

KDM queries. For a KDM query (j, f) from $\mathcal{A}$, $\mathcal{B}$ computes $m_1 \leftarrow f^{\mathsf{HH}^*, \mathsf{GG}^*}(\mathbf{sk})$ and $m_0 \leftarrow 0^{|m_1|}$. Here, $\mathcal{B}$ correctly forms L_{H}, L_{H^*}, L_{G}, and L_{G^*} through to the end, and thus if f calls HH^* or GG^*, $\mathcal{B}$ can simulate them for f. Next, $\mathcal{B}$ generates $r \xleftarrow{r} \{0,1\}^\lambda$, $K \xleftarrow{r} \{0,1\}^\lambda$, and $R \xleftarrow{r} \{0,1\}^n$. Then, $\mathcal{B}$ computes $c \leftarrow \mathsf{Enc}(pk_j, r; R)$ and $d \leftarrow \mathsf{E}(K, m_b)$, and returns (c, d) to $\mathcal{A}$. Finally, $\mathcal{B}$ adds (r, K) to L_{G^*}, $((r, d), R)$ to L_{H^*}, and (j, c, d) to L_{kdm}.

Decryption queries. For the i-th decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$, $\mathcal{B}$ responds as follows.

- In the case $i < t$, if there does not exist an entry $((r, d), R) \in L_{\mathsf{H}}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ first checks whether there is an entry of the form (r, K) in $L_{\mathsf{G}} \cup L_{\mathsf{G}^*}$. If so, $\mathcal{B}$ returns $m \leftarrow \mathsf{D}(K, d)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates $K \xleftarrow{r} \{0,1\}^\lambda$, adds (r, K) to L_{G}, and returns $m \leftarrow \mathsf{D}(K, d)$ to $\mathcal{A}$.
- In the case $i = t$, $\mathcal{B}$ first computes $r \leftarrow \mathsf{Dec}(sk_j, c)$. Then, if there exists an entry of the form $((r, d), \cdot)$ in $L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$, $\mathcal{B}$ aborts with output $\perp$. Otherwise, $\mathcal{B}$ outputs (j, r, c) and terminates.

$\mathcal{B}$ perfectly simulates Game 3 until $\mathcal{A}$ makes the t-th decryption query. We note that since t is chosen from $[q_d]$ uniformly at random and independently of $\mathcal{A}$,

and is information-theoretically hidden from the view of $\mathcal{A}$, the choice of t does not affect the behavior of $\mathcal{B}$. When $\mathcal{A}$ makes the t-th decryption query (j, c, d), $\mathcal{B}$ first computes $r \leftarrow \mathsf{Dec}(sk_j, c)$, and if there does not exist an entry of the form $((r, d), \cdot)$ in $L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$, $\mathcal{B}$ outputs (j, r, c) and terminates. Otherwise, $\mathcal{B}$ outputs $\perp$ and terminates. We can see that $\mathcal{B}$ succeeds in breaking the smoothness of Π if and only if $c = \mathsf{Enc}(pk_j, r; R)$ holds for a fresh randomness R, which means that the event $\mathtt{SMTH}_3^t$ occurs in Game 3 which $\mathcal{B}$ simulates for $\mathcal{A}$. Therefore, we can estimate the advantage of $\mathcal{B}$ $\mathsf{Adv}_{\Pi,\mathcal{B}}^{smth}(\lambda)$ as follows.

$$\begin{aligned}\mathsf{Adv}_{\Pi,\mathcal{B}}^{smth}(\lambda) &= \sum_{i\in[q_d]} \Pr[\mathtt{SMTH}_3^i \wedge t = i] \\ &= \sum_{i\in[q_d]} \Pr[\mathtt{SMTH}_3^i] \cdot \Pr[t = i] = \frac{1}{q_d} \sum_{i\in[q_d]} \Pr[\mathtt{SMTH}_3^i]\end{aligned}$$

Therefore, we see that $\Pr[\mathtt{SMTH}_3] \leq q_d \cdot \mathsf{Adv}_{\Pi,\mathcal{B}}^{smth}(\lambda)$. Since Π is smooth and q_d is a polynomial of λ, we have $\Pr[\mathtt{SMTH}_3] = \mathsf{negl}(\lambda)$. □ (**Lemma** 4)

Proof of Lemma 5. Using the adversary $\mathcal{A}$ that attacks FO_2, we construct the adversaries $\mathcal{B}_{\mathtt{SUC}}$, $\mathcal{B}_{\mathtt{BHQ}}$, and $\mathcal{B}_{\mathtt{BDQ}}$ all of which attack the OT-CPA security of Σ. We first describe $\mathcal{B}_{\mathtt{SUC}}$ below.

Initialization. On input security parameter 1^λ, $\mathcal{B}_{\mathtt{SUC}}$ first chooses $t \xleftarrow{r} [q_e]$. Then, $\mathcal{B}_{\mathtt{SUC}}$ generates ℓ key pairs $(pk_j, sk_j) \leftarrow \mathsf{KG}(1^\lambda)(j = 1, \cdots, \ell)$ and sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$. Finally, $\mathcal{B}_{\mathtt{SUC}}$ sets $\mathbf{sk} = (sk_1, \cdots, sk_\ell)$ and $L_{kdm} = L_{\mathsf{H}} = L_{\mathsf{H}^*} = L_{\mathsf{G}} = L_{\mathsf{G}^*} = \emptyset$.

Hash queries. For a hash query from $\mathcal{A}$, $\mathcal{B}_{\mathtt{SUC}}$ responds in the same manner as $\mathcal{B}$ in the proof of Lemma 4.

KDM queries. For the i-th KDM query (j, f) from $\mathcal{A}$, $\mathcal{B}_{\mathtt{SUC}}$ responds as follows.

- In the case $i < t$, $\mathcal{B}_{\mathtt{SUC}}$ first computes $m_1 \leftarrow f^{\mathsf{HH}^*,\mathsf{GG}^*}(\mathbf{sk})$ and $m_0 \leftarrow 0^{|f(\cdot)|}$. Since $\mathcal{B}_{\mathtt{SUC}}$ correctly forms L_{H}, L_{H^*}, L_{G}, and L_{G^*} up to this point, when f calls HH^* and GG^*, $\mathcal{B}_{\mathtt{SUC}}$ can simulate them for f. Next, $\mathcal{B}_{\mathtt{SUC}}$ generates $r \xleftarrow{r} \{0,1\}^\lambda$, $K \xleftarrow{r} \{0,1\}^\lambda$, and $R \xleftarrow{r} \{0,1\}^n$. Then $\mathcal{B}_{\mathtt{SUC}}$ computes $c \leftarrow \mathsf{Enc}(pk_j, r; R)$ and $d \leftarrow \mathsf{E}(K, m_b)$, and returns (c, d) to $\mathcal{A}$. Finally, $\mathcal{B}_{\mathtt{SUC}}$ adds (r, K) to L_{G^*}, $((r, d), R)$ to L_{H^*}, and (j, c, d) to L_{kdm}.
- In the case $i = t$, $\mathcal{B}_{\mathtt{SUC}}$ first computes $m_1 \leftarrow f^{\mathsf{HH}^*,\mathsf{GG}^*}(\mathbf{sk})$ and $m_0 \leftarrow 0^{|f(\cdot)|}$. Since $\mathcal{B}_{\mathtt{SUC}}$ correctly forms L_{H}, L_{H^*}, L_{G}, and L_{G^*} up to this point, when f calls HH^* and GG^*, $\mathcal{B}_{\mathtt{SUC}}$ can simulate them for f. Next, $\mathcal{B}_{\mathtt{SUC}}$ sends (m_0, m_b) as a challenge query to the challenger to get the answer d. Then, $\mathcal{B}_{\mathtt{SUC}}$ generates $r \xleftarrow{r} \{0,1\}^\lambda$ and $R \xleftarrow{r} \{0,1\}^n$, computes $c \leftarrow \mathsf{Enc}(pk_j, r; R)$, and returns (c, d) to $\mathcal{A}$. Finally, $\mathcal{B}_{\mathtt{SUC}}$ adds $(r, \perp)$ to L_{G^*}, $((r, d), R)$ to L_{H^*}, and (j, c, d) to L_{kdm}.
- In the case $i > t$, $\mathcal{B}_{\mathtt{SUC}}$ first generates $r \xleftarrow{r} \{0,1\}^\lambda$, $K \xleftarrow{r} \{0,1\}^\lambda$, and $R \xleftarrow{r} \{0,1\}^n$. Then, $\mathcal{B}_{\mathtt{SUC}}$ computes $c \leftarrow \mathsf{Enc}(pk_j, r; R)$ and $d \leftarrow \mathsf{E}(K, 0^{|f(\cdot)|})$, and returns (c, d) to $\mathcal{A}$. Finally, $\mathcal{B}_{\mathtt{SUC}}$ adds (r, K) to L_{G^*}, $((r, d), R)$ to L_{H^*}, and (j, c, d) to L_{kdm}.

Decryption queries. For a decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$, if there does not exist an entry $((r, d), R) \in L_{\mathsf{H}}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$, $\mathcal{B}_{\mathtt{SUC}}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}_{\mathtt{SUC}}$ first checks whether there is an entry of the form (r, K) in L_{G}. If so, $\mathcal{B}_{\mathtt{SUC}}$ returns $m \leftarrow \mathsf{D}(K, d)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}_{\mathtt{SUC}}$ generates $K \xleftarrow{r} \{0,1\}^{\lambda}$, adds (r, K) to L_{G}, and returns $m \leftarrow \mathsf{D}(K, d)$ to $\mathcal{A}$.

Final phase. When $\mathcal{A}$ terminates with output b', $\mathcal{B}_{\mathtt{SUC}}$ outputs $\beta_{\mathtt{SUC}} = 1$ if $b = b'$. Otherwise, $\mathcal{B}_{\mathtt{SUC}}$ outputs $\beta_{\mathtt{SUC}} = 0$.

$\mathcal{B}_{\mathtt{BHQ}}$ runs in exactly the same way as $\mathcal{B}_{\mathtt{SUC}}$ except for how to determine the final output bit $\beta_{\mathtt{BHQ}}$. $\mathcal{B}_{\mathtt{BHQ}}$ determines $\beta_{\mathtt{BHQ}}$ as follows. $\mathcal{B}_{\mathtt{BHQ}}$ initially sets $\beta_{\mathtt{BHQ}} = 0$. When $\mathcal{A}$ queries r to G or queries (r, d) to H, $\mathcal{B}_{\mathtt{BHQ}}$ first responds in the same manner as $\mathcal{B}_{\mathtt{SUC}}$. In addition, $\mathcal{B}_{\mathtt{BHQ}}$ checks whether the query is a bad hash query or not. Namely, $\mathcal{B}_{\mathtt{BHQ}}$ checks whether there exists an entry in L_{G^*} whose first component is r. If so, $\mathcal{B}_{\mathtt{BHQ}}$ sets $\beta_{\mathtt{BHQ}} = 1$. When $\mathcal{A}$ terminates with output b', $\mathcal{B}_{\mathtt{BHQ}}$ outputs $\beta_{\mathtt{BHQ}}$.

$\mathcal{B}_{\mathtt{BDQ}}$ also runs in exactly the same way as $\mathcal{B}_{\mathtt{SUC}}$ except for how to determine the final output bit $\beta_{\mathtt{BDQ}}$. $\mathcal{B}_{\mathtt{BDQ}}$ determines $\beta_{\mathtt{BDQ}}$ as follows. $\mathcal{B}_{\mathtt{BDQ}}$ initially sets $\beta_{\mathtt{BDQ}} = 0$. When $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$, $\mathcal{B}_{\mathtt{BDQ}}$ first responds in the same manner as $\mathcal{B}_{\mathtt{SUC}}$. In addition, $\mathcal{B}_{\mathtt{BDQ}}$ checks whether the query is a bad decryption query or not. Namely, $\mathcal{B}_{\mathtt{BDQ}}$ checks whether the query satisfies the following condition: There exists an entry $((r, d), R) \in L_{\mathsf{H}} \cup L_{\mathsf{H}^*}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$, and if so, for such r, $(r, \cdot) \in L_{\mathsf{G}^*}$ holds. If (j, c, d) satisfies the above condition, then $\mathcal{B}_{\mathtt{BDQ}}$ sets $\beta_{\mathtt{BDQ}} = 1$. When $\mathcal{A}$ terminates with output b', $\mathcal{B}_{\mathtt{BDQ}}$ outputs $\beta_{\mathtt{BDQ}}$.

Let β be the challenge bit in the game between the challenger and $\mathcal{B}_{\mathtt{SUC}}$. Then, the advantage of $\mathcal{B}_{\mathtt{SUC}}$ is estimated as follows.

$$\begin{aligned}\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{SUC}}}(\lambda) &= \frac{1}{2}|\Pr[\beta_{\mathtt{SUC}} = 1|\beta = 1] - \Pr[\beta_{\mathtt{SUC}} = 1|\beta = 0]| \\ &= \frac{1}{2}|\sum_{k\in[q_e]} \Pr[\beta_{\mathtt{SUC}} = 1 \wedge t = k|\beta = 1] \\ &\qquad - \sum_{k\in[q_e]} \Pr[\beta_{\mathtt{SUC}} = 1 \wedge t = k|\beta = 0]|\end{aligned}$$

Here, for any $k \in [q_e]$, we have the following two equations.

$$\begin{aligned}\Pr[\beta_{\mathtt{SUC}} = 1 \wedge t = k|\beta = 1] &= \Pr[t = k|\beta = 1]\Pr[\beta_{\mathtt{SUC}} = 1|\beta = 1 \wedge t = k] \\ \Pr[\beta_{\mathtt{SUC}} = 1 \wedge t = k|\beta = 0] &= \Pr[t = k|\beta = 0]\Pr[\beta_{\mathtt{SUC}} = 1|\beta = 0 \wedge t = k]\end{aligned}$$

We note that t is chosen from $[q_e]$ uniformly at random and independently of β. Hence, for all $k \in [q_e]$, we have $\Pr[t = k|\beta = 1] = \Pr[t = k|\beta = 0] = \frac{1}{q_e}$. Moreover, for every $k \in [q_e - 1]$, in the cases $\beta = 1 \wedge t = k$ and $\beta = 0 \wedge t = k + 1$, $\mathcal{B}_{\mathtt{SUC}}$ responds to KDM queries from $\mathcal{A}$ in exactly the same way. In the above two cases, the only difference is whether $\mathcal{B}_{\mathtt{SUC}}$ computes $f^{\mathsf{HH}^*,\mathsf{GG}^*}(\mathbf{sk})$ to responds to the $(k + 1)$-th KDM query from $\mathcal{A}$. Due to this difference, in the above two

cases, the manner L_{H} and L_{G} are formed is different. However, since $\mathcal{A}$ cannot see the contents of L_{H} and L_{G}, this does not affect the behavior of $\mathcal{A}$. Therefore, we have $\Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 1 \wedge t = k] = \Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 0 \wedge t = k+1]$ for any $k \in [q_e - 1]$. From these, we have the following equality.

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{SUC}}}(\lambda) = \frac{1}{2q_e} |\Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 1 \wedge t = q_e] - \Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 0 \wedge t = 1]|$$

Since $\mathcal{B}_{\mathtt{BHQ}}$ and $\mathcal{B}_{\mathtt{BDQ}}$ run in exactly the same way as $\mathcal{B}_{\mathtt{SUC}}$ except for how to determine the final output bit, all of the above arguments also hold for $\mathcal{B}_{\mathtt{BHQ}}$ and $\mathcal{B}_{\mathtt{BDQ}}$. Therefore, we also have the following equalities.

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{BHQ}}}(\lambda) = \frac{1}{2q_e} |\Pr[\beta_{\mathtt{BHQ}} = 1 | \beta = 1 \wedge t = q_e] - \Pr[\beta_{\mathtt{BHQ}} = 1 | \beta = 0 \wedge t = 1]|$$

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{BDQ}}}(\lambda) = \frac{1}{2q_e} |\Pr[\beta_{\mathtt{BDQ}} = 1 | \beta = 1 \wedge t = q_e] - \Pr[\beta_{\mathtt{BDQ}} = 1 | \beta = 0 \wedge t = 1]|$$

We note that, until $\mathcal{A}$ makes the t-th KDM query, $\mathcal{B}_{\mathtt{SUC}}$ correctly forms L_{H}, L_{H^*}, L_{G}, and L_{G^*}, and thus $\mathcal{B}_{\mathtt{SUC}}$ can compute KDM functions up to this point. On the other hand, $\mathcal{B}_{\mathtt{SUC}}$ cannot simulate the t-th entry of L_{G^*} because $\mathcal{B}$ simulates the security game for $\mathcal{A}$ so that the second component of the t-th entry of L_{G^*} is the value of the key the challenger generates, and thus $\mathcal{B}_{\mathtt{SUC}}$ does not know it. Therefore, when responding to the subsequent KDM queries, $\mathcal{B}_{\mathtt{SUC}}$ cannot compute KDM functions correctly. However, since $\mathcal{B}_{\mathtt{SUC}}$ only needs the output length of KDM functions to respond to the $(t+1)$-th and subsequent KDM queries, and KDM functions are length regular, $\mathcal{B}_{\mathtt{SUC}}$ need not compute f. Then, we see that when $\beta = 1 \wedge t = q_e$, $\mathcal{B}_{\mathtt{SUC}}$ perfectly simulates Game 4 for $\mathcal{A}$. On the other hand, when $\beta = 0 \wedge t = 1$, $\mathcal{B}_{\mathtt{SUC}}$ perfectly simulates Game 5 for $\mathcal{A}$. We note that t is information-theoretically hidden from the view of $\mathcal{A}$, and thus the choice of t does not affect the behavior of $\mathcal{A}$. Here, this argument also holds for $\mathcal{B}_{\mathtt{BHQ}}$ and $\mathcal{B}_{\mathtt{BDQ}}$.

In addition, $\mathcal{B}_{\mathtt{SUC}}$ outputs 1 only when $\mathcal{A}$ succeeds in guessing b, that is, $b = b'$ occurs, $\mathcal{B}_{\mathtt{BHQ}}$ outputs 1 only when $\mathcal{A}$ makes a bad hash query, and $\mathcal{B}_{\mathtt{BDQ}}$ outputs 1 only when $\mathcal{A}$ makes a bad decryption query. Therefore, we have following equalities.

$$\Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 1 \wedge t = q_e] = \Pr[\mathtt{SUC}_4],\ \Pr[\beta_{\mathtt{SUC}} = 1 | \beta = 0 \wedge t = 1] = \Pr[\mathtt{SUC}_5]$$
$$\Pr[\beta_{\mathtt{BHQ}} = 1 | \beta = 1 \wedge t = q_e] = \Pr[\mathtt{BHQ}_4],\ \Pr[\beta_{\mathtt{BHQ}} = 1 | \beta = 0 \wedge t = 1] = \Pr[\mathtt{BHQ}_5]$$
$$\Pr[\beta_{\mathtt{BDQ}} = 1 | \beta = 1 \wedge t = q_e] = \Pr[\mathtt{BDQ}_4],\ \Pr[\beta_{\mathtt{BDQ}} = 1 | \beta = 0 \wedge t = 1] = \Pr[\mathtt{BDQ}_5]$$

Therefore, we get the following equalities.

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{SUC}}}(\lambda) = \frac{1}{2q_e} |\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]|$$

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{BHQ}}}(\lambda) = \frac{1}{2q_e} |\Pr[\mathtt{BHQ}_4] - \Pr[\mathtt{BHQ}_5]|$$

$$\mathsf{Adv}^{otcpa}_{\Sigma,\mathcal{B}_{\mathtt{BDQ}}}(\lambda) = \frac{1}{2q_e} |\Pr[\mathtt{BDQ}_4] - \Pr[\mathtt{BDQ}_5]|$$

Since Σ is OT-CPA secure and q_e is a polynomial of λ, we see that $|\Pr[\mathtt{SUC}_4] - \Pr[\mathtt{SUC}_5]| = \mathsf{negl}(\lambda)$, $|\Pr[\mathtt{BHQ}_4] - \Pr[\mathtt{BHQ}_5]| = \mathsf{negl}(\lambda)$, and $|\Pr[\mathtt{BDQ}_4] - \Pr[\mathtt{BDQ}_5]| = \mathsf{negl}(\lambda)$. □ (**Lemma** 5)

Proof of Lemma 6. Using the adversary $\mathcal{A}$ that attacks FO_2, we construct the following adversary $\mathcal{B}$ that attacks the List-OW-CPA security of Π. We note that since Π is OW-CPA secure, by Lemma 1, Π is also List-OW-CPA secure.

Initialization. On input $(pk_1, \cdots, pk_\ell)$, $\mathcal{B}$ sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$, and $\mathcal{B}$ sets $L_{kdm} = L_{\mathsf{H}} = L_{\mathsf{G}} = L_{ans} = \emptyset$.

Hash queries. If $\mathcal{A}$ queries (r, d) to H, $\mathcal{B}$ simulates H. Namely, $\mathcal{B}$ first checks whether there is an entry of the form $((r, d), R)$ in L_{H}. If so, $\mathcal{B}$ returns R to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates $R \xleftarrow{r} \{0,1\}^n$, adds $((r, d), R)$ to L_{H}, and returns R to $\mathcal{A}$. Then, $\mathcal{B}$ adds r to L_{ans}. If $\mathcal{A}$ queries r to G, $\mathcal{B}$ simulates G analogously to H as above, and adds r to L_{ans}.

KDM queries. For a KDM query (j, f) from $\mathcal{A}$, $\mathcal{B}$ first queries j to the challenger as an encryption query and gets the answer c. Then, $\mathcal{B}$ generates $K \xleftarrow{r} \{0,1\}^\lambda$ and computes $d \leftarrow \mathsf{E}(K, 0^{|f(\cdot)|})$. Finally, $\mathcal{B}$ adds (j, c, d) to L_{kdm}, and returns (c, d) to $\mathcal{A}$.

Decryption queries. For a decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$, $\mathcal{B}$ responds in the same manner as $\mathcal{B}_{\mathtt{SUC}}$ in the proof of Lemma 5.

Final phase. When $\mathcal{A}$ terminates with output b', $\mathcal{B}$ outputs L_{ans}.

In the List-OW-CPA game, the challenger maintains the list L_{enc} which stores plaintexts of the challenge ciphertexts. Then, the advantage of $\mathcal{B}$ is $\Pr[L_{enc} \cap L_{ans} \neq \emptyset]$. We see that $\mathcal{B}$ perfectly simulates Game 5 for $\mathcal{A}$. If $\mathcal{A}$ queries r as a G query or (r, d) as a H query, $\mathcal{B}$ adds r to L_{ans}. In addition, in Game 5, a new entry is added to L_{G} and L_{H} only when $\mathcal{A}$ makes a G query and H query, respectively. Therefore, we can write $L_{ans} = \{r | (r, \cdot) \in L_{\mathsf{G}} \vee ((r, \cdot), \cdot) \in L_{\mathsf{H}}\}$. Here, $(r, \cdot) \in L_{\mathsf{G}}$ (resp. $((r, \cdot), \cdot) \in L_{\mathsf{H}}$) indicates that there exists an entry in L_{G} (resp. L_{H}) whose first component is r.

On the other hand, every time $\mathcal{A}$ makes a KDM query (j, f), $\mathcal{B}$ sends j to the challenger as an encryption query and gets the answer c. Here, let c be an encryption of r. Then, by the above encryption query from $\mathcal{B}$, the challenger adds r to L_{enc}. On the other hand, in Game 5, the hash value of r is computed using G^* at this point, and thus an entry of the form $(r, \cdot)$ is added to L_{G^*}. Therefore, L_{enc} can be seen as the set of the first components of the entries in L_{G^*} in Game 5. (Actually, $\mathcal{B}$ does not make L_{G^*} by itself, but $\mathcal{B}$ does not need L_{G^*} to simulate Game 5 for $\mathcal{A}$.) From these, we see that $L_{enc} \cap L_{ans} \neq \emptyset$ holds if the event $\mathtt{BHQ}_5$ occurs in Game 5 which $\mathcal{B}$ simulates for $\mathcal{A}$. Therefore, we can see that $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell}(\lambda) \geq \Pr[\mathtt{BHQ}_5]$. Since Π is List-OW-CPA secure, we see that $\Pr[\mathtt{BHQ}_5] = \mathsf{negl}(\lambda)$. □ (**Lemma** 6)

Proof of Lemma 7. First, we define the following two events.

$\mathtt{BDQ1}_5$**:** In Game 5, $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ satisfying the following condition: There exists an entry $((r, d), R) \in L_{\mathsf{H}}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$ and $(r, \cdot) \in L_{\mathsf{G}^*}$.

$\mathtt{BDQ2}_5$**:** In Game 5, $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ satisfying the following condition: There exists an entry $((r, d), R) \in L_{\mathsf{H}^*}$ which satisfies $c = \mathsf{Enc}(pk_j, r; R)$. (We note that if there is an entry of the form $((r, \cdot), \cdot) \in L_{\mathsf{H}^*}$, then there is an entry of the form $(r, \cdot) \in L_{\mathsf{G}^*}$.)

By the definition of the events, obviously, $\mathtt{BDQ}_5 = \mathtt{BDQ1}_5 \vee \mathtt{BDQ2}_5$ holds, and thus we have $\Pr[\mathtt{BDQ}_5] \leq \Pr[\mathtt{BDQ1}_5] + \Pr[\mathtt{BDQ2}_5]$. Here, regarding $\mathtt{BDQ2}_5$, when $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ satisfying the condition of $\mathtt{BDQ2}_5$, it holds that $r = r^*$, $R = R^*$, and $d = d^*$ for some entry $(j^*, c^*, d^*) \in L_{kdm}$, where $c^* = \mathsf{Enc}(pk_{j^*}, r^*; R^*)$. In addition, in this case, $j \neq j^*$ also holds. The reason is as follows. If $j = j^*$ holds, then $c = \mathsf{Enc}(pk_j, r; R) = \mathsf{Enc}(pk_{j^*}, r^*; R^*) = c^*$ holds, and thus we have $(j, c, d) = (j^*, c^*, d^*)$, which contradicts $(j, c, d) \notin L_{kdm}$. Therefore, $j \neq j^*$ holds. From these, the event $\mathtt{BDQ2}_5$ implies the following event.

$\mathtt{BDQ2}^*_5$**:** In Game 5, $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ satisfying the following condition: For some entry $(j^*, c^*, d^*) \in L_{kdm}$, where $c^* = \mathsf{Enc}(pk_{j^*}, r^*; R^*)$, it holds that $c = \mathsf{Enc}(pk_j, r^*; R^*)$ and $j \neq j^*$.

Here, we have $\Pr[\mathtt{BDQ2}_5] \leq \Pr[\mathtt{BDQ2}^*_5]$.

Then, using the adversary $\mathcal{A}$ that attacks FO_2, we construct the following adversary $\mathcal{B}$ that attacks the List-OW-CPA security of Π. Since Π is OW-CPA secure, from Lemma 1, Π is also List-OW-CPA secure. Here, we note that $\mathcal{B}$ attacks the List-OW-CPA security of Π in the case the number of keys is $\ell - 1$.

Initialization. On input $(pk^*_1, \cdots, pk^*_{\ell-1})$, $\mathcal{B}$ first chooses $s \xleftarrow{r} [\ell]$ and generates $(pk_s, sk_s) \leftarrow \mathsf{KG}(1^\lambda)$. Then, for $1 \leq j < s$, $\mathcal{B}$ sets $pk_j = pk^*_j$, and for $s < j \leq \ell$, $\mathcal{B}$ sets $pk_j = pk^*_{j-1}$. Finally, $\mathcal{B}$ sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$, and sets $L_{kdm} = L_{\mathsf{H}} = L_{\mathsf{G}} = L^1_{ans} = L^2_{ans} = \emptyset$.

Hash queries. If $\mathcal{A}$ queries (r, d) to H, $\mathcal{B}$ first simulates H. Namely, $\mathcal{B}$ first checks whether there is an entry of the form $((r, d), R)$ in L_{H}. If so, $\mathcal{B}$ returns R to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates $R \xleftarrow{r} \{0,1\}^n$, adds $((r, d), R)$ to L_{H}, and returns R to $\mathcal{A}$. Then, $\mathcal{B}$ adds r to L^1_{ans}. If $\mathcal{A}$ queries r to G, $\mathcal{B}$ just simulates G analogously to H. ($\mathcal{B}$ does not add r to L^1_{ans} in the case of G query.)

KDM queries. For a KDM query (j, f) from $\mathcal{A}$, $\mathcal{B}$ responds as follows.

- In the case $j \neq s$, $\mathcal{B}$ first queries j if $j < s$ and $j - 1$ if $j > s$ to the challenger as an encryption query and gets the answer c. Then, $\mathcal{B}$ generates $K \xleftarrow{r} \{0,1\}^\lambda$ and computes $d \leftarrow \mathsf{E}(K, 0^{|f(\cdot)|})$. Finally, $\mathcal{B}$ adds (j, c, d) to L_{kdm}, and returns (c, d) to $\mathcal{A}$.
- In the case $j = s$, $\mathcal{B}$ first generates $r \xleftarrow{r} \{0,1\}^\lambda$, $K \xleftarrow{r} \{0,1\}^\lambda$, and $R \xleftarrow{r} \{0,1\}^\lambda$. Then, $\mathcal{B}$ computes $c = \mathsf{Enc}(pk_j, r; R)$ and $d \leftarrow \mathsf{E}(K, 0^{|f(\cdot)|})$. Finally, $\mathcal{B}$ adds (j, c, d) to L_{kdm}, and returns (c, d) to $\mathcal{A}$.

Decryption queries. For a decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$, if $j = s$, $\mathcal{B}$ first computes $r \leftarrow \mathsf{Dec}(sk_s, c)$, and adds r to L^2_{ans} if $r \neq \perp$. Then, $\mathcal{B}$ responds in the same manner as $\mathcal{B}_{\mathtt{SUC}}$ in the proof of Lemma 5.

Final phase. When $\mathcal{A}$ terminates with output b', $\mathcal{B}$ chooses $\gamma \xleftarrow{r} \{1, 2\}$ and outputs L^γ_{ans}.

We see that $\mathcal{B}$ perfectly simulates Game 5 for $\mathcal{A}$. In the initialization step, $\mathcal{B}$ chooses $s \xleftarrow{r} [\ell]$ and generates $(pk_s, sk_s) \leftarrow \mathsf{KG}(1^\lambda)$. Since s is information-theoretically hidden from the view of $\mathcal{A}$, this does not affect the behavior of $\mathcal{A}$. We note that, in the List-OW-CPA game, the challenger maintains the list L_{enc} which stores plaintexts of the challenge ciphertexts. Then, it holds that $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell-1}(\lambda) = \Pr[L_{enc} \cap L^{\gamma}_{ans} \neq \emptyset]$. Here, γ is a randomness over $\{1, 2\}$ which $\mathcal{B}$ chooses in the final phase, and thus the following equality holds.

$$\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell-1}(\lambda) = \frac{1}{2}\Pr[L_{enc} \cap L^{1}_{ans} \neq \emptyset] + \frac{1}{2}\Pr[L_{enc} \cap L^{2}_{ans} \neq \emptyset]$$

In the following, we first consider $\Pr[L_{enc} \cap L^{1}_{ans} \neq \emptyset]$. When the event $\mathtt{BDQ1}_5$ occurs in Game 5 which $\mathcal{B}$ simulates for $\mathcal{A}$, there exists an entry $((r^*, d), R) \in L_{\mathsf{H}}$ which satisfies $c = \mathsf{Enc}(pk_j, r^*, R)$ and $(r^*, \cdot) \in L_{\mathsf{G}^*}$ for some decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$. We note that only when $\mathcal{A}$ makes a H query, an entry is added to L_{H}. Thus, $((r^*, d), R) \in L_{\mathsf{H}}$ means that $\mathcal{A}$ has queries (r^*, d) as a H query. Therefore, in this case, L^{1}_{ans} contains r^*. Also, $(r^*, \cdot) \in L_{\mathsf{G}^*}$ means that r^* is generated to compute the answer to a KDM query from $\mathcal{A}$. (Actually, $\mathcal{B}$ does not make L_{G^*} by itself, but $\mathcal{B}$ need not make L_{G^*} to simulate Game 5 for $\mathcal{A}$.) Here, let r^* be generated to compute the answer (c^*, d^*) to a KDM query (j^*, f) from $\mathcal{A}$. In other words, let c^* be an encryption of r^* under pk_{j^*}. Then, if $j^* \neq s$, c^* is computed by the challenger, and thus L_{enc} contains r^*. On the other hand, if $j^* = s$, c^* is computed by $\mathcal{B}$, and thus L_{enc} does not contain r^*. Therefore, at least when $\mathcal{A}$ makes a decryption query satisfying the condition of the event $\mathtt{BDQ1}_5$, and $j^* \neq s$ holds for the above j^*, $L_{enc} \cap L^{1}_{ans} \neq \emptyset$ holds. Since s is chosen from $[\ell]$ uniformly at random, and is information-theoretically hidden from the view of $\mathcal{A}$, the choice of s is independent of the behavior of $\mathcal{A}$. Therefore, the probability that $j^* \neq s$ holds under the condition that $\mathcal{A}$ has made a decryption query satisfying the condition of $\mathtt{BDQ1}_5$ is $\frac{\ell-1}{\ell}$. Moreover, since $\mathcal{B}$ perfectly simulates Game 5 for $\mathcal{A}$, the probability that $\mathcal{A}$ makes a decryption query satisfying the condition of the event $\mathtt{BDQ1}_5$ is $\Pr[\mathtt{BDQ1}_5]$. From these, we have $\Pr[L_{enc} \cap L^{1}_{ans} \neq \emptyset] \geq \frac{\ell-1}{\ell}\Pr[\mathtt{BDQ1}_5]$.

Next, we consider $\Pr[L_{enc} \cap L^{2}_{ans} \neq \emptyset]$. When the event $\mathtt{BDQ2}^*_5$ occurs in Game 5 which $\mathcal{B}$ simulates for $\mathcal{A}$, for some decryption query $(j, c, d) \notin L_{kdm}$ from $\mathcal{A}$ and some entry $(j^*, c^*, d^*) \in L_{kdm}$, it holds that $c = \mathsf{Enc}(pk_j, r^*; R^*)$ and $j \neq j^*$, where $c^* = \mathsf{Enc}(pk_{j^*}, r^*; R^*)$. Then, if $j = s$, L^{2}_{ans} contains r^*. In addition, in this case, $j^* \neq j = s$ holds, and thus L_{enc} also contains r^*. Therefore, at least when $\mathcal{A}$ makes a decryption query $(j, c, d) \notin L_{kdm}$ satisfying the condition of the event $\mathtt{BDQ2}^*_5$, and $j = s$ holds, $L_{enc} \cap L^{2}_{ans} \neq \emptyset$ holds. Since the choice of s is independent of $\mathcal{A}$, the probability that $j = s$ holds under the condition that $\mathcal{A}$ has made a decryption query satisfying the condition of $\mathtt{BDQ2}^*_5$ is $\frac{1}{\ell}$. Moreover, since $\mathcal{B}$ perfectly simlates Game 5 for $\mathcal{A}$, the probability that $\mathcal{A}$ makes a decryption query satisfying the condition of the event $\mathtt{BDQ2}^*_5$ is $\Pr[\mathtt{BDQ2}^*_5]$. Therefore, we get $\Pr[L_{enc} \cap L^{2}_{ans} \neq \emptyset] \geq \frac{1}{\ell}\Pr[\mathtt{BDQ2}^*_5]$.

From these, we can estimate $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell-1}(\lambda)$ as follows.

$$\begin{aligned}\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell-1}(\lambda) &= \frac{1}{2}\Pr[L_{enc} \cap L^1_{ans} \neq \emptyset] + \frac{1}{2}\Pr[L_{enc} \cap L^2_{ans} \neq \emptyset] \\ &\geq \frac{1}{2} \cdot \frac{\ell-1}{\ell}\Pr[\mathtt{BDQ1}_5] + \frac{1}{2} \cdot \frac{1}{\ell}\Pr[\mathtt{BDQ2}^*_5] \\ &\geq \frac{1}{2\ell}(\Pr[\mathtt{BDQ1}_5] + \Pr[\mathtt{BDQ2}^*_5]) \\ &\geq \frac{1}{2\ell}(\Pr[\mathtt{BDQ1}_5] + \Pr[\mathtt{BDQ2}_5]) \geq \frac{1}{2\ell}\Pr[\mathtt{BDQ}_5]\end{aligned}$$

From the above, we have $\Pr[\mathtt{BDQ}_5] \leq 2\ell \cdot \mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{B},\ell-1}(\lambda)$. Since Π is List-OW-CPA secure and ℓ is a polynomial of λ, we see that $\Pr[\mathtt{BDQ}_5] = \mathsf{negl}(\lambda)$. □ (**Lemma** 7)

From the inequality (1) and Lemmas 3 to 7, we have $\mathsf{Adv}^{kdmcca}_{\mathsf{FO}_2,\mathcal{A},\ell}(\lambda) = \mathsf{negl}(\lambda)$. Since the choice of ℓ and $\mathcal{A}$ is arbitrary, we see that FO_2 is KDM-CCA secure in the random oracle model. □ (**Theorem** 4)

A The Proof of Lemma 1

Here, we define OW-CPA security for PKE schemes, and then prove Lemma 1.

Definition 8. (*OW-CPA* security). *Let Π be a PKE scheme whose message space is $\mathcal{M}$. We define the OW-CPA game between a challenger and an adversary $\mathcal{A}$ as follows.*

Initialization. *First the challenger generates a key pair $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$. Then, the challenger generates $m \xleftarrow{r} \mathcal{M}$ and $c \leftarrow \mathsf{Enc}(pk, m)$, and sends (pk, c) to $\mathcal{A}$.*

Final phase. *$\mathcal{A}$ outputs m'.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as follows.

$$\mathsf{Adv}^{owcpa}_{\Pi,\mathcal{A}}(\lambda) = \Pr[m = m']$$

We say that Π is OW-CPA secure if for any PPT adversary $\mathcal{A}$, we have $\mathsf{Adv}^{owcpa}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

We give the proof of Lemma 1 below.

Proof of Lemma 1. Let $\mathcal{A}$ be an adversary for the List-OW-CPA security of Π which makes at most q encryption queries and outputs a list L_{ans} that contains at most p elements. Let $\ell = \ell(\lambda)$ be any polynomial. Then, we define the following event $S^{i,k}$ for any $i \in [q]$ and $k \in [p]$.

$S^{i,k}$**:** Let m_i be the i-th entry of L_{enc} and m'_k be the k-th entry of L_{ans}. Then, $m_i = m'_k$ holds.

Here, we have $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{A},\ell}(\lambda) \leq \sum_{i\in[q]}\sum_{k\in[p]}\Pr[S^{i,k}]$.

Then, using $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ which attacks the OW-CPA security of Π.

Initialization. On the input (pk^*, c^*), $\mathcal{B}$ first chooses $s \xleftarrow{r} [\ell]$ and $t \xleftarrow{r} [q]$. Then, $\mathcal{B}$ sets $pk_s = pk^*$, generates $\ell - 1$ key pairs $(pk_j, sk_j) \leftarrow \mathsf{KG}(1^\lambda)(j = 1, \cdots, s-1, s+1, \cdots, \ell)$, and sends $(pk_1, \cdots, pk_\ell)$ to $\mathcal{A}$.

Encryption queries. For the i-th encryption query $j \in [\ell]$ made by $\mathcal{A}$, $\mathcal{B}$ responds as follows.

- In the case $i \neq t$, $\mathcal{B}$ generates $m \xleftarrow{r} \mathcal{M}$, computes $c \leftarrow \mathsf{Enc}(pk_j, m)$, and returns c to $\mathcal{A}$.
- In the case $i = t$, if $j \neq s$, then $\mathcal{B}$ aborts with output $\perp$. Otherwise, $\mathcal{B}$ returns c^* to $\mathcal{A}$.

Final phase. When $\mathcal{A}$ outputs L_{ans}, $\mathcal{B}$ first chooses $u \xleftarrow{r} [p]$, where p is the number of entries in L_{ans}. Then, $\mathcal{B}$ outputs the u-th entry of L_{ans}.

If $\mathcal{B}$ does not abort, $\mathcal{B}$ perfectly simulates the List-OW-CPA game for $\mathcal{A}$. We note that s, t, and u are chosen uniformly at random. In addition, if $\mathcal{B}$ does not abort, the choice of them is information-theoretically hidden from the view of $\mathcal{A}$, and thus is independent of $\mathcal{A}$. When $\mathcal{A}$ makes the t-th encryption query j_t, if $j_t = s$, $\mathcal{B}$ returns c^* which is the challenge ciphertext for $\mathcal{B}$ itself. Let c^* be an encryption of r^*. Then, the t-th entry of L_{enc} in the List-OW-CPA game which $\mathcal{B}$ simulates for $\mathcal{A}$ is r^*. In addition, in the final phase, $\mathcal{B}$ outputs the u-th entry of L_{ans} output by $\mathcal{A}$. From these, for any $i \in [q]$ and $k \in [p]$, $\mathcal{B}$ succeeds in breaking the OW-CPA security of Π if the event $S^{i,k}$ occurs in the List-OW-CPA game which $\mathcal{B}$ simulates for $\mathcal{A}$, and $t = i$, $j_t = s$, and $u = k$ hold. Therefore, we can estimate the advantage of $\mathcal{B}$ as follows.

$$\begin{aligned}
\mathsf{Adv}^{owcpa}_{\Pi,\mathcal{B}}(\lambda) &= \sum_{i\in[q]} \sum_{k\in[p]} \Pr[S^{i,k} \wedge t = i \wedge s = j_i \wedge u = k] \\
&= \sum_{i\in[q]} \sum_{k\in[q]} \Pr[S^{i,k}] \cdot \Pr[t = i] \cdot \Pr[s = j_i] \cdot \Pr[u = k] \\
&= \frac{1}{\ell pq} \sum_{i\in[q]} \sum_{k\in[p]} \Pr[S^{i,k}] \geq \frac{1}{\ell pq} \mathsf{Adv}^{mowcpa}_{\Pi,\mathcal{A},\ell}(\lambda)
\end{aligned}$$

Since Π is OW-CPA secure, and ℓ, p, and q are polynomials of λ, we see that $\mathsf{Adv}^{lowcpa}_{\Pi,\mathcal{A},\ell}(\lambda) \leq \ell pq \cdot \mathsf{Adv}^{owcpa}_{\Pi,\mathcal{B}}(\lambda) = \mathsf{negl}(\lambda)$. □ (**Lemma** 1)

References

1. IEEE standard specifications for public-key cryptography - amendment 1: additional techniques. IEEE Std 1363a–2004 (Amendment to IEEE Std 1363–2000), September 2004
2. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). J. Cryptology **20**(3), 395 (2007)
3. Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness and completeness of formal encryption: the cases of key cycles and partial information leakage. J. Comput. Secur. **17**(5), 737–797 (2009)

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)
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)
6. Backes, M., Dürmuth, M., Unruh, D.: OAEP is secure under key-dependent messages. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 506–523. Springer, Heidelberg (2008)
7. 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)
8. Bellare, M., Hoang, V., Rogaway, P.: Garbling schemes. IACR Cryptology ePrint Archive(2011). Observation of strains: 265, The proceedings version appears in ACMCCS 2012 (2012)
9. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: when and how should challenge decryption be disallowed? J. Cryptology **28**(1), 29–48 (2015)
10. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
11. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
12. 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)
13. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
14. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
15. Camenisch, J.L., 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)
16. 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)
17. Davies, G.T., Stam, M.: KDM security in the hybrid framework. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 461–480. Springer, Heidelberg (2014)
18. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). STOC **1991**, 542–552 (1991)
19. Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
20. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptology **26**(1), 80–101 (2013)

22. Hofheinz, D.: Circular chosen-ciphertext security with compact ciphertexts. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 520–536. Springer, Heidelberg (2013)
23. Kitagawa, F., Matsuda, T., Hanaoka, G., Tanaka, K.: Efficient key dependent message security amplification against chosen ciphertext attacks. In: Lee, J., Kim, J. (eds.) ICISC 2014. LNCS, vol. 8949, pp. 84–100. Springer, Switzerland (2014)
24. Kitagawa, F., Matsuda, T., Hanaoka, G., Tanaka, K.: Completeness of single-bit projection-KDM security for public key encryption. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 201–219. Springer, Heidelberg (2015)
25. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011)
26. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
27. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)
28. Shoup, V.: A proposal for an ISO standard for public key encryption. IACR Cryptology ePrint Archive 2001:112 (2001)

Functional Encryption

Extended Nested Dual System Groups, Revisited

Junqing Gong[1], Jie Chen[2(✉)], Xiaolei Dong[3], Zhenfu Cao[3(✉)], and Shaohua Tang[4]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
gongjunqing@126.com
[2] Shanghai Key Laboratory of Multidimensional Information Processing and Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai, China
S080001@e.ntu.edu.sg
[3] Shanghai Key Lab for Trustworthy Computing, East China Normal University, Shanghai, China
{dongxiaolei,zfcao}@sei.ecnu.edu.cn
[4] School of Computer Science and Engineering, South China University of Technology, Shanghai, China
shtang@IEEE.org

Abstract. The notion of extended nested dual system groups (ENDSG) was recently proposed by Hofheinz *et al.* [PKC 2015] for constructing almost-tight identity based encryptions (IBE) in the multi-instance, multi-ciphertext (MIMC) setting. However only a composite-order instantiation was proposed and more efficient prime-order instantiations are absent. The paper fills the blank by presenting two constructions.

We revise the definition of ENDSG and realize it using prime-order bilinear groups based on Chen and Wee's prime-order instantiation of nested dual system groups [CRYPTO 2013]. This yields the first almost-tight IBE in the prime-order setting achieving *weak* adaptive security in MIMC scenario under the d-linear (d-Lin) assumption. We further enhanced the revised ENDSG to capture stronger security notions for IBE, including *B-weak* adaptive security and *full* adaptive security. We show that our prime-order instantiation is readily B-weak adaptive secure and full adaptive secure without introducing extra assumption.

We then try to find better solutions by fine-tuning ENDSG again and realizing it using the technique of Chen, Gay, and Wee [EUROCRYPT 2015]. This leads to an almost-tight secure IBE in the same setting with better performance than our first result, but the security relies on a non-standard assumption, d-linear assumption with auxiliary input (d-LinAI) for an even positive integer d. However we note that, the 2-LinAI assumption is implied by the external decisional linear (XDLIN) assumption. This concrete instantiation could also be realized using *symmetric* bilinear groups under standard decisional linear assumption.

Keywords: Identity based encryptions · Dual system groups · Tight security · Security model · Prime-order bilinear groups

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 133–163, 2016.
DOI: 10.1007/978-3-662-49384-7_6

1 Introduction

Dual System Encryption. Recently we have witnessed a breakthrough of proof technique in the field of functional encryptions. In 2009, Waters [36] proposed a new proof paradigm for identity based encryptions (IBE), called *dual system technique*, and obtained the first adaptively secure IBE with short public key in the standard model whose security relies on a static assumption and the security loss is $O(q)$ where q is the number of key extraction queries. From a high-level view, the dual system technique works with two copies of some target cryptographic primitive such as IBE. The first copy is put into the so-called *normal space* and acts as the real system, while the second copy is put into the so-called *semi-functional space* and only used in the proof. Furthermore, the independence of the two spaces (say, orthogonality under pairing operations) allows us to make some changes in the semi-functional space for proof but still maintain the correctness in the normal space. It is worth noting that the new technique permits the simulator to reply all queries made by the adversary and avoids the security loss caused by the classical partitioning technique [10,12,35].

The revolution was then spreading across the field of functional encryptions. In particular, the dual system technique has been applied for establishing adaptive security of various types of functional encryptions, ranging from simple functionality, such as IBE [9,14–16,22,25,32] to expressive and complicated functionality, like ABE and IPE [5,7,13,16,26,27,31,37]. Some of them applied the dual system technique in a modular and abstract fashion such as Wee's predicate encoding [37] and Attrapadung's pairing encoding [5].

Almost-Tight Reduction. The dual system technique also helped us to go further. Chen and Wee [15] combined the dual system technique with the proof idea underlying the Naor-Reingold pseudorandom function [28] and achieved the first almost-tight IBE from a standard assumption in the standard model. The security loss is $O(n)$ where n is the length of identities, and unrelated to the number of key extraction queries anymore. They established the real system in the normal space and a mirror one in the semi-functional space for proof as the original dual system technique [36]. However, instead of dealing with key extraction queries (in the semi-functional space) separately as Waters [36], they handled all (i.e., q) secret keys as a whole in the next step following the proof strategy of Naor and Reingold [28]. In detail, we may imagine the master secret key as a truly random function taking identities as input. Starting from the original master secret key whose domain is just $\{\epsilon\}$, the proof argues that one can double the domain size until it reaches the size of the identity space if identities are encoded in a bit-by-bit fashion [35]. For identity space $\{0,1\}^n$, only n steps are required. Finally, the property of the random function allows us to information-theoretically hide the challenge message.

Recent work by Hofheinz *et al.* [21] extended Chen and Wee's result [15] and achieved almost tightness in the *multi-instance, multi-ciphertext* (MIMC) setting where the adversary simultaneously attacks multiple challenge identities in multiple IBE instances. In Chen and Wee's paradigm [15], the ith step that

increases the domain size from 2^{i-1} to 2^i can only handle the situation where all challenge ciphertexts share the same ith bit, which no longer holds in the MIMC setting. The proposed solution [21] is to further split the semi-functional space into two independent (in some sense) subspaces, labelled by $\wedge$ and $\sim$ respectively. The ith step starts from ciphertexts with $\wedge$-semi-functional component. They then move the semi-functional components in all ciphertexts for identities whose ith bit is 1 to the $\sim$-semi-functional space. At this moment, (1) in the $\wedge$-semi-functional space, all ciphertexts share the same ith bit 0; (2) in the $\sim$-semi-functional space, all ciphertexts share the same ith bit 1, which means that one can now applied Chen and Wee's proof strategy [15] in both subspaces separately.

We emphasize that achieving tight reduction, especially in the MIMC setting, is of practical importance. Consider a scenario involving λ instances and Q ciphertexts per instance. A trivial but generic transformation arises multiplicative $\mathcal{O}(\lambda Q)$ security loss where both λ and Q may be quite huge quantities, say 2^{30}. Therefore a large group should be employed to compensate the loss. This always leads to longer ciphertexts and lower encryption/decryption procedures.

Problem and Goal. Hofheinz *et al.* only provided an instantiation of the above proof strategy using *composite-order* bilinear groups [21]. Our goal is to realize a fully and almost-tightly secure IBE in the MIMC setting using *prime-order* bilinear groups. We emphasize that it is not just a theoretical interest to pursue such a solution. Most schemes (including [21]) using composite-order bilinear groups base their security on the *Subgroup Decision Assumption* [8] which implies the hardness of factoring the group order. This forces us to work with elliptic curve groups with quite large, say 1024 bits, base field when implementing the scheme. In contrast, for constructions in the prime-order setting, we could employ smaller base field, say 160 bits, without sacrificing the security. Although the construction now becomes complex in general, this still brings us a considerable advantage in both computation and space efficiency.

1.1 Motivation and Observation

Hofheinz *et al.*'s work [21] roughly follows the style of [15]. In particular, they first extended the notion of *Nested Dual System Groups* (NDSG) proposed by Chen and Wee [15], then proposed a general IBE construction from the extended NDSG (ENDSG) in the MIMC setting, and finally presented an instantiation of ENDSG using composite-order bilinear groups. Therefore it is sufficient for our purpose to realize ENDSG using prime-order bilinear groups and apply the general transformation in [21]. However we observe that their definition of ENDSG sets too strong requirements on algebraic structure of underlying groups, which makes it hard to be instantiated using existing techniques for prime-order bilinear groups.

An ENDSG describes a set of abstract groups with a bunch of structural and computational requirements supporting Hofheinz *et al.*'s proof strategy. We roughly recall[1] that an ENDSG defined in [21] consists of five algorithms: SampP,

[1] The notation is slightly different from [21].

SampG, SampH, $\widehat{\mathsf{SampG}}$, and $\widetilde{\mathsf{SampG}}$. Informally, the first algorithm generates a set of groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of order N (as well as other parameters) and the other four algorithms are used to sample random elements from some subgroup of $\mathbb{G}$ or $\mathbb{H}$ (which are associated with ciphertexts and secret keys, respectively, in the context of IBE). We emphasize that they required that

- Groups $\mathbb{G}$ and $\mathbb{H}$ are generated by some $g \in \mathbb{G}$ and $h \in \mathbb{H}$, respectively. (From the specification of group generator G.)
- "The outputs of SampG, $\widehat{\mathsf{SampG}}$, and $\widetilde{\mathsf{SampG}}$ are distributed uniformly over the generators of different nontrivial subgroups of $\mathbb{G}^{n+1}$ of coprime order, respectively." (From the G-subgroups.)

However, nearly all techniques realizing dual system technique in the prime-order setting employs vector spaces over $\mathbb{F}_p$ (for a prime p) to simulate group $\mathbb{G}$ and $\mathbb{H}$ [13,15,16,25,27,31]. Meanwhile subgroups of $\mathbb{G}$ and $\mathbb{H}$ are naturally simulated by its subspaces. Firstly, since a vector space is an additive group but not cyclic in general, neither $\mathbb{G}$ nor $\mathbb{H}$ is cyclic. Secondly, any d-dimensional subspace has p^d vectors, thus the orders of the outputs of SampG, $\widehat{\mathsf{SampG}}$, and $\widetilde{\mathsf{SampG}}$ must share a common factor p. In a word, techniques based on vector spaces by no means meets the requirements shown above.

Fortunately, we observe that both requirements are applied nowhere but to provide random self-reducibility of computational requirements (including LS1, LS2, NH) when they proved "ENDSG implies IBE". For example, the *Left Subgroup Indistinguishability 1* (LS1) said that, for any $(\mathrm{PP}, \mathrm{SP}) \leftarrow \mathsf{SampP}(k, n)$, the following two distributions are computationally indistinguishable.

$$\{\mathbf{g} : \mathbf{g} \leftarrow \mathsf{SampG}(\mathrm{PP})\} \quad \text{and} \quad \left\{\mathbf{g} \cdot \widehat{\mathbf{g}} : \mathbf{g} \leftarrow \mathsf{SampG}(\mathrm{PP}),\ \widehat{\mathbf{g}} \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})\right\}.$$

Given $\mathbf{T}$ which is either $\mathbf{g}$ or $\mathbf{g} \cdot \widehat{\mathbf{g}}$, the simulator (in the proof) can sample $s \leftarrow \mathbb{Z}_N^*$ and generate another independent problem instance $\mathbf{T}^s$ following the two requirements we have reviewed. We note that this property is crucial for achieving almost-tight reduction in the MIMC setting where the adversary is able to enquire more than one challenge ciphertext. This suggests that, if we adapt the ENDSG to support such random self-reducibility *explicitly*, it will still imply an IBE in MIMC setting and the limitations on underlying groups may be removed. As this happens, many existing techniques in the prime-order setting can now be applied to realize ENDSG and finally derive an almost-tight IBE in the MIMC setting using *prime-order* bilinear groups.

1.2 Contributions and Techniques

In this paper, we revise the definition of ENDSG, and show that the revised ENDSG not only almost-tightly implies an IBE in the MIMC setting but also can be *tightly* instantiated using prime-order bilinear groups. Putting them together, we obtain a fully and almost-tightly secure IBE in the same setting from prime-order bilinear groups. In particular, we proposed two instantiations: the first one

is proven secure under the d-linear assumption (d-Lin), while the second one is proven secure under a stronger assumption, d-linear assumption with auxiliary input, d-LinAI for short, but achieves shorter keys and ciphertexts.

Revisiting Extended Nested Dual System Groups. Our ENDSG is defined mainly in the spirit of [21] but with the difference that we provide (in requirements like LS1) enough independently-sampled subgroup elements directly instead of assuming some special algebraic structure. As an example, we define LS1 as: for any $(\mathrm{PP}, \mathrm{SP}) \leftarrow \mathsf{SampP}(k, n)$, the following two distributions are computationally indistinguishable.

$$\left\{ \{\mathbf{g}_j\}_{j\in[q]} : \mathbf{g}_j \leftarrow \mathsf{SampG}(\mathrm{PP}) \right\} \quad \text{and}$$
$$\left\{ \{\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j\}_{j\in[q]} : \mathbf{g}_j \leftarrow \mathsf{SampG}(\mathrm{PP}),\ \widehat{\mathbf{g}}_j \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP}) \right\}.$$

Here the parameter q depends on the number of challenge ciphertexts. This makes the definition more general and allows us to realize the notion using diverse algebra frameworks, especially prime-order bilinear groups. On the other hand, it still almost-tightly implies a fully secure IBE in the MIMC setting. The construction and the proof are nearly the same as [21].

To be fair, Hofheinz *et al.*'s definition is more convenient in the sense that any instantiation of ENDSG immediately results in an *almost-tight* IBE in the MIMC setting. In contrast, an instantiation of our definition with *loose* security reduction (say, with security loss $\mathcal{O}(q)$) clearly can not lead to *tightly secure* IBE. Hence, when working with our definition, we should not jump to the conclusion before checking the tightness. We also remark that we *do not* negate prime-order instantiations of Hofheinz *et al.*'s ENDSG.

Instantiation from d-Linear Assumption. We realize our revised ENDSG by extending the prime-order instantiation of NDSG by Chen and Wee [15]. The security only relies on the d-Lin assumption and the security loss is $\mathcal{O}(d)$ and independent of the number of samples, say q in the LS1 example, given to the adversary. By the generic construction [21], we obtain the first almost-tight IBE in the MIMC setting in the prime-order setting and fill the blank left in [21].

Technically, we extend the basis from $2d \times 2d$ matrix used in [15] to $3d \times 3d$ matrix in order to accommodate the additional semi-functional space. In detail, the first d-dimension subspace is the normal space, the next d-dimension subspace is the $\wedge$-semi-functional space, and the last d-dimension subspace is the $\sim$-semi-functional space.

The main challenge is to realize the *Left Subgroup Indistinguishability 2 (LS2)* property (c.f. Sect. 3). Roughly, we must prove that $\mathbf{g} \cdot \widehat{\mathbf{g}}$ (sampled from the normal space and $\wedge$-semi-functional space of $\mathbb{G}$) and $\mathbf{g} \cdot \widetilde{\mathbf{g}}$ (sampled from the normal space and $\sim$-semi-functional space of $\mathbb{G}$) are computationally indistinguishable even when the adversary can access to $\widehat{h}^* \cdot \widetilde{h}^* \in \mathbb{H}$ where $\widehat{h}^* \in \mathbb{H}$ is orthogonal to the normal and $\sim$-semi-functional space of $\mathbb{G}$ and $\widetilde{h}^* \in \mathbb{H}$ to the normal and $\wedge$-semi-functional space of $\mathbb{G}$. To simulate $\widehat{h}^* \cdot \widetilde{h}^*$, we further extend the subspace of $\widehat{h}^*$ and $\widetilde{h}^*$ from 1-dimension in [15] to d-dimension which

allows us to utilize the technique for proving *right subgroup indistinguishability* of Chen-Wee's prime-order instantiation of dual system groups [16]. So as to support this technical extension and conform to our revision, we model the process of sampling $\widehat{h}^*$ and $\widetilde{h}^*$ as two algorithms $\widehat{\mathsf{SampH}}^*$ and $\widetilde{\mathsf{SampH}}^*$ respectively, and give adversary adequate samples in related computational requirements. With such high-dimension $\widehat{h}^*$ and $\widetilde{h}^*$, the proof of *Nested-hiding Indistinguishability (NH)* (c.f. Sect. 3) will also be extended accordingly.

Achieving Stronger Security Guarantee. Hofheinz *et al.* [21] achieved weak security from their ENDSG where the adversary is allowed to make single challenge query for each identity in each instance. They introduced a variant of the BDDH assumption (s-BDDH) and proved the full security of their original construction where the above restriction on the adversary is removed. This additional computational requirement is realized under the *dual system bilinear DDH assumption* (DS-BDDH).

The revisions we have made do not involve the s-BDDH assumption, and the resulting ENDSG only leads to weak security. Motivated by and based on our prime-order instantiation, we investigate two flavors of stronger security: B*-weak* and *full adaptive security*. The former model allows adversary to make at most B challenge queries for each identity in each instance where B is a prior bound, while the latter one sets no limitation on the number of challenge queries on a single identity, i.e., polynomially many queries are allowed.

For each of them, we follow Hofheinz *et al.*'s workflow. Concretely, to achieve stronger security, we enhance the *non-degeneracy* property in our revised ENDSG and update the last step of Hofheinz *et al.*'s proof (decoupling challenge messages and ciphertexts) to make it sound in stronger models, where the non-degeneracy property is applied. We then prove that our instantiation of ENDSG under the d-Lin assumption (see Sect. 4) indeed satisfies the enhanced non-degeneracy property. The two results together imply an IBE with stronger security guarantee and almost-tight reduction in the MIMC setting. In particular,

1. We enhance the non-degenerate property to B*-bounded* version which states that the non-degeneracy property holds even when a single $\widehat{h}^*$ works with B $\widehat{g}_0$'s where B is a prior bound. It is easy to show that our instantiation under the d-Lin assumption is d-bounded non-degenerated unconditionally.
2. We enhance the non-degeneracy property to *computational* version which is essentially similar to the s-BDDH assumption [21] and states that the non-degeneracy property holds even when a single $\widehat{h}^*$ works with *polynomially many* $\widehat{g}_0$'s. Luckily, we can prove that our instantiation is computationally non-degenerated under the d-Lin assumption, and no additional assumption is required.

Towards More Efficient Instantiation. Having obtained the first construction, we continue to purse more efficient solutions. The main idea is to reduce the dimensions of two semi-functional spaces. However this forces us to base the security on a non-standard assumption, d-LinAI assumption (c.f. Sect. 7) for *an*

even positive integer d. We argue that the concrete assumption with $d = 2$ is implied by the classical external decision linear assumption (XDLIN) [1]. We give an overview of our method and the resulting IBE scheme in Sect. 7. All details are given in the full version of the paper.

1.3 Comparison and Discussion

We make a comparison among existing almost-tightly secure IBE schemes in the MIMC setting in terms of time and space efficiency. The details are shown in Table 1. Our comparison involves the composite-order construction by Hofheinz *et al.* [21], the prime-order construction in Sect. 5 based on the decisional linear (DLIN, 2-Lin) and symmetric external Diffie-Hellman (SXDH, 1-Lin) assumption, and the prime-order construction from Sect. 7 based on the XDLIN (2-LinAI) assumption. As a base line, we also consider the efficiency of prime-order construction by Chen and Wee [15] and Blazy *et al.* [9], which is *not* built for the MIMC setting.

Hofheinz *et al.*'s construction (see the third row) works with a symmetric bilinear group whose order is the product of four distinct primes, the sizes of group elements are much larger, and exponentiation and pairing operations are much more expensive. Therefore the overall efficiency is not acceptable even

Table 1. Comparing Efficiency among existing and proposed almost-tight IBE schemes. n is the length of identities. Column $|\mathsf{MPK}|$, $|\mathsf{SK}|$, and $|\mathsf{CT}|$ show the size of master public keys, user's secret keys and ciphertexts, respectively. Each sub-column contains the number of elements in G, G_1, G_2, and G_T. Column T_{Enc} and T_{Dec} show encryption and decryption cost, respectively. Each sub-column E, E_1, and E_T shows the number of exponentiations on group G, G_1, and G_T, respectively, and sub-column P shows the number of pairings. Column "Assum." shows the underlying assumption. "Static" means static assumptions in the composite-order bilinear group. Column "$|G|$" indicates the group order, "P" for prime and "C" for composite order, respectively.

Scheme	$\lvert G\rvert$	Assum.	$\lvert\mathsf{MPK}\rvert$		$\lvert\mathsf{SK}\rvert$	$\lvert\mathsf{CT}\rvert$		T_{Enc}		T_{Dec}	MIMC
			G_1/G	G_T	G_2/G	G_1/G	G_T	E_1/E	E_T	P	
[15]	P	d-Lin	$2d^2(2n+1)$	d	$4d$	$4d$	1	$4d^2$	d	$4d$	✗
		DLIN	$16n+8$	2	8	8	1	16	2	8	
		SXDH	$4n+2$	1	4	4	1	4	1	4	
[9]	P	d-Lin	$(2n+1)d^2+d$	d	$2d+1$	$2d+1$	1	$2d^2+1$	d	$2d+1$	✗
		DLIN	$8n+6$	2	5	5	1	9	2	5	
		SXDH	$2n+2$	1	3	3	1	3	1	3	
[21]	C	Static	$2n+1$	1	2	2	1	2	1	2	✔
Sec. 5	P	d-Lin	$3d^2(2n+1)$	d	$6d$	$6d$	1	$6d^2$	d	$6d$	✔
		DLIN	$24n+12$	2	12	12	1	24	2	12	
		SXDH	$6n+3$	1	6	6	1	6	1	6	
Sec. 7	P	d-LinAI	$2d^2(2n+1)$	d	$4d$	$4d$	1	$4d^2$	d	$4d$	✔
		XDLIN	$16n+8$	2	8	8	1	16	2	8	

though the numbers of group elements in MSK, SK and CT are smaller and Enc and Dec involve less exponentiation and pairing operations.

When instantiating our first proposal (see the fourth row) under the DLIN assumption, each group element in $\mathbb{G}$ and $\mathbb{H}$ is a 6-dimension vector over G_1 and G_2, respectively, where G_1 and G_2 are source groups of a prime-order bilinear group. When instantiating under the SXDH assumption, each group element in $\mathbb{G}$ and $\mathbb{H}$ is a 3-dimension vector over G_1 and G_2, respectively. Compared with Blazy *et al.*'s construction [9], both size of MPK, SK and CT and cost of Enc and Dec are (at least) doubled in our construction. On the other hand, in our second instantiation based on the XDLIN assumption (see the last row), each group element in $\mathbb{G}$ and $\mathbb{H}$ is a vector of 4-dimension over G. Although the resulting IBE is still less efficient than Blazy *et al.*'s construction [9] under the DLIN assumption, the stronger computational assumption (i.e., XDLIN) helps us to narrow the gap. We may view this as a tradeoff between strength of security and efficiency without changing the security model. We leave it as an open problem to find more efficient fully secure IBE with tight reduction in the MIMC setting, especially from standard d-Lin assumption.

1.4 Related Work

Dual System Groups and Its Variants. Chen and Wee proposed the notion of dual system groups [16], which captures key algebraic structure supporting the dual system technique. They used this abstract primitive to obtain an HIBE scheme with constant-size ciphertexts using prime-order bilinear groups. The nested dual system group, an variant of dual system groups, was proposed by Chen and Wee [15] to reach almost-tight adaptively secure IBE in the standard model. Recently, the dual system group had been combined with the predicate/pairing encoding [2,13] and led to a lot of functional encryptions in the prime-order setting. Very recent work by Gong *et al.* [20] extended the concept of dual system groups to build an unbounded HIBE [24,25] with shorter ciphertexts in the prime-order setting.

Identity Based Encryption. The notion of identity based encryptions was introduced by Shamir [33] in 1984. The first practical realization was proposed by Boneh and Franklin [12] using bilinear groups and Cocks [17] using quadratic residue. Both of them rely on the heuristic random oracle model. Before Waters proposed his seminal work, there were several classical and practical solutions in the standard model, including Boneh-Boyen's IBE [10,11], Waters' IBE [35], and Gentry's IBE [18]. IBE can also be realized using algebra frameworks other than bilinear groups, such as lattices [3,4,19].

1.5 Independent Work

The independent work by Attrapadung, Hanaoka, and Yamada [6] also involves several constructions of almost-tight IBE in the MIMC setting. They developed an elegant framework for building almost-tight IBE in the MIMC setting from

the so-called *broadcast encoding*, which is a special form of Attrapadung's pairing encoding [5], and obtained a series of concrete schemes with various properties (including sub-linear size master public key and anonymous version) using both composite-order and prime-order bilinear groups. Their results and ours partially overlap. Their scheme with constant-size ciphertext in prime-order group (i.e., $\Phi_{\mathsf{cc}}^{\mathsf{prime}}$) is similar to our second construction based on the XDLIN assumption shown in Sect. 7. In fact, they share the same performance in terms of the size of ciphertexts and secret keys and running time of Enc and Dec. However we note that we also provide an generalization of this construction but proven secure under the non-standard d-LinAI assumption. Furthermore, our first construction in Sect. 5 is full-adaptively secure under the *standard* d-Lin assumption, and derives a SXDH-based concrete scheme, which has the best (space and time) performance among all proposed solutions so far.

Outline. Section 2 presents necessary background. Section 3 gives our revised definition of ENDSG. We realize our revised ENDSG in the prime-order setting in Sect. 4 and investigate how to update our ENDSG and its prime-order instantiation to achieve higher security level in Sect. 6. At last, Sect. 7 is an overview of obtaining a more efficient solution.

2 Preliminaries

2.1 Notations

For a finite set S, we use $s \leftarrow S$ to denote the process of picking s from S at random. For any $n \in \mathbb{Z}^+$, we take $[n]$ as the brief representation of set $\{1, \ldots, n\}$. For a probabilistic algorithm Alg and an fixed input x, we use $[\mathsf{Alg}(x)]$ to indicate the set of all possible outputs of algorithm Alg on input x. "p.p.t." stands for "probabilistic polynomial time". We let $\mathbf{e}_i$ denote the vector with 1 on the ith position and 0 elsewhere. For a group G and $g \in G$, let $h^{\mathbf{e}_i}$ be a vector over G with h on the ith position and 1 elsewhere. For two vectors $\mathbf{g} := (g_1, \ldots, g_n) \in G^n$ and $\mathbf{g}' := (g'_1, \ldots, g'_n) \in G^n$, we define $\mathbf{g} \cdot \mathbf{g}' = (g_1 \cdot g'_1, \ldots, g_n \cdot g'_n) \in G^n$ where "$\cdot$" on the right-hand side is the group operation of G. For any vector $\mathbf{x} = (x_1, \ldots, x_n)$ and $i \in [n]$, we define $\mathbf{x}_{-i}$ as a vector $(x_1, \ldots, x_{i-1}, \perp, x_{i+1}, \ldots, x_n)$ whose ith position is unknown (we take $\perp$ as a placeholder).

2.2 Identity Based Encryptions

Algorithms. An IBE scheme in the multi-instance setting consists of five p.p.t. algorithms defined as follows[2]. (1) The *parameter generation algorithm* $\mathsf{Param}(1^k, \mathrm{SYS})$ takes as input a security parameter $k \in \mathbb{Z}^+$ in its unary form and a system-level parameter SYS, and outputs a global parameter GP. (2) The *setup algorithm* $\mathsf{Setup}(\mathrm{GP})$ takes as input a global parameter GP, and outputs

[2] The definition shown here is slightly different from that in [21]. The adaptation is purely conceptual and made for clarity. The security model is tuned accordingly.

a master public/secret key pair $(\mathrm{MPK}, \mathrm{MSK})$. (3) The *key generation algorithm* $\mathsf{KeyGen}(\mathrm{MPK}, \mathrm{MSK}, \mathbf{y})$ takes as input a master public key MPK, a master secret key MSK and an identity $\mathbf{y}$, and outputs a secret key $\mathrm{SK}_{\mathbf{y}}$ for the identity. (4) The *encryption algorithm* $\mathsf{Enc}(\mathrm{MPK}, \mathbf{x}, \mathrm{M})$ takes as input a master public key MPK, an identity $\mathbf{x}$ and a message M, outputs a ciphertext $\mathrm{CT}_{\mathbf{x}}$ for the message under the identity. (5) The *decryption algorithm* $\mathsf{Dec}(\mathrm{MPK}, \mathrm{SK}, \mathrm{CT})$ takes as input a master public key MPK, a secret key SK and a ciphertext CT, outputs a message M or a failure symbol $\perp$.

The so-called "multi-instance setting" indicates that we are considering a collection of IBE instances established under the same global parameter GP. We leave the system-level parameter SYS undefined for generality. It may depend on concrete constructions or application scenarios.

Correctness. For any parameter $k \in \mathbb{Z}^+$, any SYS, any $\mathrm{GP} \in [\mathsf{Param}(1^k, \mathrm{SYS})]$, any $(\mathrm{MPK}, \mathrm{MSK}) \in [\mathsf{Setup}(\mathrm{GP})]$, any identity $\mathbf{x}$, and any message M, it holds that

$$\Pr\left[\mathsf{Dec}(\mathrm{MPK}, \mathsf{KeyGen}(\mathrm{MPK}, \mathrm{MSK}, \mathbf{x}), \mathsf{Enc}(\mathrm{MPK}, \mathbf{x}, \mathrm{M})) = \mathrm{M}\right] \geqslant 1 - 2^{-\Omega(k)}.$$

The probability space is defined by the random coins consumed by algorithm KeyGen and Enc.

Adaptive Security in the Multi-instance, Multi-ciphertext Setting. Roughly, the adaptive security in the multi-instance, multi-ciphertext setting extends the traditional adaptive security model for IBE [12] in the sense that the adversary can access to multiple IBE instances (obtaining master public key and users' keys) and attack multiple ciphertexts (i.e., challenge ciphertexts), which is formalized by Hofheinz *et al.* [21]. Ideally, the adversary is free to choose the challenge instance, the challenge identity and the challenge message pair. Hofheinz *et al.* [21] also identified a weaker variant in which only one challenge ciphertext is allowed for each challenge identity in each challenge instance, and called the ideal one *full security*.

We review the experiment $\mathbf{Exp}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_R)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ [21], which captures both the weaker and full security notion.

Setup. $\mathcal{C}$ gets $\mathrm{GP} \leftarrow \mathsf{Param}(1^k, \mathrm{SYS})$ and creates $(\mathrm{MPK}_\iota, \mathrm{MSK}_\iota) \leftarrow \mathsf{Setup}(\mathrm{GP})$ for $\iota \in [\lambda]$. All master public keys $\{\mathrm{MPK}_\iota\}_{\iota \in [\lambda]}$ are sent to $\mathcal{A}$. $\mathcal{C}$ also chooses a secret random bit $\beta \in \{0, 1\}$ and initializes Q_K and Q_C as empty sets.

Query. $\mathcal{A}$ is allowed to make two types of queries: key extraction queries and challenge queries. $\mathcal{C}$ answers every queries as follows: (1) For each key extraction query $(\iota, \mathbf{y})$, $\mathcal{C}$ returns $\mathrm{SK} \leftarrow \mathsf{KeyGen}(\mathrm{MPK}_\iota, \mathrm{MSK}_\iota, \mathbf{y})$ and updates $Q_K := Q_K \cup \{(\iota, \mathbf{y})\}$. (2) For each challenge query $(\iota^*, \mathbf{x}^*, \mathrm{M}_0^*, \mathrm{M}_1^*)$, $\mathcal{C}$ returns $\mathrm{CT}^* \leftarrow \mathsf{Enc}(\mathrm{MPK}_{\iota^*}, \mathbf{x}^*, \mathrm{M}_\beta^*)$ and updates $Q_C := Q_C \cup \{(\iota^*, \mathbf{x}^*)\}$.

Guess. $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$.

We say an adversary $\mathcal{A}$ wins experiment $\mathbf{Exp}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_R)$, denoted by $\mathbf{Exp}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_R) = 1$, if and only if (1) $\beta = \beta'$, (2) $Q_K \cap Q_C = \emptyset$, (3) $\mathcal{A}$ made at most q_K key extraction queries, (4) there are at most q_C challenge

identities, and (5) for each of them, there exist at most q_R challenge ciphertexts. We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_R) = \left| \Pr[\mathbf{Exp}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_R) = 1] - 1/2 \right|.$$

The probability space is defined by random coins consumed by both $\mathcal{C}$ and $\mathcal{A}$. An IBE is (λ, q_K, q_C, q_R)-*adaptively-secure* if, for any p.p.t. adversary $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, q_K)$ is bounded by $2^{-\Omega(k)}$. Clearly, the (λ, q_k, q_C, q_R)-adaptive security with unbounded q_R is consistent with the full security, while the $(\lambda, q_k, q_C, 1)$-adaptive security is exactly the weak security. Furthermore, we define B-*weak adaptive security*, an intermediate security notion between them, as (λ, q_K, q_C, B)-adaptive security for a priori bound $B \geqslant 1$.

3 Revisiting Extended Nested Dual System Groups

This section revises the ENDSG proposed by Hofheinz *et al.* [21]. Following the intuitive discussion in Sect. 1, the key points are: we (1) remove special group requirements, (2) explicitly provide samples in each computational assumption, (3) generalize subgroup of $\widehat{h}^*$ and $\widetilde{h}^*$. We show our definition followed by a series of remarks clarifying motivations behind several technical decisions.

Syntax. Our revised ENDSG consists of eight p.p.t. algorithms as follows:

- $\mathsf{SampP}(1^k, n)$: Output: (1) PP containing (a) group description $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ and an admissible bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$; (b) an efficient linear map μ defined on $\mathbb{H}$; (c) an efficient sampler for $\mathbb{H}$ and $\mathbb{Z}_{\mathrm{ord}(\mathbb{H})}$, respectively; (d) public parameters for SampG and SampH. (2) SP containing secret parameters for $\widehat{\mathsf{SampG}}$, $\widetilde{\mathsf{SampG}}$, $\widehat{\mathsf{SampH}}^*$ and $\widetilde{\mathsf{SampH}}^*$.
- SampGT: $\mathrm{Im}(\mu) \to \mathbb{G}_T$.
- $\mathsf{SampG}(\mathrm{PP})$: Output $\mathbf{g} = (g_0, g_1, \ldots, g_n) \in \mathbb{G}^{n+1}$.
- $\mathsf{SampH}(\mathrm{PP})$: Output $\mathbf{h} = (h_0, h_1, \ldots, h_n) \in \mathbb{H}^{n+1}$.
- $\widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$: Output $\widehat{\mathbf{g}} = (\widehat{g}_0, \widehat{g}_1, \ldots, \widehat{g}_n) \in \mathbb{G}^{n+1}$.
- $\widetilde{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$: Output $\widetilde{\mathbf{g}} = (\widetilde{g}_0, \widetilde{g}_1, \ldots, \widetilde{g}_n) \in \mathbb{G}^{n+1}$.
- $\widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$: Output $\widehat{h}^* \in \mathbb{H}$.
- $\widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$: Output $\widetilde{h}^* \in \mathbb{H}$.

The first four algorithms are used in the real system, while the remaining ones are defined for the proof. We let SampG_0 refer to the first element in the output of SampG, i.e., g_0. The notation also applies to SampH, $\widehat{\mathsf{SampG}}$, and $\widetilde{\mathsf{SampG}}$.

Correctness. For all $k, n \in \mathbb{Z}^+$ and all $(\mathrm{PP}, \mathrm{SP}) \in [\mathsf{SampP}(1^k, n)]$, we require

(Projective.) For all $h \in \mathbb{H}$ and all possible random coins s, $\mathsf{SampGT}(\mu(h); s) = e(\mathsf{SampG}_0(\mathrm{PP}; s), h)$.

(Associative.) For all $(g_0, g_1, \ldots, g_n) \in [\mathsf{SampG}(\mathrm{PP})]$ and all $(h_0, h_1, \ldots, h_n) \in [\mathsf{SampH}(\mathrm{PP})]$, $e(g_0, h_i) = e(g_i, h_0)$ for $i \in [n]$.

Security. For all $k, n \in \mathbb{Z}^+$ and all $(\mathrm{PP}, \mathrm{SP}) \in [\mathsf{SampP}(1^k, n)]$, we require

(Orthogonality.) For all $\widehat{h}^* \in [\widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})]$ and all $\widetilde{h}^* \in [\widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})]$,
1. $\mu(\widehat{h}^*) = \mu(\widetilde{h}^*) = 1$;
2. $e(\widehat{g}_0, \widetilde{h}^*) = 1$ for all $\widehat{g}_0 \in [\widehat{\mathsf{SampG}}_0(\mathrm{PP}, \mathrm{SP})]$;
3. $e(\widetilde{g}_0, \widehat{h}^*) = 1$ for all $\widetilde{g}_0 \in [\widetilde{\mathsf{SampG}}_0(\mathrm{PP}, \mathrm{SP})]$;

The first requirement implies that $e(g_0, \widetilde{h}^*) = e(g_0, \widehat{h}^*) = 1$ for all $g_0 \in [\mathsf{SampG}_0(\mathrm{PP})]$ by the *projective* property (c.f. Sect. 3.2 in [15]).

(Non-degeneracy.) Over the probability space defined by $\widehat{g}_0 \leftarrow \widehat{\mathsf{SampG}}_0(\mathrm{PP}, \mathrm{SP})$, with overwhelming probability $1 - 2^{-\Omega(k)}$, $e(\widehat{g}_0, \widehat{h}^*)$ is distributed uniformly over $\mathbb{G}_T$ when sampling $\widehat{h}^* \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$.

(**$\mathbb{H}$-subgroup.)** The output of $\mathsf{SampH}(\mathrm{PP})$ is distributed uniformly over some subgroup of $\mathbb{H}^{n+1}$, while those of $\widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$ and $\widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$ are distributed uniformly over some subgroup of $\mathbb{H}$, respectively.

(Left subgroup indistinguishability 1 (LS1).) For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}^{\mathrm{LS1}}_{\mathcal{A}}(k, q) := \left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right|,$$

where

$$D := (\mathrm{PP}),\ T_0 := \{\mathbf{g}_j\}_{j\in[q]},\ T_1 := \left\{\mathbf{g}_j \cdot \boxed{\widehat{\mathbf{g}}_j}\right\}_{j\in[q]}$$

and $\mathbf{g}_j \leftarrow \mathsf{SampG}(\mathrm{PP})$ and $\widehat{\mathbf{g}}_j \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$.

(Left subgroup indistinguishability 2 (LS2).) For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}^{\mathrm{LS2}}_{\mathcal{A}}(k, q, q') := \left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right|,$$

where

$$D := \left(\mathrm{PP}, \left\{\widehat{h}^*_j \cdot \widetilde{h}^*_j\right\}_{j\in[q+q']}, \left\{\mathbf{g}'_j \cdot \widehat{\mathbf{g}}'_j\right\}_{j\in[q]}\right),$$

$$T_0 := \{\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j\}_{j\in[q]},\ T_1 := \left\{\mathbf{g}_j \cdot \boxed{\widetilde{\mathbf{g}}_j}\right\}_{j\in[q]}$$

and $\widehat{h}^*_j \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$, $\widetilde{h}^*_j \leftarrow \widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$, $\mathbf{g}'_j \leftarrow \mathsf{SampG}(\mathrm{PP})$, $\widehat{\mathbf{g}}'_j \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$, $\mathbf{g}_j \leftarrow \mathsf{SampG}(\mathrm{PP})$, $\widehat{\mathbf{g}}_j \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$, $\widetilde{\mathbf{g}}_j \leftarrow \widetilde{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$.

(Nested-hiding indistinguishability (NH).) For any $\eta \in [\lfloor n/2 \rfloor]$ and any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}^{\mathrm{NH}(\eta)}_{\mathcal{A}}(k, q, q') := \left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right|,$$

where

$$D := \left(\mathrm{PP}, \left\{\widehat{h}^*_j\right\}_{j\in[q+q']}, \left\{\widetilde{h}^*_j\right\}_{j\in[q+q']}, \left\{(\widehat{\mathbf{g}}_j)_{-(2\eta-1)}\right\}_{j\in[q]}, \left\{(\widetilde{\mathbf{g}}_j)_{-2\eta}\right\}_{j\in[q]}\right),$$

$$T_0 := \{\mathbf{h}_j\}_{j\in[q']},\ T_1 := \left\{\mathbf{h}_j \cdot \boxed{(\widehat{h}_j^{**})^{\mathbf{e}_{2\eta-1}} \cdot (\widetilde{h}_j^{**})^{\mathbf{e}_{2\eta}}}\right\}_{j\in[q']}$$

and $\widehat{h}_j^* \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP},\mathrm{SP})$, $\widetilde{h}_j^* \leftarrow \widetilde{\mathsf{SampH}}^*(\mathrm{PP},\mathrm{SP})$, $\widehat{\mathbf{g}}_j \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP},\mathrm{SP})$, $\widetilde{\mathbf{g}}_j \leftarrow \widetilde{\mathsf{SampG}}(\mathrm{PP},\mathrm{SP})$, $\mathbf{h}_j \leftarrow \mathsf{SampH}(\mathrm{PP})$, $\widehat{h}_j^{**} \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP},\mathrm{SP})$, $\widetilde{h}_j^{**} \leftarrow \widetilde{\mathsf{SampH}}^*(\mathrm{PP},\mathrm{SP})$. We let $\mathsf{Adv}^{\mathrm{NH}}_{\mathcal{A}}(k,q,q') := \max_{\eta\in[\lfloor n/2\rfloor]}\left\{\mathsf{Adv}^{\mathrm{NH}(\eta)}_{\mathcal{A}}(k,q,q')\right\}$.

Remark 1 (notations). ENDSG is mainly defined for building IBE. We remark that, in the description of LS1, LS2, and NH, the parameter q and q' roughly correspond to the maximum number of challenge queries and key extraction queries, respectively.

Remark 2 (sampling $\widehat{h}^*$ and $\widetilde{h}^*$, and $\mathbb{H}$-subgroup). We model the process of sampling over subgroup generated by $\widehat{h}^*$ and $\widetilde{h}^*$ (in [21]) as algorithm $\widehat{\mathsf{SampH}}^*$ and $\widetilde{\mathsf{SampH}}^*$, respectively. This allows us to employ more complex algebraic structure (say, subspaces of higher dimensions), which is crucial for our prime-order instantiation in Sect. 4. Accordingly, we extend $\mathbb{H}$-subgroup property to take $\widehat{\mathsf{SampH}}^*$ and $\widetilde{\mathsf{SampH}}^*$ into account.

Remark 3 ($\mathbb{G}$-subgroup and $\mathbb{H}$-subgroup). Since we provide adequate samples of $\mathbb{G}^{n+1}$ directly in the last three computational security requirements and further re-randomization is not necessary in the proof, the $\mathbb{G}$-subgroup in the original definition could be safely removed. However this won't let the revised ENDSG free from $\mathbb{H}$-subgroup property. The simulator still need the property to re-randomize T_0 or T_1 in $\mathrm{NH}(\eta)$ using $\mathsf{SampH}(\mathrm{PP})$ to maintain the consistency of truly random functions on two identities sharing the same η-bit prefix.

On one hand, our revised definition for ENDSG is essentially consistent with Hofheinz *et al.*'s definition [21]. In particular, it is not hard to see that one may use Hofheinz *et al.*'s ENDSG [21] to realize this revised version. Therefore their instantiation using composite-order bilinear groups can also be taken as an instantiation of the revised version above. On the other hand, our revised definition still almost-tightly implies an IBE in the MIMC setting. In fact, the construction, the security result and its proof are nearly the same as those presented in [21]. One may consider them as rewriting Hofheinz *et al.*'s results [21] in the language of our revised ENDSG. We present the construction and sketch of the proof in the full version of the paper. It is worth noting that the construction only achieves weak adaptive security. We will show how to enhance *non-degeneracy* to reach full adaptive security in Sect. 6.

4 Instantiating ENDSG from d-Linear Assumption

This section gives an instantiation of our revised ENDSG (defined in Sect. 3) using prime-order bilinear groups. See Sect. 1 for more motivation.

4.1 Prime-Order Bilinear Groups and Computational Assumptions

A prime-order bilinear group generator $\mathsf{GrpGen}(1^k)$ takes security parameter 1^k as input and outputs $\mathcal{G} := (p, G_1, G_2, G_T, e)$, where G_1, G_2 and G_T are finite cyclic groups of prime order p, and $e : G_1 \times G_2 \to G_T$ is a non-degenerated and efficiently computable bilinear map. We let g_1, g_2 and $g_T := e(g_1, g_2)$ be a generator of G_1, G_2 and G_T, respectively. We state the (standard) d-linear assumption (d-Lin) in G_1 (see Assumption 1), the analogous assumption in G_2 can be defined by exchanging the role of G_1 and G_2.

Assumption 1 (d-Linear Assumption in G_1). For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}_{\mathcal{A}}^{d\text{-Lin}}(k) := \left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right|,$$

where

$$D := \left(\mathcal{G}, g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 s_1}, \ldots, g_1^{a_d s_d}\right),$$

$$T_0 := g_1^{a_{d+1}(s_1+\cdots+s_d)},\ T_1 := g_1^{a_{d+1}(s_1+\cdots+s_d)+\boxed{s_{d+1}}}$$

and $\mathcal{G} \leftarrow \mathsf{GrpGen}(1^k)$, $a_1, \ldots, a_d, a_{d+1}, s_{d+1} \leftarrow \mathbb{Z}_p^*$ and $s_1, \ldots, s_d \leftarrow \mathbb{Z}_p$.

"Matrix-in-the-exponent" Notation. For an $m \times n$ matrix $\mathbf{X} = (x_{i,j})$ over $\mathbb{Z}_p$ and a group element g of G, we define $g^{\mathbf{X}} := (g^{x_{i,j}})$, an $m \times n$ matrix over G. We extend pairing e as: given two matrices $\mathbf{A} \in \mathbb{Z}_p^{t\times m}$ and $\mathbf{B} \in \mathbb{Z}_p^{t\times n}$, we define $e(g_1^{\mathbf{A}}, g_2^{\mathbf{B}}) := e(g_1, g_2)^{\mathbf{A}^\top \mathbf{B}} \in G_T^{m\times n}$. For vectors $\mathbf{x}$ and $\mathbf{y}$ over $\mathbb{Z}_p$ of the same length, we have $e(g_1^{\mathbf{x}}, g_2^{\mathbf{y}}) := e(g_1, g_2)^{\mathbf{x}^\top \mathbf{y}} \in G_T$, the standard inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ in the exponent. We will use $\mathbf{0}$ to denote both vectors and matrices with only zero entries, and give out its dimension or size in the subscript if necessary.

An Extended Version of d-Lifted Linear Assumption. We describe an extension of the d-Lifted Linear (d-LLin) assumption [23] for improving the readability of our proofs, which is called (d, ℓ, q)-Lifted Linear ((d, ℓ, q)-LLin) Assumption. We present the assumption in G_1 and the counterpart in G_2 is readily derived. We then give Lemma 1 showing that the (d, ℓ, q)-LLin assumption is tightly implied by the d-Lin assumption following [15,23]. The proof could be found in the full version of the paper. We remark that, since ℓ corresponds to a relatively small parameter, say 2, in our construction and q corresponds to the amount of adversary's queries which may be 2^{30}, we prove the Lemma under the assumption that $\ell < q$ for simplicity.

Assumption 2 ((d, ℓ, q)-Lifted Linear Assumption in G_1). For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}_{\mathcal{A}}^{(d,\ell,q)\text{-LLin}}(k) := \left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right|,$$

where

$$D := \left(\mathcal{G}, g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, \left\{g_1^{b_{i,j}}\right\}_{i\in[\ell], j\in[d]}, \left\{g_1^{a_1 s_{1,j}}, \ldots, g_1^{a_d s_{d,j}}\right\}_{j\in[q]}\right),$$

$$T_0 := \left\{ g_1^{b_{i,1}s_{1,j}+\cdots+b_{i,d}s_{d,j}} \right\}_{i\in[\ell],j\in[q]},\ T_1 := \left\{ g_1^{b_{i,1}s_{1,j}+\cdots+b_{i,d}s_{d,j}+\boxed{s_{d+i,j}}} \right\}_{i\in[\ell],j\in[q]}$$

and $\mathcal{G} \leftarrow \mathsf{GrpGen}(1^k)$, $a_1, \ldots, a_d, b_{i,j}, s_{d+i,j} \leftarrow \mathbb{Z}_p^*$, $s_{1,j}, \ldots, s_{d,j} \leftarrow \mathbb{Z}_p$.

Lemma 1 (d-Lin $\Rightarrow$ (d,ℓ,q)-LLin). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{(d,\ell,q)\text{-LLin}}_{\mathcal{A}}(k) \leqslant \ell \cdot \mathsf{Adv}^{d\text{-Lin}}_{\mathcal{B}}(k) + 1/(p-1),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + \ell^2 d \cdot \mathsf{poly}(k)$ *where* $\mathsf{poly}(k)$ *is independent of* $\mathsf{Time}(\mathcal{A})$.

4.2 Construction

We let $\pi_{\mathrm{L}}(\cdot)$, $\pi_{\mathrm{M}}(\cdot)$, and $\pi_{\mathrm{R}}(\cdot)$ be functions mapping from a $3d \times 3d$ matrix to its left-most d columns, its middle d columns, and its right-most d columns, respectively. Algorithms of our revised ENDSG are shown as follows.

- $\mathsf{SampP}(1^k, n)$: Run $(p, G_1, G_2, G_T, e) \leftarrow \mathsf{GrpGen}(1^k)$ and set $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e) := (G_1^{3d}, G_2^{3d}, G_T, e)$. Sample $\mathbf{B}, \mathbf{R} \leftarrow \mathrm{GL}_{3d}(\mathbb{Z}_p)$ and $\mathbf{A}_1, \ldots, \mathbf{A}_n \leftarrow \mathbb{Z}_p^{3d\times 3d}$. Set $\mathbf{B}^* := (\mathbf{B}^{-1})^\top$. Define

$$\begin{array}{l} \mathbf{D} := \pi_{\mathrm{L}}(\mathbf{B}),\ \mathbf{D}_i = \pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_i);\ \mathbf{E} := \pi_{\mathrm{M}}(\mathbf{B}),\ \mathbf{E}_i = \pi_{\mathrm{M}}(\mathbf{B}\mathbf{A}_i); \\ \mathbf{D}^* := \mathbf{B}^*\mathbf{R},\ \mathbf{D}_i^* = \mathbf{B}^*\mathbf{A}_i^\top\mathbf{R};\ \mathbf{F} := \pi_{\mathrm{R}}(\mathbf{B}),\ \mathbf{F}_i = \pi_{\mathrm{R}}(\mathbf{B}\mathbf{A}_i); \end{array}$$

  for $i \in [n]$. Define $\mu(g_2^{\mathbf{k}}) := e(g_1^{\mathbf{D}}, g_2^{\mathbf{k}}) = e(g_1, g_2)^{\mathbf{D}^\top\mathbf{k}}$ for all $\mathbf{k} \in \mathbb{Z}_p^{3d}$. Output

$$\mathrm{PP} := \begin{pmatrix} g_1^{\mathbf{D}},\ g_1^{\mathbf{D}_1}, \ldots, g_1^{\mathbf{D}_n} \\ g_2^{\mathbf{D}^*}, g_2^{\mathbf{D}_1^*}, \ldots, g_2^{\mathbf{D}_n^*} \end{pmatrix} \text{ and } \mathrm{SP} := \begin{pmatrix} g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)}, g_1^{\mathbf{E}}, g_1^{\mathbf{E}_1}, \ldots, g_1^{\mathbf{E}_n} \\ g_2^{\pi_{\mathrm{R}}(\mathbf{B}^*)}, g_1^{\mathbf{F}}, g_1^{\mathbf{F}_1}, \ldots, g_1^{\mathbf{F}_n} \end{pmatrix}.$$

  We assume PP always contains $\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, \mu$ and group order p.
- $\mathsf{SampGT}(g_T^{\mathbf{p}})$: Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output $g_T^{\mathbf{s}^\top\mathbf{p}} \in G_T$.
- $\mathsf{SampG}(\mathrm{PP})$: Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output $\left(g_1^{\mathbf{D}\mathbf{s}}, g_1^{\mathbf{D}_1\mathbf{s}}, \ldots, g_1^{\mathbf{D}_n\mathbf{s}}\right) \in (G_1^{3d})^{n+1}$.
- $\mathsf{SampH}(\mathrm{PP})$: Sample $\mathbf{r} \leftarrow \mathbb{Z}_p^{3d}$ and output $\left(g_2^{\mathbf{D}^*\mathbf{r}}, g_2^{\mathbf{D}_1^*\mathbf{r}}, \ldots, g_2^{\mathbf{D}_n^*\mathbf{r}}\right) \in (G_2^{3d})^{n+1}$.
- $\widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$: Sample $\widehat{\mathbf{s}} \leftarrow \mathbb{Z}_p^d$ and output $\left(g_1^{\mathbf{E}\widehat{\mathbf{s}}}, g_1^{\mathbf{E}_1\widehat{\mathbf{s}}}, \ldots, g_1^{\mathbf{E}_n\widehat{\mathbf{s}}}\right) \in (G_1^{3d})^{n+1}$.
- $\widetilde{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$: Sample $\widetilde{\mathbf{s}} \leftarrow \mathbb{Z}_p^d$ and output $\left(g_1^{\mathbf{F}\widetilde{\mathbf{s}}}, g_1^{\mathbf{F}_1\widetilde{\mathbf{s}}}, \ldots, g_1^{\mathbf{F}_n\widetilde{\mathbf{s}}}\right) \in (G_1^{3d})^{n+1}$.
- $\widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$: Sample $\widehat{\mathbf{r}} \leftarrow \mathbb{Z}_p^d$ and output $g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)\widehat{\mathbf{r}}} \in G_2^{3d}$.
- $\widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$: Sample $\widetilde{\mathbf{r}} \leftarrow \mathbb{Z}_p^d$ and output $g_2^{\pi_{\mathrm{R}}(\mathbf{B}^*)\widetilde{\mathbf{r}}} \in G_2^{3d}$.

4.3 Security Analysis

One can easily check the *projective*, *associative*, *orthogonality*, *non-degeneracy*, $\mathbb{H}$-*subgroup*, and *LS1* properties following [15]. Due to lack of space, we just

give the proof of *left subgroup indistinguishability 2 (LS2)* and sketch the proof of *nested-hiding indistinguishability (NH)*, and leave detailed proofs in the full version of the paper. We emphasize that all three computational properties are *tightly* reduced to the d-Lin assumption.

Left Subgroup Indistinguishability 2. We first rewrite entries involved in the LS2 advantage function $\mathsf{Adv}^{\mathrm{LS2}}_{\mathcal{A}}(k,q,q')$ in terms of $\mathbf{B},\mathbf{B}^*,\mathbf{A}_i,\mathbf{R}$ as follows

$$\mathrm{PP} := \begin{pmatrix} g_1^{\pi_{\mathrm{L}}(\mathbf{B})}, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_1)}, & \cdots, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_n)} \\ g_2^{\mathbf{B}^*\mathbf{R}}, & g_2^{\mathbf{B}^*\mathbf{A}_1^\top\mathbf{R}}, & \cdots, & g_2^{\mathbf{B}^*\mathbf{A}_n^\top\mathbf{R}} \end{pmatrix};$$

$$\widehat{h}_j^* \cdot \widetilde{h}_j^* := g_2^{\mathbf{B}^*\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{r}}_j\\ \widetilde{\mathbf{r}}_j\end{pmatrix}};$$

$$\mathbf{g}'_j \cdot \widehat{\mathbf{g}}'_j := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}'_j\\ \widehat{\mathbf{s}}'_j\\ \mathbf{0}_d\end{pmatrix}}, g_1^{\mathbf{B}\mathbf{A}_1\begin{pmatrix}\mathbf{s}'_j\\ \widehat{\mathbf{s}}'_j\\ \mathbf{0}_d\end{pmatrix}}, \ldots, g_1^{\mathbf{B}\mathbf{A}_n\begin{pmatrix}\mathbf{s}'_j\\ \widehat{\mathbf{s}}'_j\\ \mathbf{0}_d\end{pmatrix}}\right);$$

$$\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \mathbf{0}_d\end{pmatrix}}, g_1^{\mathbf{B}\mathbf{A}_1\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \mathbf{0}_d\end{pmatrix}}, \ldots, g_1^{\mathbf{B}\mathbf{A}_n\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \mathbf{0}_d\end{pmatrix}}\right);$$

$$\mathbf{g}_j \cdot \widetilde{\mathbf{g}}_j := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_j\\ \mathbf{0}_d\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}, g_1^{\mathbf{B}\mathbf{A}_1\begin{pmatrix}\mathbf{s}_j\\ \mathbf{0}_d\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}, \ldots, g_1^{\mathbf{B}\mathbf{A}_n\begin{pmatrix}\mathbf{s}_j\\ \mathbf{0}_d\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}\right);$$

where $\widehat{\mathbf{r}}_j, \widetilde{\mathbf{r}}_j, \mathbf{s}'_j, \widehat{\mathbf{s}}'_j, \mathbf{s}_j, \widehat{\mathbf{s}}_j, \widetilde{\mathbf{s}}_j \leftarrow \mathbb{Z}_p^d$. Then we prove the following lemma.

Lemma 2 ((d,d,q)-LLin $\Rightarrow$ LS2). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{LS2}}_{\mathcal{A}}(k,q,q') \leqslant 2 \cdot \mathsf{Adv}^{(d,d,q)\text{-LLin}}_{\mathcal{B}}(k),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (q+q')d^2 \cdot \mathsf{poly}(k,n)$. *($\mathsf{poly}(k,n)$ is independent of $\mathcal{A}$)*

Overview of Proof. We will prove Lemma 2 in two steps with the help of a transitional distribution $T_{1/2} = \{\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j \cdot \widetilde{\mathbf{g}}_j\}_{j\in[q]}$ where

$$\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j \cdot \widetilde{\mathbf{g}}_j := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}, g_1^{\mathbf{B}\mathbf{A}_1\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}, \ldots, g_1^{\mathbf{B}\mathbf{A}_n\begin{pmatrix}\mathbf{s}_j\\ \widehat{\mathbf{s}}_j\\ \widetilde{\mathbf{s}}_j\end{pmatrix}}\right).$$

In particular, we prove that, given D, distribution T_0 and $T_{1/2}$ are computational indistinguishable under the (d,d,q)-LLin assumption (see Lemma 3), and so do $T_{1/2}$ and T_1 (see Lemma 4). These immediately prove Lemma 2.

Lemma 3 (from T_0 to $T_{1/2}$). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathcal{A}(D,T_0)=1] - \Pr[\mathcal{A}(D,T_{1/2})=1]\right| \leqslant \mathsf{Adv}^{(d,d,q)\text{-LLin}}_{\mathcal{B}}(k),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (q+q')d^2 \cdot \mathsf{poly}(k,n)$. *($\mathsf{poly}(k,n)$ is independent of $\mathcal{A}$)*

Proof Given an instance of (d, d, q)-LLin problem (i.e., set $\ell = d$)

$$\left(g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, \left\{ g_1^{b_{i,j}} \right\}_{i,j\in[d]}, \left\{ g_1^{a_1 s_{1,j}}, \ldots, g_1^{a_d s_{d,j}} \right\}_{j\in[q]}, \left\{ g_1^{b_{i,1}s_{1,j}+\cdots+b_{i,d}s_{d,j}+s_{d+i,j}} \right\}_{i\in[d],j\in[q]} \right)$$

as input where either $s_{d+i,j} = 0$ or $s_{d+i,j} \leftarrow \mathbb{Z}_p^*$, adversary $\mathcal{B}$ works as follows:

Programming $\widehat{\mathbf{s}}_j$ and $\widetilde{\mathbf{s}}_j$ for $j \in [q]$. Adversary $\mathcal{B}$ implicitly sets

$$\widehat{\mathbf{s}}_j := (s_{1,j}, \ldots, s_{d,j})^\top \quad \text{and} \quad \widetilde{\mathbf{s}}_j := (s_{d+1,j}, \ldots, s_{2d,j})^\top.$$

Programming $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{R}$. We define $\mathbf{W}$ as

$$\mathbf{W} := \left(\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & & a_1 & & \\ & & & & \ddots & \\ & & & & & a_d \\ \hline & & & b_{1,1} & \cdots & b_{1,d} \\ & & & \vdots & & \vdots \\ & & & b_{d,1} & \cdots & b_{d,d} \end{array}\right.\left.\begin{array}{c} \\ \\ \\ \\ \\ \\ \hline 1 & & \\ & \ddots & \\ & & 1 \end{array}\right) \in \mathbb{Z}_p^{3d\times 3d}$$

and set $\mathbf{W}^* := (\mathbf{W}^{-1})^\top$. Sample[3] $\bar{\mathbf{B}}, \bar{\mathbf{R}} \leftarrow \mathrm{GL}_{3d}(\mathbb{Z}_p)$ and set $\bar{\mathbf{B}}^* := (\bar{\mathbf{B}}^{-1})^\top$. Also sample $\bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n \leftarrow \mathbb{Z}_p^{3d\times 3d}$, and implicitly set

$$(\mathbf{B}, \mathbf{B}^*) := (\bar{\mathbf{B}}\mathbf{W}, \bar{\mathbf{B}}^*\mathbf{W}^*), \quad \mathbf{R} := \mathbf{W}^\top\bar{\mathbf{R}}, \quad \mathbf{A}_i := \mathbf{W}^{-1}\bar{\mathbf{A}}_i\mathbf{W}, \tag{1}$$

for $i \in [n]$. Observe that $\mathbf{B}, \mathbf{B}^*, \mathbf{R}$ and all $\mathbf{A}_i$ are distributed properly, and

$$\mathbf{B}\mathbf{A}_i = \bar{\mathbf{B}}\bar{\mathbf{A}}_i\mathbf{W}, \quad \mathbf{B}^*\mathbf{R} = \bar{\mathbf{B}}^*\bar{\mathbf{R}}, \quad \mathbf{B}^*\mathbf{A}_i^\top\mathbf{R} = \bar{\mathbf{B}}^*\bar{\mathbf{A}}_i^\top\bar{\mathbf{R}}. \tag{2}$$

Simulating PP. $\mathcal{B}$ can simulate

$$g_1^{\pi_{\mathrm{L}}(\mathbf{B})} = g_1^{\pi_{\mathrm{L}}(\bar{\mathbf{B}}\mathbf{W})} = g_1^{\bar{\mathbf{B}}\pi_{\mathrm{L}}(\mathbf{W})} \quad \text{and} \quad g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_i)} = g_1^{\pi_{\mathrm{L}}(\bar{\mathbf{B}}\bar{\mathbf{A}}_i\mathbf{W})} = g_1^{\bar{\mathbf{B}}\bar{\mathbf{A}}_i\pi_{\mathrm{L}}(\mathbf{W})},$$
$$g_2^{\mathbf{B}^*\mathbf{R}} = g_2^{\bar{\mathbf{B}}^*\bar{\mathbf{R}}} \quad \text{and} \quad g_2^{\mathbf{B}^*\mathbf{A}_i^\top\mathbf{R}} = g_2^{\bar{\mathbf{B}}^*\bar{\mathbf{A}}_i^\top\bar{\mathbf{R}}},$$

for $i \in [n]$ using the knowledge of $\pi_{\mathrm{L}}(\mathbf{W})$ and $\bar{\mathbf{B}}, \bar{\mathbf{B}}^*, \bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n, \bar{\mathbf{R}}$.

[3] In our symbol system, a variable with a bar on the top, say $\bar{\mathbf{B}}$, is sampled by the simulator (i.e., $\mathcal{B}$) and is completely known to it.

Simulating $\widehat{h}_j^* \cdot \widetilde{h}_j^*$ for $j \in [q+q']$. It is not hard to compute $\mathbf{W}^* \in \mathbb{Z}_p^{3d \times 3d}$ as

$$\mathbf{W}^* := \left(\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & & a_1^{-1} & & -a_1^{-1}b_{1,1} \cdots -a_1^{-1}b_{d,1} \\ & & & & \ddots & \vdots \qquad\qquad \vdots \\ & & & & & a_d^{-1} \quad -a_d^{-1}b_{1,d} \cdots -a_d^{-1}b_{d,d} \\ \hline & & & & & 1 \\ & & & & & & \ddots \\ & & & & & & & 1 \end{array}\right).$$

For all $j \in [q+q']$, we sample $\bar{\mathbf{r}}_j \leftarrow \mathbb{Z}_p^{2d}$ and implicitly set

$$\begin{pmatrix} \mathbf{0}_d \\ \widehat{\mathbf{r}}_j \\ \widetilde{\mathbf{r}}_j \end{pmatrix} = (\mathbf{W}^*)^{-1} \begin{pmatrix} \mathbf{0}_d \\ \bar{\mathbf{r}}_j \end{pmatrix} = \mathbf{W}^\top \begin{pmatrix} \mathbf{0}_d \\ \bar{\mathbf{r}}_j \end{pmatrix}.$$

Since the right-bottom $2d \times 2d$ sub-matrix of $\mathbf{W}^*$ is full-rank with overwhelming probability, $\widehat{\mathbf{r}}_j$ and $\widetilde{\mathbf{r}}_j$ are distributed properly and $\mathcal{B}$ can simulate

$$\widehat{h}_j^* \cdot \widetilde{h}_j^* = g_2^{\mathbf{B}^* \begin{pmatrix} \mathbf{0}_d \\ \widehat{\mathbf{r}}_j \\ \widetilde{\mathbf{r}}_j \end{pmatrix}} = g_2^{\bar{\mathbf{B}}^* \mathbf{W}^* \begin{pmatrix} \mathbf{0}_d \\ \widehat{\mathbf{r}}_j \\ \widetilde{\mathbf{r}}_j \end{pmatrix}} = g_2^{\bar{\mathbf{B}}^* \begin{pmatrix} \mathbf{0}_d \\ \bar{\mathbf{r}}_j \end{pmatrix}}$$

using the knowledge of $\bar{\mathbf{B}}^*$ and $\bar{\mathbf{r}}_j$.

Simulating $\mathbf{g}_j' \cdot \widehat{\mathbf{g}}_j'$ for $j \in [q]$. $\mathcal{B}$ can sample $\mathbf{s}_j', \widehat{\mathbf{s}}_j' \leftarrow \mathbb{Z}_p^d$ and simulate

$$g_1^{\mathbf{B} \begin{pmatrix} \mathbf{s}_j' \\ \widehat{\mathbf{s}}_j' \\ \mathbf{0}_d \end{pmatrix}} = g_1^{\bar{\mathbf{B}} \mathbf{W} \begin{pmatrix} \mathbf{s}_j' \\ \widehat{\mathbf{s}}_j' \\ \mathbf{0}_d \end{pmatrix}} \quad \text{and} \quad g_1^{\mathbf{B} \mathbf{A}_i \begin{pmatrix} \mathbf{s}_j' \\ \widehat{\mathbf{s}}_j' \\ \mathbf{0}_d \end{pmatrix}} = g_1^{\bar{\mathbf{B}} \bar{\mathbf{A}}_i \mathbf{W} \begin{pmatrix} \mathbf{s}_j' \\ \widehat{\mathbf{s}}_j' \\ \mathbf{0}_d \end{pmatrix}}$$

for $i \in [n]$ and using the knowledge of $g_1^{\mathbf{W}}$ and $\bar{\mathbf{B}}, \bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n$.

Simulating the challenge. Algorithm $\mathcal{B}$ can sample $\mathbf{s}_j \leftarrow \mathbb{Z}_p^d$ and simulate

$$g_1^{\mathbf{B} \begin{pmatrix} \mathbf{s}_j \\ \widehat{\mathbf{s}}_j \\ \widetilde{\mathbf{s}}_j \end{pmatrix}} = g_1^{\bar{\mathbf{B}} \mathbf{W} \begin{pmatrix} \mathbf{s}_j \\ \widehat{\mathbf{s}}_j \\ \widetilde{\mathbf{s}}_j \end{pmatrix}} \quad \text{and} \quad g_1^{\mathbf{B} \mathbf{A}_i \begin{pmatrix} \mathbf{s}_j \\ \widehat{\mathbf{s}}_j \\ \widetilde{\mathbf{s}}_j \end{pmatrix}} = g_1^{\bar{\mathbf{B}} \bar{\mathbf{A}}_i \mathbf{W} \begin{pmatrix} \mathbf{s}_j \\ \widehat{\mathbf{s}}_j \\ \widetilde{\mathbf{s}}_j \end{pmatrix}}$$

for $i \in [n]$ and $j \in [q]$ using the knowledge of $\bar{\mathbf{B}}, \bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n$ and

$$g_1^{\mathbf{W} \begin{pmatrix} \mathbf{s}_j \\ \widehat{\mathbf{s}}_j \\ \widetilde{\mathbf{s}}_j \end{pmatrix}} = g_1^{\begin{pmatrix} \mathbf{s}_j \\ a_1 s_{1,j} \\ \vdots \\ a_d s_{d,j} \\ b_{1,1}s_{1,j} + \cdots + b_{1,d}s_{d,j} + s_{d+1,j} \\ \vdots \\ b_{d,1}s_{1,j} + \cdots + b_{d,d}s_{d,j} + s_{2d,j} \end{pmatrix}}.$$

Analysis. Observe that if all $s_{d+i,j} = 0$, then all $\widetilde{\mathbf{s}}_j = \mathbf{0}$ and the output challenge is distributed as $\{\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j\}_{j\in[q]}$; otherwise, if all $s_{d+i,j} \leftarrow \mathbb{Z}_p^*$, then all $\widetilde{\mathbf{s}}_j \leftarrow (\mathbb{Z}_p^*)^d$ and the output challenge is distributed as $\{\mathbf{g}_j \cdot \widehat{\mathbf{g}}_j \cdot \widetilde{\mathbf{g}}_j\}_{j\in[q]}$. Therefore we may conclude that $\left|\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_{1/2}) = 1]\right| \leqslant \mathsf{Adv}_{\mathcal{B}}^{(d,d,q)\text{-LLin}}(k)$. □

Lemma 4 (from $T_{1/2}$ to T_1). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathcal{A}(D, T_{1/2}) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]\right| \leqslant \mathsf{Adv}_{\mathcal{B}}^{(d,d,q)\text{-LLin}}(k),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (q+q')d^2 \cdot \mathsf{poly}(k, n)$. *($\mathsf{poly}(k, n)$ is independent of $\mathcal{A}$)*

Proof. Given an instance of (d, d, q)-LLin problem, adversary $\mathcal{B}$ behaves in a similar manner to $\mathcal{B}$ in the proof of Lemma 3 with the differences that:

Programming $\widehat{\mathbf{s}}_j$ and $\widetilde{\mathbf{s}}_j$ for $j \in [q]$. Adversary $\mathcal{B}$ implicitly sets

$$\widehat{\mathbf{s}}_j = (s_{2d,j}, \ldots, s_{d+1,j})^\top \quad \text{and} \quad \widetilde{\mathbf{s}}_j = (s_{d,j}, \ldots, s_{1,j})^\top.$$

Defining $\mathbf{W}$. Adversary $\mathcal{B}$ defines $\mathbf{W}$ as

$$\mathbf{W} := \left(\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & & 1 & & & b_{d,d} & \cdots & b_{d,1} \\ & & & & \ddots & & \vdots & & \vdots \\ & & & & & 1 & b_{1,d} & \cdots & b_{1,1} \\ \hline & & & & & & a_d & & \\ & & & & & & & \ddots & \\ & & & & & & & & a_1 \end{array}\right) \in \mathbb{Z}_p^{3d\times 3d}.$$

In fact, $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_i, \mathbf{R}$ are programmed as Eq. (1). All entries in PP and $\{\mathbf{g}'_j \cdot \widehat{\mathbf{g}}'_j\}$ can be simulated exactly as in the proof of Lemma 3. The strategy for creating $\{\widehat{h}_j^* \cdot \widetilde{h}_j^*\}$ and the challenge there also works well. □

Combining Lemmas 1 and 2, we have Corollary 1 showing that our instantiation satisfies *left subgroup indistinguishability 2* requirement with tight reduction, i.e., with security loss $2d$, to the d-Lin assumption.

Corollary 1 (d-Lin ⇒ LS2). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{LS2}}(k, q, q') \leqslant 2d \cdot \mathsf{Adv}_{\mathcal{B}}^{d\text{-Lin}}(k) + 2/(p-1),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (q+q')d^2 \cdot \mathsf{poly}(k, n)$. *($\mathsf{poly}(k, n)$ is independent of $\mathcal{A}$)*

Nested-Hiding indistinguishability. Since $\widehat{h}_j^{**}$ and $\widetilde{h}_j^{**}$ are respective random vectors in d-dimensional subspace $g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)}$ and $g_2^{\pi_{\mathrm{R}}(\mathbf{B}^*)}$ now, we must "create" more entropy from $\mathbf{h}_j$ than Chen and Wee did in [15]. To do so, we establish a generalized version of many-tuple lemma (see Lemma 5) in [15], which takes the (d, d, d)-LLin assumption as starting point instead of the d-Lin assumption.

Lemma 5 (Generalized Many-Tuple Lemma). *There exists an efficient algorithm that on input $q \in \mathbb{Z}^+$, a finite cyclic group G generated by $g \in G$ and*

$$\left(g, g^{a_1}, \ldots, g^{a_d}, \left\{g^{b_{i,j}}\right\}_{i,j\in[d]}, \left\{g^{a_1 r_{1,j}}, \ldots, g^{a_d r_{d,j}}\right\}_{j\in[d]}, \left\{g^{b_{i,1}r_{1,j}+\cdots+b_{i,d}r_{d,j}+r_{d+i,j}}\right\}_{i,j\in[d]}\right),$$

outputs $\left(g^{\mathbf{VZ}}, g^{\mathbf{Z}}\right)$ for some matrix $\mathbf{V} \in \mathbb{Z}_p^{d\times d}$ along with $\left\{\left(g^{\mathbf{t}_j}, g^{\mathbf{V}\mathbf{t}_j+\boldsymbol{\tau}_j}\right)\right\}_{j\in[q]}$, where $\mathbf{t}_j \leftarrow \mathbb{Z}_p^d$, all $\boldsymbol{\tau}_j$ are either $\mathbf{0}_d$ or uniformly distributed over $\mathbb{Z}_p^d$. And $\mathbf{Z}$ is an invertible diagonal matrix.

Then the proof for the NH property can be obtained by properly embedding matrix $\mathbf{V}$ into $\mathbf{A}_{2\eta-1}$ and $\mathbf{A}_{2\eta}$ and matrix $\mathbf{Z}$ into $\mathbf{R}$, and naturally extending Chen and Wee's simulation strategy [15].

5 Concrete IBE from d-Linear Assumption

This section describes the concrete IBE scheme derived from our prime-order instantiation in Sect. 4 following Hofheinz *et al.*'s framework [21]. Let GrpGen be the bilinear group generator described in Sect. 4.1 and $\pi_{\mathrm{L}}(\cdot)$ be the function mapping from a $3d \times 3d$ matrix to its left-most d columns.

– $\mathsf{Param}(1^k, n)$: Run $(p, G_1, G_2, G_T, e) \leftarrow \mathsf{GrpGen}(1^k)$. Sample $\mathbf{B}, \mathbf{R} \leftarrow \mathrm{GL}_{3d}(\mathbb{Z}_p)$ and $\mathbf{A}_1, \ldots, \mathbf{A}_{2n} \leftarrow \mathbb{Z}_p^{3d\times 3d}$, and set $\mathbf{B}^* := (\mathbf{B}^{-1})^\top$. Output

$$\mathrm{GP} := \begin{pmatrix} g_1^{\pi_{\mathrm{L}}(\mathbf{B})}, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_1)}, & \cdots, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_{2n})} \\ g_2^{\mathbf{B}^*\mathbf{R}}, & g_2^{\mathbf{B}^*\mathbf{A}_1^\top\mathbf{R}}, & \cdots, & g_2^{\mathbf{B}^*\mathbf{A}_{2n}^\top\mathbf{R}} \end{pmatrix}.$$

– $\mathsf{Setup}(\mathrm{GP})$: Sample $\mathbf{k} \leftarrow \mathbb{Z}_p^{3d}$ and output

$$\mathrm{MPK} := \left(g_1^{\pi_{\mathrm{L}}(\mathbf{B})}, g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_1)}, \ldots, g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_{2n})}; e(g_1, g_2)^{\pi_{\mathrm{L}}(\mathbf{B})^\top\mathbf{k}}\right) \in (G_1^{3d\times d})^{2n+1} \times G_T^d;$$
$$\mathrm{MSK} := \left(g_2^{\mathbf{B}^*\mathbf{R}}, g_2^{\mathbf{B}^*\mathbf{A}_1^\top\mathbf{R}}, \ldots, g_2^{\mathbf{B}^*\mathbf{A}_{2n}^\top\mathbf{R}}; g_2^{\mathbf{k}}\right) \in (G_2^{3d\times 3d})^{2n+1} \times G_2^{3d}.$$

– $\mathsf{KeyGen}(\mathrm{MPK}, \mathrm{MSK}, \mathbf{y})$: Let $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$. Sample $\mathbf{r} \leftarrow \mathbb{Z}_p^{3d}$ and output

$$\mathrm{SK}_{\mathbf{y}} := \left(g_2^{\mathbf{B}^*\mathbf{R}\mathbf{r}}, \; g_2^{\mathbf{k}+\mathbf{B}^*(\mathbf{A}_{2-y_1}+\cdots+\mathbf{A}_{2n-y_n})^\top\mathbf{R}\mathbf{r}}\right) \in G_2^{3d} \times G_2^{3d}.$$

– $\mathsf{Enc}(\mathrm{MPK}, \mathbf{x}, \mathrm{M})$: Let $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ and $\mathrm{M} \in \mathbb{G}_T$. Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output

$$\mathrm{CT}_{\mathbf{x}} := \left(g_1^{\pi_{\mathrm{L}}(\mathbf{B})\mathbf{s}},\ g_1^{\pi_{\mathrm{L}}(\mathbf{B}(\mathbf{A}_{2-x_1}+\cdots+\mathbf{A}_{2n-x_n}))\mathbf{s}},\ e(g_1,g_2)^{\mathbf{s}^\top \pi_{\mathrm{L}}(\mathbf{B})^\top \mathbf{k}} \cdot \mathrm{M} \right)$$
$$\in G_1^{3d} \times G_1^{3d} \times G_T.$$

– $\mathsf{Dec}(\mathrm{MPK}, \mathrm{SK}, \mathrm{CT})$: Let $\mathrm{SK} = (K_0, K_1)$ and $\mathrm{CT} = (C_0, C_1, C_2)$. Output

$$\mathrm{M} := C_2 \cdot e(C_1, K_0)/e(C_0, K_1).$$

Note that we only put necessary entries for Enc into MPK, while entries from GP (or PP) for running KeyGen are put into MSK. We describe the following theorem.

Theorem 1. *For any p.p.t. adversary $\mathcal{A}$ making at most q_K key extraction queries and at most q_C challenge queries for pairwise distinct challenge identity against at most λ instances, there exists adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{IBE}}_{\mathcal{A}}(k, \lambda, q_K, q_C, 1) \leqslant d \cdot (5n+1) \cdot \mathsf{Adv}^{d\text{-Lin}}_{\mathcal{B}}(k) + 2^{-\Omega(k)},$$

where $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (\lambda + q_C + q_K) \cdot d^2 \cdot \mathsf{poly}(k, n)$ and $\mathsf{poly}(k, n)$ is independent of $\mathsf{Time}(\mathcal{A})$.

6 Achieving Stronger Security Guarantee

This section will investigate two flavors of stronger adaptive security: *B-weak* and *full* adaptive security (see Sect. 2) by enhancing the *non-degeneracy* property and updating the proof of "ENDSG implies IBE".

6.1 Warmup: Achieving B-weak Adaptive Security

Recall that the original non-degeneracy property said that:

(Non-degeneracy (Recalled).) Over the probability space defined by $\widehat{g}_0 \leftarrow \widehat{\mathsf{SampG}}_0(\mathrm{PP}, \mathrm{SP})$, with overwhelming probability $1 - 2^{-\Omega(k)}$, $e(\widehat{g}_0, \widehat{h}^*)$ is distributed uniformly over $\mathbb{G}_T$ when sampling $\widehat{h}^* \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$.

We observe that $\widehat{h}^*$ in our prime-order instantiation (see Sect. 4) actually contains higher entropy than those in Hofheinz *et al.*'s composite-order instantiation [21]. In particular, $\widehat{h}^*$ is uniformly distributed over a d-dimension subspace of G_2^{3d} containing p^d elements (vectors), while $e(\widehat{g}_0, \widehat{h}^*)$ is an element in G_T containing just p elements. This suggests that, given $e(\widehat{g}_0, \widehat{h}^*)$, there may be leftover entropy in $\widehat{h}^*$, and our prime-order instantiation may achieve stronger non-degeneracy even relying on no computational assumption.

To formally investigate the above idea, we describe the notion of *B-bounded non-degeneracy* which roughly ensures the non-degeneracy when a single $\widehat{h}^*$ is paired with at most B $\widehat{g}_0$'s.

(B-bounded non-degeneracy.) Over the probability space defined by sampling $(\widehat{g}_{0,1}, \ldots, \widehat{g}_{0,B}) \leftarrow \widehat{\mathsf{SampG}_0}^{B}(\mathrm{PP}, \mathrm{SP})$, with overwhelming probability $1 - 2^{-\Omega(k)}$, $(e(\widehat{g}_{0,1}, \widehat{h}^*), \ldots, e(\widehat{g}_{0,B}, \widehat{h}^*))$ is distributed uniformly over $\mathbb{G}_T^B$ when sampling $\widehat{h}^* \leftarrow \widehat{\mathsf{SampH}}^{*}(\mathrm{PP}, \mathrm{SP})$.

It is obvious that the ENDSG with B-bounded non-degeneracy almost-tightly implies a B-weak adaptively secure IBE in the MIMC setting. We now prove that our prime-order instantiation in Sect. 4 indeed reaches this stronger version of non-degeneracy.

Lemma 6. *Our prime-order instantiation of ENDSG in Sect. 4 based on the d-Lin assumption is d-bounded non-degenerated.*

Proof. The proof is just a simple statistical argument extended from the proof for the original non-degeneracy. For $\widehat{\mathbf{s}}_1, \ldots, \widehat{\mathbf{s}}_d \leftarrow \mathbb{Z}_p^d$ and $\widehat{\mathbf{r}} \leftarrow \mathbb{Z}_p^d$, we have that

$$\begin{pmatrix} e(g_1^{\mathbf{E}\widehat{\mathbf{s}}_1}, g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)\widehat{\mathbf{r}}}) \\ \vdots \\ e(g_1^{\mathbf{E}\widehat{\mathbf{s}}_d}, g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)\widehat{\mathbf{r}}}) \end{pmatrix} = \begin{pmatrix} e(g_1, g_2)^{\widehat{\mathbf{s}}_1^\top \widehat{\mathbf{r}}} \\ \vdots \\ e(g_1, g_2)^{\widehat{\mathbf{s}}_d^\top \widehat{\mathbf{r}}} \end{pmatrix} = e(g_1, g_2)^{\begin{pmatrix} \widehat{\mathbf{s}}_1^\top \\ \vdots \\ \widehat{\mathbf{s}}_d^\top \end{pmatrix} \widehat{\mathbf{r}}}.$$

With probability at least $1 - \frac{1}{p-1}$, the matrix $(\widehat{\mathbf{s}}_1, \ldots, \widehat{\mathbf{s}}_d)^\top$ is full-rank, in which case $(\widehat{\mathbf{s}}_1, \ldots, \widehat{\mathbf{s}}_d)^\top \widehat{\mathbf{r}}$ is distributed uniformly over $\mathbb{Z}_p^d$ when picking $\widehat{\mathbf{r}} \leftarrow \mathbb{Z}_p^d$. □

Therefore, when we build our instantiation with parameter $d > 1$, we actually obtain an IBE with strictly stronger security guarantee which ensures the confidentiality of at most d ciphertexts for each identity. As a special case, if we set $d = 1$ (i.e., the SXDH assumption), the resulting IBE is still weak secure.

6.2 Computational Non-degeneracy and Full Adaptive Security

The attempt in the previous subsection more or less suggests that it is probably inevitable to introduce additional computational arguments in order to achieve fully adaptive security where a single $\widehat{h}^*$ can be paired with polynomially many $\widehat{g}_0$'s without violating the non-degeneracy property.

As a first step, we describe a computational version of non-degeneracy which is essentially similar to the s-BDDH assumption [21]. Our presentation follows the style of our revised ENDSG (in Sect. 3) in order to keep generality.

(Computational non-degeneracy (ND).) For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ND}}(k, q, q', q'') := |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|,$$

where

$$D := \left(\mathrm{PP}, \left\{\widehat{h}_j^* \cdot \widetilde{h}_j^*\right\}_{j \in [q']}, \{\widehat{\mathbf{g}}_{j,j'}\}_{j \in [q], j' \in [q'']}\right),$$

$$T_0 := \left\{ e(\widehat{g}_{0,j,j'}, \widehat{h}_j^{**}) \right\}_{j\in[q],j'\in[q'']},\ T_1 := \{R_{j,j'}\}_{j\in[q],j'\in[q'']}$$

and $\widehat{h}_j^* \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$, $\widetilde{h}_j^* \leftarrow \widetilde{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$, $\widehat{h}_j^{**} \leftarrow \widehat{\mathsf{SampH}}^*(\mathrm{PP}, \mathrm{SP})$, $\widehat{\mathbf{g}}_{j,j'} = (\widehat{g}_{0,j,j'}, \widehat{g}_{1,j,j'}, \ldots, \widehat{g}_{n,j,j'}) \leftarrow \widehat{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$ and $R_{j,j'} \leftarrow \mathbb{G}_T$.

It is not hard to see that an ENDSG with computational non-degeneracy property almost-tightly implies a fully adaptively secure IBE in MIMC setting, where we ensure the confidentiality of polynomial-many ciphertexts for each identity. The detailed proof can be found in the full version of the paper.

6.3 Computational Non-degeneracy from *d*-Linear Assumption

We now prove that the prime-order instantiation proposed in Sect. 4 has realized the computational non-degeneracy. And this immediately implies that the concrete IBE scheme shown in Sect. 5 is fully adaptively secure in MIMC setting with almost-tight reduction.

As before, we first rewrite all entries involved in the ND advantage function $\mathsf{Adv}^{\mathrm{ND}}_{\mathcal{A}}(k, q, q', q'')$ in terms of $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_i, \mathbf{R}$ as follows

$$\begin{aligned}
\mathrm{PP} &:= \begin{pmatrix} g_1^{\pi_{\mathrm{L}}(\mathbf{B})}, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_1)}, & \ldots, & g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_n)} \\ g_2^{\mathbf{B}^*\mathbf{R}}, & g_2^{\mathbf{B}^*\mathbf{A}_1^\top\mathbf{R}}, & \ldots, & g_2^{\mathbf{B}^*\mathbf{A}_n^\top\mathbf{R}} \end{pmatrix}; \\
\widehat{h}_j^* \cdot \widetilde{h}_j^* &:= g_2^{\mathbf{B}^* \begin{pmatrix} \mathbf{0}_d \\ \widehat{\mathbf{r}}'_j \\ \widetilde{\mathbf{r}}'_j \end{pmatrix}}; \\
\widehat{\mathbf{g}}_{j,j'} &:= \left(g_1^{\pi_{\mathrm{M}}(\mathbf{B})\widehat{\mathbf{s}}_{j,j'}}, g_1^{\pi_{\mathrm{M}}(\mathbf{B}\mathbf{A}_1)\widehat{\mathbf{s}}_{j,j'}}, \ldots, g_1^{\pi_{\mathrm{M}}(\mathbf{B}\mathbf{A}_n)\widehat{\mathbf{s}}_{j,j'}} \right); \\
e(\widehat{g}_{0,j,j'}, \widehat{h}_j^{**}) &:= e(g_1^{\pi_{\mathrm{M}}(\mathbf{B})\widehat{\mathbf{s}}_{j,j'}}, g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)\widehat{\mathbf{r}}_j}) = e(g_1, g_2)^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j}; \\
R_{j,j'} &:= e(\widehat{g}_{0,j,j'}, \widehat{h}_j^{**}) \cdot e(g_1, g_2)^{\widehat{\gamma}_{j,j'}} = e(g_1, g_2)^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j} \cdot e(g_1, g_2)^{\widehat{\gamma}_{j,j'}};
\end{aligned}$$

where $\widehat{\mathbf{r}}'_j, \widetilde{\mathbf{r}}'_j, \widehat{\mathbf{r}}_j, \widehat{\mathbf{s}}_{j,j'} \leftarrow \mathbb{Z}_p^d$ and $\widehat{\gamma}_{j,j'} \leftarrow \mathbb{Z}_p$. Then we prove the following lemma.

Lemma 7. **($(d, 1, qq'')$-LLin $\Rightarrow$ ND).** *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathrm{ND}}_{\mathcal{A}}(k, q, q', q'') \leqslant \mathsf{Adv}^{(d,1,qq'')\text{-LLin}}_{\mathcal{B}}(k),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (qq'' + q')d^2 \cdot \mathsf{poly}(k, n)$. *($\mathsf{poly}(k, n)$ is independent of $\mathcal{A}$)*

Overview of Proof. From the observation that all $\widehat{h}_j^{**} = g_2^{\pi_{\mathrm{M}}(\mathbf{B}^*)\widehat{\mathbf{r}}_j}$ are independently distributed and will never be given to $\mathcal{A}$ individually, we essentially prove a stronger result:

"Given D, $g_1^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j}$ are computationally indistinguishable from $g_1^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j + \widehat{\gamma}_{j,j'}}$."

It is direct to based the pseudo-randomness of the challenge terms on the (d, q, q'')-LLin assumption. However the assumption is reduced to d-Lin assumption with reduction loss $\mathcal{O}(q)$. In order to obtain a tight reduction, we further rewrite the challenge term as

$$g_1^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j} = g_1^{\widehat{\mathbf{s}}_{j,j'}^\top \mathbf{V}^\top \bar{\mathbf{r}}_j} = g_1^{\bar{\mathbf{r}}_j^\top \mathbf{V} \widehat{\mathbf{s}}_{j,j'}}$$

where $\mathbf{V}$ is a $(d+1) \times d$ matrix over $\mathbb{Z}_p$ of rank d and $\bar{\mathbf{r}}_j \leftarrow \mathbb{Z}_p^{d+1}$. Clearly, we implicitly define $\widehat{\mathbf{r}}_j := \mathbf{V}^\top \bar{\mathbf{r}}_j$. Since the matrix $\mathbf{V}$ is shared by all $\widehat{\mathbf{r}}_j$'s in challenge terms, we could now deal with polynomially many distinct $\widehat{\mathbf{r}}_j$'s *uniformly* which results in a proof with *constant* security loss.

Proof. Given an instance of $(d, 1, qq'')$-LLin problem (i.e., set $\ell = 1$ and $q = qq''$)

$$\left(g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, \left\{ g_1^{b_i} \right\}_{i \in [d]}, \left\{ g_1^{a_1 s_{1,j,j'}}, \ldots, g_1^{a_d s_{d,j,j'}} \right\}_{j \in [q], j' \in [q'']}, \left\{ g_1^{b_1 s_{1,j,j'} + \cdots + b_d s_{d,j,j'} + s_{d+1,j,j'}} \right\}_{j \in [q], j' \in [q'']} \right)$$

as input where either $s_{d+1,j,j'} = 0$ or $s_{d+1,j,j'} \leftarrow \mathbb{Z}_p^*$, $\mathcal{B}$ works as follows:

Programming $\widehat{\mathbf{s}}_{j,j'}$ for $j \in [q], j' \in [q']$. Adversary $\mathcal{B}$ implicitly sets

$$\widehat{\mathbf{s}}_{j,j'} := (s_{1,j,j'}, \ldots, s_{d,j,j'})^\top.$$

Programming $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{R}$. Define $\mathbf{W}$ as

$$\mathbf{W} := \left(\begin{array}{ccc|ccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ \hline & & & a_1 & & \\ & & & & \ddots & \\ & & & & & a_d \\ \hline & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{array} \right) \in \mathbb{Z}_p^{3d \times 3d}$$

and set $\mathbf{W}^* := (\mathbf{W}^{-1})^\top$. Sample $\bar{\mathbf{B}}, \bar{\mathbf{R}} \leftarrow \mathrm{GL}_{3d}(\mathbb{Z}_p)$ and set $\bar{\mathbf{B}}^* := (\bar{\mathbf{B}}^{-1})^\top$. Sample $\bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n \leftarrow \mathbb{Z}_p^{3d \times 3d}$, and implicitly set $\mathbf{B}$, $\mathbf{B}^*$, $\mathbf{R}$, and all $\mathbf{A}_i$ as Eq. (1). Of course, we also have the same relation as Eq. (2).

Simulating PP. Algorithm $\mathcal{B}$ can simulate

$$g_1^{\pi_{\mathrm{L}}(\mathbf{B})} = g_1^{\pi_{\mathrm{L}}(\bar{\mathbf{B}}\mathbf{W})} = g_1^{\bar{\mathbf{B}}\pi_{\mathrm{L}}(\mathbf{W})} \quad \text{and} \quad g_1^{\pi_{\mathrm{L}}(\mathbf{B}\mathbf{A}_i)} = g_1^{\pi_{\mathrm{L}}(\bar{\mathbf{B}}\bar{\mathbf{A}}_i\mathbf{W})} = g_1^{\bar{\mathbf{B}}\bar{\mathbf{A}}_i\pi_{\mathrm{L}}(\mathbf{W})},$$
$$g_2^{\mathbf{B}^*\mathbf{R}} = g_2^{\bar{\mathbf{B}}^*\bar{\mathbf{R}}} \quad \text{and} \quad g_2^{\mathbf{B}^*\mathbf{A}_1^\top\mathbf{R}} = g_2^{\bar{\mathbf{B}}^*\bar{\mathbf{A}}_i^\top\bar{\mathbf{R}}},$$

for $i \in [n]$ using the knowledge of $\pi_{\mathrm{L}}(\mathbf{W})$ and $\bar{\mathbf{B}}, \bar{\mathbf{B}}^*, \bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n, \bar{\mathbf{R}}$.

Simulating $\widehat{h}_j^* \cdot \widetilde{h}_j^*$ for $j \in [q']$. It is not hard to compute $\mathbf{W}^* \in \mathbb{Z}_p^{3d \times 3d}$ as

$$\mathbf{W}^* := \left(\begin{array}{ccc|ccc|ccc} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ \hline & & & a_1^{-1} & & & & & \\ & & & & \ddots & & & & \\ & & & & & a_d^{-1} & & & \\ \hline & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{array}\right).$$

Observe that the right-bottom $2d \times 2d$ sub-matrix of $\mathbf{W}^*$ is full-rank with overwhelming probability, adversary $\mathcal{B}$ can simulate all $\widehat{h}_j^* \cdot \widetilde{h}_j^*$ as in the proof of Lemma 3 for the same reason.

Simulating $\widehat{\mathbf{g}}_{j,j'}$ for $j \in [q], j' \in [q']$. Algorithm $\mathcal{B}$ can simulate

$$g_1^{\mathbf{B}\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{s}}_{j,j'}\\ \mathbf{0}_d\end{pmatrix}} = g_1^{\bar{\mathbf{B}}\mathbf{W}\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{s}}_{j,j'}\\ \mathbf{0}_d\end{pmatrix}} \quad \text{and} \quad g_1^{\mathbf{B}\mathbf{A}_i\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{s}}_{j,j'}\\ \mathbf{0}_d\end{pmatrix}} = g_1^{\bar{\mathbf{B}}\bar{\mathbf{A}}_i\mathbf{W}\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{s}}_{j,j'}\\ \mathbf{0}_d\end{pmatrix}}$$

for $i \in [n]$ using the knowledge of $\bar{\mathbf{B}}, \bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_n$ and

$$g_1^{\mathbf{W}\begin{pmatrix}\mathbf{0}_d\\ \widehat{\mathbf{s}}_{j,j'}\\ \mathbf{0}_d\end{pmatrix}} = g_1^{\begin{pmatrix}\mathbf{0}_d\\ a_1 s_{1,j,j'}\\ \vdots\\ a_d s_{d,j,j'}\\ \mathbf{0}_d\end{pmatrix}}.$$

Simulating the challenge. Define matrix $\mathbf{V} \in \mathbb{Z}_p^{(d+1)\times d}$ of rank d as

$$\mathbf{V} := \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_d \\ b_1 & \cdots & b_d \end{pmatrix}.$$

For all $j \in [q]$, algorithm $\mathcal{B}$ samples $\bar{\mathbf{r}}_j \leftarrow \mathbb{Z}_p^{d+1}$ and implicitly set $\widehat{\mathbf{r}}_j^\top := \bar{\mathbf{r}}_j^\top \mathbf{V}$. Algorithm $\mathcal{B}$ computes

$$g_1^{\widehat{\mathbf{r}}_j^\top \widehat{\mathbf{s}}_{j,j'} + \widehat{\gamma}_{j,j'}} = g_1^{\bar{\mathbf{r}}_j^\top \begin{pmatrix} a_1 s_{1,j,j'} \\ \vdots \\ a_d s_{d,j,j'} \\ b_1 s_{1,j,j'} + \cdots + b_d s_{d,j,j'} + s_{d+1,j,j'} \end{pmatrix}}$$

and outputs $e(g_1^{\widehat{\mathbf{r}}_j^\top \widehat{\mathbf{s}}_{j,j'} + \widehat{\gamma}_{j,j'}}, g_2)$ as challenges.

Analysis. Observe that, if $s_{d+1,j,j'} = 0$, the output challenge is distributed as

$$e(g_1^{\bar{\mathbf{r}}_j^\top(\mathbf{V}\widehat{\mathbf{s}}_{j,j'})}, g_2) = e(g_1, g_2)^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j}$$

which is identical to T_0 where $\widehat{\gamma}_{j,j'} = 0$; if $s_{d+1,j,j'} \leftarrow \mathbb{Z}_p^*$, the output challenge is distributed as

$$e(g_1^{\bar{\mathbf{r}}_j^\top(\mathbf{V}\widehat{\mathbf{s}}_{j,j'} + \mathbf{e}_{d+1} s_{d+1,j,j'})}, g_2) = e(g_1, g_2)^{\widehat{\mathbf{s}}_{j,j'}^\top \widehat{\mathbf{r}}_j} \cdot \boxed{e(g_1, g_2)^{s_{d+1,j,j'} \mathbf{e}_{d+1}^\top \bar{\mathbf{r}}_j}}$$

which is identical to T_1 where $\widehat{\gamma}_{j,j'} := s_{d+1,j,j'} \mathbf{e}_{d+1}^\top \bar{\mathbf{r}}_j$ (in the box) is uniformly distributed over $\mathbb{Z}_p$. Therefore we may conclude that $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ND}}(k, q, q', q'') \leqslant \mathsf{Adv}_{\mathcal{B}}^{(d,1,qq'')\text{-LLin}}(k)$. □

Applying Lemma 1, we obtain the following corollary.

Corollary 2. (d-Lin ⇒ ND). *For any p.p.t. adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ND}}(k, q, q', q'') \leqslant \mathsf{Adv}_{\mathcal{B}}^{d\text{-Lin}}(k) + 1/(p-1),$$

and $\mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{A}) + (qq'' + q')d^2 \cdot \mathsf{poly}(k, n)$. *($\mathsf{poly}(k, n)$ is independent of $\mathcal{A}$)*

7 Towards More Efficient Solution: An Overview

7.1 Motivation and Technique

To obtain more efficient solutions, a promising idea is to reduce the dimension of two semi-functional spaces. Because we hope to continue to base our construction on the standard d-Lin assumption, we found the attempt gives rise to two technical problems due to the lack of dimensions.

- We can not prove *Left Subgroup Indistinguishability 2* (LS2) property using the technique provided by Chen and Wee in [16]. In particular, the simulator will need some elements in another source group (i.e., G_2) to simulate $\widehat{h}^* \cdot \widetilde{h}^*$ which is not given in the standard d-Lin assumption.
- We can not prove *Computational Non-degeneracy* (ND) property as before since neither $\widehat{g}_0$ nor $\widehat{h}^*$ has enough dimensions to program the d-Lin problem during the simulation.

The second issue is easy to solve by the observation that there are two semi-functional spaces and we only use one of them so far. We first define a variant of computational non-degeneracy property taking the $\sim$-semi-functional space into account. As long as two semi-functional spaces together has at least d dimensions, this computational non-degeneracy property should be proved as before. On the other hand, from the view of IBE, we could use the pseudo-randomness of $e(\widehat{g}_0 \cdot \widetilde{g}_0, \widehat{h}^* \cdot \widetilde{h}^*)$ to prove the security (decoupling challenge messages and

ciphertexts) instead of just $e(\widehat{g}_0, \widehat{h}^*)$. To make the intuition explicit and general, we define three Left-subgroup indistinguishability (LS) requirements as: (1) LS1: $\mathbf{g} \approx \mathbf{g} \cdot \widehat{\mathbf{g}} \cdot \widetilde{\mathbf{g}}$; (2) LS2: $\mathbf{g} \cdot \widehat{\mathbf{g}} \cdot \widetilde{\mathbf{g}} \approx \mathbf{g} \cdot \widetilde{\mathbf{g}}$; (3) LS3: $\mathbf{g} \cdot \widehat{\mathbf{g}} \cdot \widetilde{\mathbf{g}} \approx \mathbf{g} \cdot \widehat{\mathbf{g}}$, where $\approx$ stands for "computationally indistinguishable".

In contrast, the first issue is seemingly hard to circumvent. Therefore, we decide to prove the LS2 property under an enhanced d-Lin assumption where we give adversary more elements on another source group G_2 for simulating $\widehat{h}^* \cdot \widetilde{h}^*$, which is called d-linear assumption with auxiliary input (d-LinAI) for *an even positive integer* d. Even though this assumption is non-standard in general, we point out that the concrete assumption with $d = 2$ is implied by the external decision linear assumption (XDLIN) [1] (see below), which has been formally introduced and used to build other cryptographic primitives.

We further fine-tune the ENDSG by hiding public parameters for SampH from the adversary when defining computational requirements, including LS1, LS2, LS3, NH, and ND. We argue that the absence of this part of public parameters will not arise difficulty in building IBE since they always correspond to the master secret key which is not necessary to be public according to the security model. Instead, we give the adversary enough samples from $\mathbb{H}^{n+1}$ which is sufficient for answering key extraction queries in the proof of "ENDSG implies IBE". We hope it will bring us a simple, clean and efficient solution.

In summary, we have fine-tuned the ENDSG in three aspects: (1) update non-degeneracy requirement; (2) re-define LS requirements; (3) hide parameters for SampH. Due to the lack of space, the fine-tuned ENDSG is given in the full version of the paper and we also verify there that these modifications won't prevent ENDSG from almost-tightly deriving a fully secure IBE in MIMC setting.

The starting point of instantiating the fine-tuned ENDSG is the prime-order instantiation of dual system groups recently proposed by Chen *et al.* [13], which is quite simple due to a new basis randomizing technique. We technically work with $2d \times 2d$ matrix (for even positive integer d) and generate the basis using the dual pairing vector space method [26,29,30]. The first d-dimension subspace is normal space, the remaining two $d/2$-dimension subspaces act as $\wedge$-semi-functional subspace and $\sim$-semi-functional subspace, respectively. Note that the latter two are now smaller but enough for our proof (the entire semi-functional space has d dimensions). Finally, the basis is then randomized following [13]. Its security is tightly based on the d-LinAI assumption, which leads to an almost-tightly secure IBE in the MIMC setting with full security and higher efficiency. We describe, in the next subsection, the d-LinAI assumption and the resulting IBE scheme. More details could be found in the full version of the paper.

7.2 Concrete IBE from d-Linear Assumption with Auxiliary Input

Assume a prime-order bilinear group generator $\mathsf{GrpGen}(1^k)$ as defined in Sect. 4. The *d-linear assumption in G_1 with auxiliary input in G_2* (d-LinAI) is defined as follows, the analogous assumption in G_2 can be defined by exchanging the role of G_1 and G_2. We prove that the assumption holds in the generic model [34]

in the full version of the paper. *Note that we always let d be an even positive integer.*

Assumption 3 (d-Linear Assumption in G_1 with Auxiliary Input). For any p.p.t. adversary $\mathcal{A}$, the following advantage function is negligible in k,

$$\mathsf{Adv}_{\mathcal{A}}^{d\text{-LinAI}}(k) := \left|\Pr[\mathcal{A}(D, \textsc{Aux}, T_0) = 1] - \Pr[\mathcal{A}(D, \textsc{Aux}, T_1) = 1]\right|,$$

where

$$\begin{aligned} D &:= \left(\mathcal{G}, g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 s_1}, \ldots, g_1^{a_d s_d}\right) \\ \textsc{Aux} &:= \left(g_2^{a a_1^{-1} a_{d+1}}, \ldots, g_2^{a a_{d/2}^{-1} a_{d+1}}, g_2^{a}\right) \\ T_0 &:= g_1^{a_{d+1}(s_1+\cdots+s_d)},\ T_1 := g_1^{a_{d+1}(s_1+\cdots+s_d)+\boxed{s_{d+1}}} \end{aligned}$$

and $\mathcal{G} \leftarrow \mathsf{GrpGen}(1^k)$, $a_1, \ldots, a_{d+1}, s_{d+1} \leftarrow \mathbb{Z}_p^*$, $a := a_1 \cdots a_{d/2}$, $s_1, \ldots, s_d \leftarrow \mathbb{Z}_p$.

Let $\pi_{\mathrm{L}}(\cdot)$ be the function mapping from a $2d \times 2d$ matrix to its left-most d columns. Given an bilinear group generator GrpGen such that d-LinAI assumption holds, the resulting IBE scheme built according to the main idea shown in the previous subsection is defined as follows.

- $\mathsf{Param}(1^k, n)$: Run $(p, G_1, G_2, G_T, e) \leftarrow \mathsf{GrpGen}(1^k)$. Sample $\mathbf{D} \leftarrow \mathrm{GL}_{2d}(\mathbb{Z}_p)$ and $\mathbf{W}_1, \ldots, \mathbf{W}_{2n} \leftarrow \mathbb{Z}_p^{2d \times 2d}$, and set $\mathbf{D}^* := (\mathbf{D}^{-1})^\top$. Output

$$\textsc{gp} := \begin{pmatrix} g_1^{\pi_{\mathrm{L}}(\mathbf{D})}, & g_1^{\mathbf{W}_1^\top \pi_{\mathrm{L}}(\mathbf{D})}, & \cdots, & g_1^{\mathbf{W}_{2n}^\top \pi_{\mathrm{L}}(\mathbf{D})} \\ g_2^{\pi_{\mathrm{L}}(\mathbf{D}^*)}, & g_2^{\mathbf{W}_1 \pi_{\mathrm{L}}(\mathbf{D}^*)}, & \cdots, & g_2^{\mathbf{W}_{2n} \pi_{\mathrm{L}}(\mathbf{D}^*)} \end{pmatrix}.$$

- $\mathsf{Setup}(\textsc{gp})$: Sample $\mathbf{k} \leftarrow \mathbb{Z}_p^{2d}$ and output

$$\begin{aligned} \textsc{mpk} &:= \left(g_1^{\pi_{\mathrm{L}}(\mathbf{D})}, g_1^{\mathbf{W}_1^\top \pi_{\mathrm{L}}(\mathbf{D})}, \ldots, g_1^{\mathbf{W}_{2n}^\top \pi_{\mathrm{L}}(\mathbf{D})}; e(g_1, g_2)^{\pi_{\mathrm{L}}(\mathbf{D})^\top \mathbf{k}}\right) \in (G_1^{2d \times d})^{2n+1} \times G_T^d; \\ \textsc{msk} &:= \left(g_2^{\pi_{\mathrm{L}}(\mathbf{D}^*)}, g_2^{\mathbf{W}_1 \pi_{\mathrm{L}}(\mathbf{D}^*)}, \ldots, g_2^{\mathbf{W}_{2n} \pi_{\mathrm{L}}(\mathbf{D}^*)}; g_2^{\mathbf{k}}\right) \in (G_2^{2d \times d})^{2n+1} \times G_2^{2d}. \end{aligned}$$

- $\mathsf{KeyGen}(\textsc{mpk}, \textsc{msk}, \mathbf{y})$: Let $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$. Sample $\mathbf{r} \leftarrow \mathbb{Z}_p^d$ and output

$$\textsc{sk}_{\mathbf{y}} := \left(g_2^{\pi_{\mathrm{L}}(\mathbf{D}^*)\mathbf{r}},\ g_2^{\mathbf{k} + (\mathbf{W}_{2-y_1} + \cdots + \mathbf{W}_{2n-y_n}) \pi_{\mathrm{L}}(\mathbf{D}^*)\mathbf{r}}\right) \in G_2^{2d} \times G_2^{2d}.$$

- $\mathsf{Enc}(\textsc{mpk}, \mathbf{x}, \textsc{m})$: Let $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ and $\textsc{m} \in \mathbb{G}_T$. Sample $\mathbf{s} \leftarrow \mathbb{Z}_p^d$ and output

$$\begin{aligned} \textsc{ct}_{\mathbf{x}} := \Big(&g_1^{\pi_{\mathrm{L}}(\mathbf{D})\mathbf{s}},\ g_1^{(\mathbf{W}_{2-x_1} + \cdots + \mathbf{W}_{2n-x_n})^\top \pi_{\mathrm{L}}(\mathbf{D})\mathbf{s}},\ e(g_1, g_2)^{\mathbf{s}^\top \pi_{\mathrm{L}}(\mathbf{D})^\top \mathbf{k}} \cdot \textsc{m}\Big) \\ &\in G_1^{2d} \times G_1^{2d} \times G_T. \end{aligned}$$

– $\mathsf{Dec}(\mathrm{MPK}, \mathrm{SK}, \mathrm{CT})$. Let $\mathrm{SK} = (K_0, K_1)$ and $\mathrm{CT} = (C_0, C_1, C_2)$. Output

$$\mathrm{M} := C_2 \cdot e(C_1, K_0)/e(C_0, K_1).$$

One may argue that the d-LinAI assumption is not standard and complex. We show that, by setting $d = 2$, we derive the DLIN assumption with auxiliary input $\mathrm{AUX} := (g_2^{a_3}, g_2^{a_1})$. It is easy to verify that this special instantiation is implied by the *External Decision Linear Assumption* [1]. Motivated by this observation, we remark that we may build the above IBE system using *symmetric* bilinear pairings and base the security on the well-known and *standard* Decisional Linear Assumption, where $G_1 = G_2$ and AUX in G_2 is automatically revealed.

Acknowledgements. We want to thank Hoeteck Wee for helpful discussions and all the anonymous reviewers for their helpful comments on earlier drafts of this paper. This work is supported by the National Natural Science Foundation of China (Grant Nos. 61472142, 61411146001, 61321064, 61371083, 61373154, 61172085, 61170080, U1135004), 973 Program (No. 2014CB360501), Science and Technology Commission of Shanghai Municipality (Grant Nos. 14YF1404200, 13JC1403500), the Specialized Research Fund for the Doctoral Program of Higher Education of China through the Prioritized Development Projects under Grant 20130073130004.

References

1. Abe, M., Chase, M., David, B., Kohlweiss, M., Nishimaki, R., Ohkubo, M.: Constant-size structure-preserving signatures: generic constructions and simple assumptions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 4–24. Springer, Heidelberg (2012)
2. Agrawal, S., Chase, M.: A study of pair encodings: predicate encryption in prime order groups. In: Kushilevitz, E., et al. (eds.) TCC 2016-A. LNCS, vol. 9563, pp. 259–288. Springer, Heidelberg (2016)
3. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
4. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)
5. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014)
6. Attrapadung, N., Hanaoka, G., Yamada, S.: A framework for identity-based encryption with almost tight security. In: Iwata, T., et al. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 521–548. Springer, Heidelberg (2015)
7. Attrapadung, N., Yamada, S.: Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 87–105. Springer, Heidelberg (2015)
8. 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)

9. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) Identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (2014)
10. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
11. 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)
12. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 213. Springer, Heidelberg (2001)
13. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015)
14. Chen, J., Lim, H.W., Ling, S., Wang, H., Wee, H.: Shorter IBE and signatures via asymmetric pairings. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 122–140. Springer, Heidelberg (2013)
15. Chen, J., Wee, H.: Fully, (Almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (2013)
16. Chen, J., Wee, H.: Dual system groups and its applications - compact HIBE and more. IACR Crypt. ePrint Arch. **2014**, 265 (2014)
17. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, p. 360. Springer, Heidelberg (2001)
18. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 197–206 (2008)
20. Gong, J., Cao, Z., Tang, S., Chen, J.: Extended dual system group and shorter unbounded hierarchical identity based encryption. Des. Codes Crypt. (2015). doi:10.1007/s10623-015-0117-z
21. Hofheinz, D., Koch, J., Striecks, C.: Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 799–822. Springer, Heidelberg (2015)
22. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013)
23. Jutla, C.S., Roy, A.: Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 295–312. Springer, Heidelberg (2014)
24. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
25. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)

26. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
27. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
28. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM **51**(2), 231–262 (2004)
29. Okamoto, T., Takashima, K.: Homomorphic encryption and signatures from vector decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
30. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
31. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
32. Ramanna, S.C., Chatterjee, S., Sarkar, P.: Variants of waters' dual system primitives using asymmetric pairings - (extended abstract). Pub. Key Crypt. - PKC **2012**, 298–315 (2012)
33. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
34. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
35. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
36. 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)
37. Wee, H.: Dual system encryption via predicate encodings. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 616–637. Springer, Heidelberg (2014)

Functional Encryption for Inner Product with Full Function Privacy

Pratish Datta(✉), Ratna Dutta, and Sourav Mukhopadhyay

Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur 721302, India
{pratishdatta,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. *Functional encryption* (FE) supports constrained decryption keys that allow decrypters to learn specific functions of encrypted messages. In numerous practical applications of FE, confidentiality must be assured not only for the encrypted data but also for the functions for which functional keys are provided. This paper presents a *non-generic simple* private key FE scheme for the *inner product* functionality, also known as *inner product encryption* (IPE). In contrast to the existing similar schemes, our construction achieves the *strongest* indistinguishability-based notion of *function privacy* in the *private key* setting without employing any computationally expensive cryptographic tool or non-standard complexity assumption. Our construction is built in the *asymmetric bilinear pairing group* setting of *prime* order. The security of our scheme is based on the well-studied *Symmetric External Diffie-Hellman* (SXDH) assumption.

Keywords: Functional encryption · Inner product · Function privacy · Asymmetric bilinear group

1 Introduction

The recent advancement in cloud technology has triggered an emerging trend among individuals and organizations to outsource potentially sensitive private informations to external untrustworthy servers and remotely carry out various computations on the outsourced data at some later point in time by querying the server. *Functional encryption* (FE) is an ambitious vision of modern cryptography that attempts to preserve confidentiality of externally stored data while allowing entities to delegate computations on the outsourced data in such cloud computing platforms. FE supports "restricted" decryption keys, also known as "functional keys", that enable decrypters to learn specific functions of the encrypted data and nothing else. More precisely, in an FE scheme for certain function family $\mathcal{F}$, it is possible to derive functional keys SK_f for any function $f \in \mathcal{F}$ from a master secret key. Any party given such a functional key SK_f and a ciphertext CT_z encrypting some message z, should be able to learn $f(z)$ and nothing beyond that about z.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 164–195, 2016.
DOI: 10.1007/978-3-662-49384-7_7

A principle focus of research on FE has been to identify what class of functions $\mathcal{F}$ can be supported and what notion of security can be achieved. In terms of functionality, starting with the seminal notions of *identity-based encryption* (IBE) and *attribute-based encryption* (ABE), FE has progressively evolved through a series of distinguished works to support more and more expressive function families culminating into the recent state of the art schemes which are now able to realize computation of *arbitrary polynomial-size circuits* [6,7,10–12]. Regarding security, the vast majority of research on FE so far has concentrated on protecting privacy of the encrypted contents [6,15].

1.1 Function Privacy in Functional Encryption

A wide range of practical applications, however, demands not only privacy of the encrypted messages but also privacy of the functions for which functional keys are provided. This is especially desirable whenever the function embedded in the functional key itself contains sensitive informations.

Consider the following motivating scenario: Assume that a health organization subscribes to a cloud service provider to store medical records of its patients. To ensure confidentiality of informations, the organization encrypts those records locally using an FE scheme prior to uploading them to the cloud server. Now, using the inherent feature of FE, later on the organization can request the cloud server to perform some analysis on the encrypted records by providing the server the functional key for the respective function. However, if the FE scheme in use does not guarantee any hiding for the functions, which may include sensitive contents, embedded in the functional keys, then the functional keys might reveal the functions completely to the cloud, thereby leaking sensitive informations.

Private key vs public key setup: Countless real-life applications have driven the research on function privacy in the context of FE, using the private key setting first by Shen et al. [16] followed by the works of [2,8], while in the public key setting by Boneh et al. [4,5]. Intuitively, function privacy requires that functional keys reveal no unnecessary information on their functionality. However, the extent to which function privacy can be satisfied differs dramatically between the private key and public key regimes. Specifically, in the public key domain, where anyone can encrypt messages, only a limited form of function privacy can be attained. To formulate a meaningful security definition, a framework must assume that the functions come from a distribution having sufficient entropy [4,5]. On the contrary, in the private key setting, function privacy has been shown to have tremendously greater potential compared to the public key domain, both as a stand-alone feature and as a very useful building block.

Full-hiding security model for private key FE: For private key FE schemes, the *strongest* (indistinguishability-based) notion of function privacy, also known as *full-hiding* security, formulated in [2,8] considers both privacy of functional keys and privacy of encrypted data in a perfectly symmetric manner. More precisely, full-hiding security considers adversaries that interact with

(I) a left-or-right functional key generation oracle and
(II) a left-or-right encryption oracle,

where both oracles operate using the same bit $c \in \{0,1\}$. The adversaries submit a pair of functions $(f^{(j,0)}, f^{(j,1)})$ to the functional key generation oracle in order to make the j-th functional key query while they submit a pair of messages $(z^{(\ell,0)}, z^{(\ell,1)})$ to the encryption oracle for making the ℓ-th ciphertext query. Depending on the bit c, the functional key generation oracle returns the functional key $\mathrm{SK}_{f^{(j,c)}}$ whereas the encryption oracle sends back the ciphertext $\mathrm{CT}_{z^{(\ell,c)}}$. The adversaries are allowed to interact with these oracles for any polynomial number of queries and the adversaries' goal is to distinguish the cases $c = 0$ and $c = 1$. The constraint on the adversaries is that for all $(f^{(j,0)}, f^{(j,1)})$ and $(z^{(\ell,0)}, z^{(\ell,1)})$ with which they query the functional key generation and encryption oracles respectively, it should hold that $f^{(j,0)}(z^{(\ell,0)}) = f^{(j,1)}(z^{(\ell,1)})$. This is clearly the minimum necessary restriction as otherwise the adversaries can trivially determine the bit c used by the oracles.

Regarding the construction of function private FE schemes in the private key setting, recently Brakerski and Segev [8] have presented a generic transformation from any private key (possibly non-function-private) FE scheme for general polynomial-size circuits into one that achieves function privacy in the strongest model discussed above. Then by combining [8] with the works of [11,12], or [10], one can obtain private key function-private FE scheme supporting general circuits with strong security guarantee. However, the most significant drawback of the resulting constructions is that they would employ computationally intensive tools for secure computation such as fully homomorphic encryption or program obfuscation and their security would rely on strong assumptions such as indistinguishability obfuscation, extractability obfuscation, or polynomial hardness of simple assumptions on multilinear maps. Consequently, these solutions are far from being practical.

1.2 Inner Product Encryption and Function Privacy

A current motivation of cryptographic research community is to design direct and efficient FE schemes for functionalities of practical interest which are still expressive enough for real-life applications. As a first attempt, researchers have focused on the *inner product* functionality which is an extremely useful functionality in the context of descriptive statistics, for example, to compute the weighted mean of a collection of informations. Further, the inner product enables computation of conjunctions, disjunctions, polynomial evaluations, and exact thresholds.

An inner product function family $\mathcal{IP}_p$ is parameterized by a prime integer p. A function $\mathrm{IP}_{\vec{y}} \in \mathcal{IP}_p$ is associated with a vector $\vec{y} \in \mathbb{Z}_p^n$ of length n over the finite field $\mathbb{Z}_p$. On a message $\vec{x} \in \mathbb{Z}_p^n$, $\mathrm{IP}_{\vec{y}}(\vec{x})$ is defined to be the inner product $\langle \vec{x}, \vec{y} \rangle$ modulo p of the vectors $\vec{x}$ and $\vec{y}$. We stress that this formulation of inner-product FE, also referred to as *inner product encryption* (IPE) is distinct from [2,13,14,16] which study inner product in the context of predicate encryption (PE). In inner product PE, a message M is encrypted along with a tag $\vec{x} \in \mathbb{Z}_p^n$

and decryption with a key corresponding to a vector $\vec{y} \in \mathbb{Z}_p^n$ yields M if and only if $\langle \vec{x}, \vec{y} \rangle = 0$. In contrast, the objective in the IPE formulation is to learn the actual inner product value in $\mathbb{Z}_p$ itself.

The first construction of IPE was presented by Abdalla et al. [1] who developed a selectively secure construction in traditional discrete log groups. However, this construction is built in public key domain and do not support any form of function privacy. Very recently, Bishop et al. [3] have taken a first step forward towards exploring the possibility of attaining function privacy in the context of IPE utilizing efficient and well-studied primitives. In fact, they have constructed a function-private IPE scheme in private key domain that withstands any polynomial number of ciphertext and functional key queries. Their construction makes use of asymmetric bilinear pairing groups and derives its security from the well-studied Symmetric External Diffie-Hellman (SXDH) assumption albeit in a rather weak and unrealistic security model.

1.3 Our Contribution

The current state of the art leaves open the problem of constructing a private key IPE scheme achieving the strongest practical notion of full-hiding security under standard assumptions without employing any heavy-duty cryptographic tool. In this paper we provide a positive answer to this challenging problem. In particular, we develop a *simple* and *efficient* private key IPE scheme achieving the *strongest* notion of *function privacy* based on well-studied complexity assumption. As in [3], our construction utilizes asymmetric bilinear pairing groups of prime order and we are able to establish the stronger form of security under the SXDH assumption. In order to ensure correctness of our construction, like [1,3], we assume that the target inner products will be contained within a range of polynomial-size. As pointed out in [1,3], this assumption is quite reasonable for statistical applications, where, for instance, the average of some bounded quantity over a polynomial-size database will naturally be included in a polynomial range.

Although our construction has some resemblance to that of [3], we highlight several differences below:

– We innovate new technical ideas in order to realize the strongest notion of full-hiding security while maintaining the simplicity of the scheme. For all $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ and $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ with which the adversaries query the functional key generation and encryption oracles respectively, the security framework of [3] assumes that

$$\langle \vec{x}^{(\ell,0)}, \vec{y}^{(j,0)} \rangle = \langle \vec{x}^{(\ell,0)}, \vec{y}^{(j,1)} \rangle = \langle \vec{x}^{(\ell,1)}, \vec{y}^{(j,0)} \rangle = \langle \vec{x}^{(\ell,1)}, \vec{y}^{(j,1)} \rangle \tag{1}$$

whereas according to the full-hiding security framework of [2,8], the only constraint should be

$$\langle \vec{x}^{(\ell,0)}, \vec{y}^{(j,0)} \rangle = \langle \vec{x}^{(\ell,1)}, \vec{y}^{(j,1)} \rangle. \tag{2}$$

The additional restriction in the security model of [3] has not only weakened the security of their construction significantly but also it has rendered the security model itself rather unrealistic. Our security framework is free from any such restriction beyond that specified in Eq. (2), therefore, much more practical compared to that of [3].

- As in [3], we make use of the concept of *dual pairing vector spaces* (DPVS) introduced in [13,14] to obtain the features of hidden subspaces in prime order bilinear group setting. However, our two DPVS have dimensions $4n + 2$ and 6 respectively while those of [3] have dimensions $2n$ and 2 respectively. Here n is the dimension of vectors for functional keys and ciphertexts. This results in some loss in efficiency. However, this seems rather unavoidable for strengthening the security both from theoretical and practical point of view.
- Analogous to [3], we consider two pairs of *dual orthonormal bases*, one for each of the two dimensions considered. But instead of including the complete bases like [3], we put certain portions of them in the master secret key while preserve the remaining dimensions for the security reduction. Specifically, we employ $3n$ and 3 hidden dimensions of the pairs of bases of dimensions $4n + 2$ and 6 respectively to move things forward in our hybrid security argument.
- At a technical level, [3] used each component of the vectors *twice* while encoding the vectors in ciphertexts and functional keys by coupling them with the basis vectors included in the master secret key. On the contrary, in our construction, we utilize the components of these vectors only *once* in the process of encoding with the basis vectors of the master secret key.
- Although similar to [3], we treat ciphertexts and functional keys in a symmetric fashion in our construction, our hybrid security proof does not maintain any such symmetry. Specifically, the approach of [3] first established the privacy of encrypted messages in the multiple ciphertext framework and then leveraged the symmetry between the structures of ciphertexts and functional keys to flip the same reasoning to argue for function privacy. In doing so, they relied on an information theoretic step that required the additional constraint as in Eq. (1) on the queries of the adversaries. In order to remove the extra restriction, we face several challenges. For our security analysis, we design our hybrid argument differently using a different information theoretic property of DPVS proven by [13] in a non-trivial way. We begin our hybrid game transition by changing the form of the queried ciphertexts and instead of finishing it off completely, at some appropriate point, we initiate change in the queried functional keys. Since then the transformations of functional keys and ciphertexts proceed hand in hand.

2 Preliminaries

Throughout this paper we will follow notations presented in Fig. 1.

Symbol	Explanation
$\aleph \xleftarrow{\$} A$	$\aleph$ is randomly selected from A according to A's distribution, when A is a random variable, and $\aleph$ is uniformly selected from A, when A is a set.
$\vec{v}$	a vector $(v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ of length n for some positive integers p and n.
$\langle \vec{v}, \vec{w} \rangle$	the inner product of vectors $\vec{v}$ and $\vec{w} \in \mathbb{Z}_p^n$, i.e., $\sum_{i=1}^{n} v_i w_i \mod p$.
$g^{\vec{v}}$	an n-tuple of group elements, $(g^{v_1}, \ldots, g^{v_n}) \in \mathbb{G}^n$ for some cyclic group $\mathbb{G}$ of order p, where $\vec{v} \in \mathbb{Z}_p^n$ and $g \in \mathbb{G}$.
$g^{a\vec{v}}$	$(g^{av_1}, \ldots, g^{av_n}) \in \mathbb{G}^n$, where $a \in \mathbb{Z}_p$, $\vec{v} \in \mathbb{Z}_p^n$, and $g \in \mathbb{G}$.
$g^{\vec{v}+\vec{w}}$	$(g^{v_1+w_1}, \ldots, g^{v_n+w_n}) \in \mathbb{G}^n$, where $\vec{v}, \vec{w} \in \mathbb{Z}_p^n$ and $g \in \mathbb{G}$.
$\mathsf{GL}(n, \mathbb{Z}_p)$	the group of all $n \times n$ invertible matrices over $\mathbb{Z}_p$.

Fig. 1. Notations

2.1 The Notion of Private Key Function-Private IPE

We adopt the general notion of function-private functional encryption in the private key setting, introduced in [2,8], to the particular functionality of computing inner products of n-length vectors over $\mathbb{Z}_p$ for some prime integer p and some positive integer n. We will consider only non-zero vectors. Note that this is a reasonable consideration for all practical applications of inner products.

■ **Syntax:** A private key function-private IPE (PKFP-IPE) scheme consists of the following probabilistic polynomial-time algorithms:

PKFP-IPE.Setup$(1^\lambda, n)$: The data owner takes as input the security parameter 1^λ and a positive integer n (polynomial in λ) specifying the desired length of vectors for the functional keys and ciphertexts. It generates a master secret key MSK for itself while publishes public parameters PP. (Note that we are not dealing with a public key scheme, so PP are not sufficient to encrypt – those are just parameters that need not be kept secret.)

PKFP-IPE.Encrypt$(\text{MSK}, \text{PP}, \vec{x})$: On input the master secret key MSK, the public parameters PP, and a vector $\vec{x} \in \mathbb{Z}_p^n \backslash \{\vec{0}\}$, where $\vec{0}$ denotes the all zero vector in $\mathbb{Z}_p^n$, the data owner produces a ciphertext $\text{CT}_{\vec{x}}$.

PKFP-IPE.KeyGen$(\text{MSK}, \text{PP}, \vec{y})$: Taking as input the master secret key MSK, the public parameters PP, and a vector $\vec{y} \in \mathbb{Z}_p^n \backslash \{\vec{0}\}$, the data owner provides a functional key $\text{SK}_{\vec{y}}$ to a legitimate decrypter.

PKFP-IPE.Decrypt$(\text{PP}, \text{CT}_{\vec{x}}, \text{SK}_{\vec{y}})$: A decrypter takes as input the public parameters PP, a ciphertext $\text{CT}_{\vec{x}}$ encrypting some vector $\vec{x}$, and a functional key $\text{SK}_{\vec{y}}$ corresponding to some vector $\vec{y}$. It outputs either a value $m \in \mathbb{Z}_p$ or the distinguished symbol $\perp$.

■ **Correctness:** The correctness of an PKFP-IPE scheme requires the following: For all $\vec{x}, \vec{y} \in \mathbb{Z}_p^n \backslash \{\vec{0}\}$,

$$\mathsf{Pr}\big[(\mathrm{MSK}, \mathrm{PP}) \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.Setup}(1^\lambda, n); \mathrm{CT}_{\vec{x}} \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.Encrypt}(\mathrm{MSK}, \mathrm{PP}, \vec{x}); \\ \mathrm{SK}_{\vec{y}} \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.KeyGen}(\mathrm{MSK}, \mathrm{PP}, \vec{y}) : \\ \mathsf{PKFP\text{-}IPE.Decrypt}(\mathrm{PP}, \mathrm{CT}_{\vec{x}}, \mathrm{SK}_{\vec{y}}) = \langle \vec{x}, \vec{y} \rangle \big] > 1 - \epsilon(\lambda)$$

for some negligible function ϵ. As in [1,3], in our construction as well we would only require that the above holds when $\langle \vec{x}, \vec{y} \rangle$ is from a fixed polynomial range of values inside $\mathbb{Z}_p$.

■ **Security:** The indistinguishability-based full hiding security notion for a PKFP-IPE scheme is defined by the following game between a probabilistic adversary $\mathcal{A}$ and a probabilistic challenger $\mathcal{C}$:

Setup: $\mathcal{C}$ generates $(\mathrm{MSK}, \mathrm{PP}) \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.Setup}(1^\lambda, n)$. It gives PP to $\mathcal{A}$. It also selects $c \xleftarrow{\$} \{0, 1\}$.

Query Phase: Throughout the game, $\mathcal{A}$ may adaptively make any polynomial number of queries of the following two types:

- *Functional key query*: To make the j-th functional key query, $\mathcal{A}$ submits a pair of vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)}) \in \left(\mathbb{Z}_p^n \backslash \{\vec{0}\}\right)^2$ to $\mathcal{C}$. $\mathcal{C}$ creates a functional key $\mathrm{SK}^{(j)} \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.KeyGen}(\mathrm{MSK}, \mathrm{PP}, \vec{y}^{(j,c)})$ and hands $\mathrm{SK}^{(j)}$ to $\mathcal{A}$.
- *Ciphertext query*: To make the ℓ-th ciphertext query, $\mathcal{A}$ sends a pair of vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)}) \in \left(\mathbb{Z}_p^n \backslash \{\vec{0}\}\right)^2$ to $\mathcal{C}$. $\mathcal{C}$ forms $\mathrm{CT}^{(\ell)} \xleftarrow{\$} \mathsf{PKFP\text{-}IPE.Encrypt}(\mathrm{MSK}, \mathrm{PP}, \vec{x}^{(\ell,c)})$ and returns $\mathrm{CT}^{(\ell)}$ to $\mathcal{A}$.

Suppose that $\mathcal{A}$ makes q_1 number of functional key queries and q_2 number of ciphertext queries during the game. The restriction on the queries is that for all $j = 1, \ldots, q_1$ and for all $\ell = 1, \ldots, q_2$, $\langle \vec{x}^{(\ell,0)}, \vec{y}^{(j,0)} \rangle = \langle \vec{x}^{(\ell,1)}, \vec{y}^{(j,1)} \rangle$.

Guess: $\mathcal{A}$ eventually outputs a bit $c' \in \{0, 1\}$.

Let $\mathsf{View}_{\mathcal{A}}(c)$ denotes the view of $\mathcal{A}$ in the above game when the $c \in \{0, 1\}$ is the random bit selected by $\mathcal{C}$ in the setup phase.

Definition 1. *A* PKFP-IPE *is said to achieve (full) indistinguishability-based full hiding security if for any probabilistic polynomial-time adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above game,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PKFP-IPE}}(\lambda) = \left|\mathsf{Pr}\big[\mathcal{A}(\mathsf{View}_{\mathcal{A}}(0)) = 1\big] - \mathsf{Pr}\big[\mathcal{A}(\mathsf{View}_{\mathcal{A}}(1)) = 1\big]\right| < \epsilon(\lambda)$ *for some negligible function ϵ.*

2.2 Asymmetric Bilinear Group and SXDH Assumption

Definition 2 (Asymmetric Bilinear Pairing Group). *An asymmetric bilinear pairing group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime integer p;*

cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ *of order* p *each with polynomial-time computable group operations; generators* $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$*; and a polynomial-time computable non-degenerate bilinear pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*, i.e., e satisfies*

- *(bilinearity)* $e(g_1^s, g_2^{\check{s}}) = e(g_1, g_2)^{s\check{s}}$ *for all* $s, \check{s} \in \mathbb{Z}_p$ *and*
- *(non-degeneracy)* $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$*, where* $1_{\mathbb{G}_T}$ *denotes the identity element of the group* $\mathbb{G}_T$*.*

Let $\mathcal{G}_{\mathrm{ABPG}}$ *be an algorithm that on input the security parameter* 1^λ*, outputs a description* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ *of an asymmetric bilinear pairing group.*

Assumption 1 (Symmetric External Diffie-Hellman: SXDH). *The* SXDH *problem is to distinguish between the distributions* $\varrho_\beta = \big((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e), g_1^\mu, g_1^\nu, \Re_\beta, \big)$ *for* $\beta \in \{0,1\}$ *such that* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{\mathsf{ABPG}}(1^\lambda)$*,* $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$*, and* $\Re_\beta = g_1^{\mu\nu+r}$ *where* $r = 0$ *or* $r \xleftarrow{\$} \mathbb{Z}_p$ *according as* $\beta = 0$ *or 1 respectively.*

The SXDH *assumption states that for any probabilistic polynomial-time algorithm* $\mathcal{C}$*, for any security parameter* λ*,* $\mathsf{Adv}_{\mathcal{C}}^{\mathsf{SXDH}}(\lambda) = \big|\mathsf{Pr}\big[\mathcal{C}(\varrho_0) = 1\big] - \mathsf{Pr}\big[\mathcal{C}(\varrho_1) = 1\big]\big| < \epsilon(\lambda)$ *for some negligible function* ϵ*. It also states that the same is true for the analogous distributions obtained from switching the roles of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, i.e.,* $\check{\varrho}_\beta = \big((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e), g_2^{\check{\mu}}, g_2^{\check{\nu}}, \check{\Re}_\beta\big)$ *for* $\beta \in \{0,1\}$ *such that* $\check{\mu}, \check{\nu} \xleftarrow{\$} \mathbb{Z}_p$*, and* $\check{\Re}_\beta = g_2^{\check{\mu}\check{\nu}+\check{r}}$ *where* $\check{r} = 0$ *or* $\check{r} \xleftarrow{\$} \mathbb{Z}_p$ *according as* $\beta = 0$ *or 1 respectively.*

2.3 Dual Pairing Vector Spaces

Definition 3 (Dual Pairing Vector Spaces (DPVS)). *A dual pairing vector space* (DPVS) $(p, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, E)$ *by a direct product of asymmetric pairing groups* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ *is a tuple of a prime integer* p*; n-dimensional vector space* $\mathbb{V}_h = \mathbb{G}_h^n$ *over* $\mathbb{Z}_p$ *under vector addition* $\oplus$ *and scalar multiplication* $\otimes$ *defined respectively as* $g_h^{\vec{v}} \oplus g_h^{\vec{w}} = g_h^{\vec{v}+\vec{w}}$ *and* $a \otimes g_h^{\vec{v}} = g_h^{a\vec{v}}$*, for* $h = 1, 2$*, where* $\vec{v}, \vec{w} \in \mathbb{Z}_p^n$*, and* $a \in \mathbb{Z}_p$*; canonical bases* $\mathbb{A}_h = \{g_h^{\vec{e}_i}\}_{i=1,\ldots,n}$ *of* $\mathbb{V}_h$*, for* $h = 1, 2$*, where* $\vec{e}_i = (\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{n-i}) \in \mathbb{Z}_p^n$*; and a pairing* $E : \mathbb{V}_1 \times \mathbb{V}_2 \to \mathbb{G}_T$*. The pairing* E *is defined by* $E(g_1^{\vec{v}}, g_2^{\vec{w}}) = \prod_{i=1}^{n} e(g_1^{v_i}, g_2^{w_i}) = e(g_1, g_2)^{\langle \vec{v}, \vec{w} \rangle} \in \mathbb{G}_T$*, where* $\vec{v}, \vec{w} \in \mathbb{Z}_p^n$*. Observe that the map* E *is non-degenerate bilinear, i.e.,* E *satisfies*

- *(bilinearity)* $E(s \otimes g_1^{\vec{v}}, \check{s} \otimes g_2^{\vec{w}}) = E(g_1^{s\vec{v}}, g_2^{\check{s}\vec{w}}) = E(g_1^{\vec{v}}, g_2^{\vec{w}})^{s\check{s}}$ *for* $s, \check{s} \in \mathbb{Z}_p, \vec{v}, \vec{w} \in \mathbb{Z}_p^n$ *and*
- *(non-degeneracy) if* $E(g_1^{\vec{v}}, g_2^{\vec{w}}) = 1_{\mathbb{G}_T}$ *for all* $\vec{w} \in \mathbb{Z}_p^n$*, then* $\vec{v} = \vec{0}$*.*

When clear from the context, we will often omit the symbols $\oplus$ *and* $\otimes$ *for vector addition and scalar multiplication respectively in* DPVS*'s. The* DPVS *generation algorithm* $\mathcal{G}_{\mathsf{DPVS}}$ *takes input a positive integer* n *together with* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1,$

$g_2, e) \xleftarrow{\$} \mathcal{G}_{\mathsf{ABPG}}(1^\lambda)$ *and outputs a description* $(p, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, E)$ *of* DPVS *with* n*-dimensional vector spaces* $\mathbb{V}_h$ *for* $h = 1, 2$.

In Fig. 2 we describe random *dual orthonormal* basis generator $\mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^n)$ for some prime integer p and positive integer n. This algorithm would be utilized as a subroutine in our PKFP-IPE construction.

$\mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^n)$: This algorithm performs the following operations:

1. Choose $\boldsymbol{B} = (b_{i,j})_{i,j=1,\ldots,n} \xleftarrow{\$} \mathsf{GL}(n, \mathbb{Z}_p)$.
2. Compute $\boldsymbol{B}^* = (b^*_{i,j})_{i,j=1,\ldots,n} = (\boldsymbol{B}^\intercal)^{-1}$, where $\boldsymbol{B}^\intercal$ denotes transpose of the matrix $\boldsymbol{B}$. Let, $\vec{b}_i$ and $\vec{b}^*_i$ represent the i-th rows of $\boldsymbol{B}$ and $\boldsymbol{B}^*$ respectively, for $i = 1, \ldots, n$. Set $\mathbb{B} = \{\vec{b}_1, \ldots, \vec{b}_n\}$ and $\mathbb{B}^* = \{\vec{b}^*_1, \ldots, \vec{b}^*_n\}$. Note that $(\mathbb{B}, \mathbb{B}^*)$ are dual orthonormal in the sense that for $i, i' = 1, \ldots, n$,
$$\langle \vec{b}_i, \vec{b}^*_{i'} \rangle = \begin{cases} 1, \text{ if } i = i' \\ 0, \text{ otherwise} \end{cases}$$
3. Return $(\mathbb{B}, \mathbb{B}^*)$.

Fig. 2. Dual orthonormal basis generator $\mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^n)$

3 Our PKFP-IPE Scheme

■ **Construction:**

PKFP-IPE.Setup$(1^\lambda, n)$: The data owner takes as input the security parameter 1^λ and a positive integer n specifying the desired length of vectors for the keys and ciphertexts. It proceeds as follows:

1. It first generates an asymmetric bilinear group
$$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{\mathsf{ABPG}}(1^\lambda).$$
2. Then it forms
$$(p, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, E) \xleftarrow{\$} \mathcal{G}_{\mathsf{DPVS}}\big(4n + 2, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)\big) \text{ and}$$
$$(p, \mathbb{V}'_1, \mathbb{V}'_2, \mathbb{G}_T, \mathbb{A}'_1, \mathbb{A}'_2, E') \xleftarrow{\$} \mathcal{G}_{\mathsf{DPVS}}\big(6, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)\big).$$
3. Next, it samples dual orthonormal bases
$$\big(\mathbb{B} = \{\vec{b}_1, \ldots, \vec{b}_{4n+2}\}, \mathbb{B}^* = \{\vec{b}^*_1, \ldots, \vec{b}^*_{4n+2}\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2}) \text{ and}$$
$$\big(\mathbb{D} = \{\vec{d}_1, \ldots, \vec{d}_6\}, \mathbb{D}^* = \{\vec{d}^*_1, \ldots, \vec{d}^*_6\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^6).$$
It defines $\widehat{\mathbb{B}} = \{\vec{b}_1, \ldots, \vec{b}_n, \vec{b}_{4n+2}\}, \widehat{\mathbb{B}}^* = \{\vec{b}^*_1, \ldots, \vec{b}^*_n, \vec{b}^*_{4n+1}\}, \widehat{\mathbb{D}} = \{\vec{d}_1, \vec{d}_6\}$, and $\widehat{\mathbb{D}}^* = \{\vec{d}^*_1, \vec{d}^*_5\}$.

4. It keeps the master secret key $\text{MSK} = (\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$ to itself while publishes the public parameters $\text{PP} = \big(p, \{\mathbb{V}_h, \mathbb{V}'_h\}_{h=1,2}, \mathbb{G}_T, \{\mathbb{A}_h, \mathbb{A}'_h\}_{h=1,2}, E, E'\big)$.

PKFP-IPE.Encrypt($\text{MSK}, \text{PP}, \vec{x}$): Taking as input the master secret key MSK, the public parameters PP, and a vector $\vec{x} \in \mathbb{Z}_p^n \backslash \{\vec{0}\}$, the data owner prepares the ciphertext as follows:

1. It selects $\alpha, \xi, \xi_0 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\boldsymbol{c}_1 = g_1^{\alpha \sum_{i=1}^{n} x_i \vec{b}_i + \xi \vec{b}_{4n+2}} = g_1^{\alpha \sum_i x_i \vec{b}_i + \xi \vec{b}_{4n+2}}, \boldsymbol{c}_2 = g_1^{\alpha \vec{d}_1 + \xi_0 \vec{d}_6} \tag{3}$$

   utilizing $\widehat{\mathbb{B}}$ and $\widehat{\mathbb{D}}$ respectively from MSK, where a sum over index i ranges from $i = 1$ to $i = n$ unless explicitly specified otherwise. We will follow the same convention in the sequel as well.
2. It outputs the ciphertext $\text{CT}_{\vec{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$.

PKFP-IPE.KeyGen($\text{MSK}, \text{PP}, \vec{y}$): On input the master secret key MSK, the public parameters PP, and a vector $\vec{y} \in \mathbb{Z}_p^n \backslash \{\vec{0}\}$, the data owner performs the following:

1. It picks $\gamma, \eta, \eta_0 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\boldsymbol{k}_1^* = g_2^{\gamma \sum_i y_i \vec{b}_i^* + \eta \vec{b}_{4n+1}^*}, \boldsymbol{k}_2^* = g_2^{\gamma \vec{d}_1^* + \eta_0 \vec{d}_5^*} \tag{4}$$

   utilizing $\widehat{\mathbb{B}}^*$ and $\widehat{\mathbb{D}}^*$ respectively from MSK.
2. It provides the functional key $\text{SK}_{\vec{y}} = (\boldsymbol{k}_1^*, \boldsymbol{k}_2^*)$ to a legitimate decrypter.

PKFP-IPE.Decrypt($\text{PP}, \text{CT}_{\vec{x}}, \text{SK}_{\vec{y}}$): A decrypter takes as input the public parameters PP, a ciphertext $\text{CT}_{\vec{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$, and a functional key $\text{SK}_{\vec{y}} = (\boldsymbol{k}_1^*, \boldsymbol{k}_2^*)$. It proceeds as follows:

1. It computes $T_1 = E(\boldsymbol{c}_1, \boldsymbol{k}_1^*), T_2 = E'(\boldsymbol{c}_2, \boldsymbol{k}_2^*)$.
2. It then attempts to determine a value $m \in \mathbb{Z}_p$ such that $T_2^m = T_1$ as elements of $\mathbb{G}_T$ by checking a specified polynomial-size range of possible values. If it is successful, then it outputs m. Otherwise it outputs $\perp$.

We stress that the polynomial running time of our decryption algorithm is ensured by restricting the output to lie within a fixed polynomial-size range.

■ **Correctness:** The correctness of the above PKFP-IPE construction can be verified as follows: Observe that for any ciphertext $\text{CT}_{\vec{x}} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ encrypting some vector $\vec{x}$ and any functional key $\text{SK}_{\vec{y}} = (\boldsymbol{k}_1^*, \boldsymbol{k}_2^*)$ corresponding to some vector $\vec{y}$, we have

$$T_1 = E(\boldsymbol{c}_1, \boldsymbol{k}_1^*) = e(g_1, g_2)^{\alpha\gamma\langle \vec{x}, \vec{y} \rangle}, T_2 = E'(\boldsymbol{c}_2, \boldsymbol{k}_2^*) = e(g_1, g_2)^{\alpha\gamma}.$$

This follows from the expressions of $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{k}_1^*, \boldsymbol{k}_2^*$ together with the fact that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ are dual orthonormal bases. Thus if $\langle \vec{x}, \vec{y} \rangle$ is contained in the specified polynomial-size range of possible values that the decryption algorithm checks, it would output $\langle \vec{x}, \vec{y} \rangle$ as desired.

■ **Discussion:** In our PKFP-IPE construction, we begin with the intuition of [3] to use an asymmetric bilinear group setting $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, visualizing

$\mathbb{G}_1$ as the ciphertext space whereas $\mathbb{G}_2$ as the functional key space. The plaintext vectors are encrypted in the exponent of g_1 while the functional key vectors are encapsulated in the exponent of g_2, so that the bilinearity of the pairing e can be employed to compute the inner product of the plaintext and functional key vectors in the exponent without the explicit knowledge of the vectors.

As discussed earlier in this paper, the only PKFP-IPE scheme available in the literature so far [3] achieves a rather limited and unrealistic form of function privacy. In particular, for the sake of managing the hybrid security proof of their construction, they put further restrictions on the queries of the adversaries, as shown in Eq. (1), beyond those specified in the strongest framework of full-hiding security described in Sect. 2.1. This additional constraint not only leads to a weak security but it is also not conformal with the intuitive spirit of function privacy. With the motivation to remove such an undesirable restriction we recourse to an information theoretic step that uses a nice property of DPVS introduced in [13] that enables to hide a pair of ciphertext and functional key vectors perfectly among all vectors having the same inner product.

To generate space for our hybrid proof, we consider two pairs of dual orthonormal bases, namely, $(\mathbb{B}, \mathbb{B}^*)$ of dimension $4n + 2$ and $(\mathbb{D}, \mathbb{D}^*)$ of dimension 6, where n is the length of vectors for ciphertexts and functional keys. The $n + 2$ dimensions of the first pair of bases and 3 of the second pair are used in the actual scheme while the remaining dimensions are preserved to move things forward in the security proof. As displayed in Eq. (3), to encode a vector $\vec{x}$ in the ciphertext, we construct a linear combination of the first n vectors together with the $(4n+2)$-th vector of $\mathbb{B}$, where the n components of $\vec{x}$ masked with a random scalar α are used as coefficients of the first n vectors of $\mathbb{B}$. The resulting vector is then placed in the exponent of $g_1 \in \mathbb{G}_1$. After that, the randomness α is encoded by forming another linear combination of the first and sixth members of $\mathbb{D}$ in the exponent of g_1 using the masking factor α as coefficient of the first vector of $\mathbb{D}$. The $(4n + 2)$-th dimension of $\mathbb{B}$ and the sixth dimension of $\mathbb{D}$ are utilized to supply additional randomization for strengthening the security of our ciphertexts. The encoding of a vector for the functional key is performed in a directly symmetric fashion utilizing bases $\mathbb{B}^*, \mathbb{D}^*$, and $g_2 \in \mathbb{G}_2$ in place of $\mathbb{B}, \mathbb{D}$, and g_1 respectively, as can be seen from Eq. (4), where the additional randomization is provided by the $(4n + 1)$-th dimension of $\mathbb{B}^*$ and the fifth dimension of $\mathbb{D}^*$.

In contrast, the construction of [3] considers two pairs of dual orthonormal bases, one of dimension $2n$ and the other of dimension 2. Moreover, they make use of the complete bases in their construction itself and employ each component of a vector as coefficient twice during formation of the linear combinations in the process of encoding the vector for ciphertext or functional key, once for basis vectors in the range 1 to n and again for the basis vectors ranging from $n + 1$ to $2n$. Further, [3] rely on the orthogonality of all the queried functional key vectors (respectively all queried ciphertext vectors) to the difference of a pair of queried ciphertext vectors (respectively a pair of queried functional key vectors) to simulate a hidden dimension in the bases in the security proof that they employ to switch from one vector of the pair to the other. However, it

is precisely this approach which necessitates the additional constraint imposed by them on the adversaries' queries as in Eq. (1). Furthermore, increasing the dimensions of the DPVS's in use seems rather unavoidable for managing the security reduction without requiring the extra restriction. In fact the $3n$ and 3 hidden dimensions of our two pairs of bases respectively that we keep aside for the security argument play a vital role to elegantly isolate a pair of ciphertext and functional key vectors in an n-dimensional hidden subspace in order to apply our information theoretic argument.

In summery, although our construction has some kind of resemblance to that of [3], our proof idea is widely apart. The most significant contribution of our work lies in a rigorous proof of full-hiding security of a fairly simple construction. The detail security reduction is presented in the next section.

In terms of communication cum storage complexity, observe that both the ciphertexts and functional keys of our PKFP-IPE construction consist of $4n + 8$ group elements while our master secret key contains $8n^2 + 12n + 28$ members of the finite field $\mathbb{Z}_p$. In contrast, the ciphertexts and functional keys in the construction of [3] are comprised of $2n + 2$ group elements each whereas the master secret key is composed of $8n^2 + 8$ $\mathbb{Z}_p$ components.

Regarding computation complexity, note that both our encryption and functional key generation algorithms require $4n+8$ exponentiations while the decryption algorithm involves $4n+8$ pairing operations followed by an exhaustive search over a polynomial range of values in order to solve a discrete log. On the contrary, the encryption and functional key generation algorithms of [3] amount to $2n + 2$ exponentiations each. Other than a similar exhaustive search step, their decryption algorithm incurs $2n + 2$ pairings.

It is evident that our scheme loses a constant factor of 2 compared to that of [3] in both communication cum storage and computation efficiency. However, the additional cost is compensated with stronger and realistic data as well as function privacy guarantees provided by our construction as opposed to a rather limited form of security achieved by [3]. Given the rapid advancements in computing technology and the growing security breaches, high security is often desirable even at the expense of an admissible increase in complexity.

The ciphertexts and master public key of the only known IPE scheme in public key setup [1] involve $n+1$ and n elements respectively in a discrete log group of prime order p while the master secret key and functional keys are comprised of n and 1 $\mathbb{Z}_p$ components respectively. The encryption and decryption algorithms of [1] respectively incur $2n+1$ exponentiations and $n+1$ exponentiations followed by an analogous exhaustive search step towards determining a discrete log. However, the scheme of [1] offers no function privacy and, moreover, provides only selective data privacy.

4 Security Analysis

Theorem 1. *The* PKFP-IPE *scheme described in* Sect. 3 *is secure as per the security model of* Sect. 2.1 *under the* SXDH *assumption.*

Proof. The proof of Theorem 1 is structured as a hybrid argument over a series of games which differ in the construction of the functional keys and ciphertexts queried by the adversary $\mathcal{A}$ in the security game described in Sect. 2.1. In the first game, the queried functional keys and ciphertexts are constructed as those in the security game of Sect. 2.1 where the bit used by the challenger is $c = 0$. We then progressively change the functional keys and ciphertexts in multiple hybrid games to those in the security game of Sect. 2.1 where the bit used by the challenger is $c = 1$. We prove that each game is indistinguishable from the previous one, thus proving our PKFP-IPE construction to be secure in the security model of Sect. 2.1. Let q_1 be the number of $\mathcal{A}$'s functional key queries and q_2 the number of $\mathcal{A}$'s ciphertext queries. The hybrid game transition is described below. In these games, a portion of an exponent framed by a white box indicates those terms which were added or modified in a transition from the previous game, unless explicitly specified otherwise, while a part of an exponent which was deleted in the transformation from the earlier game is highlighted in the text.

■ **Sequence of Hybrid Games:**

⟨I⟩ **Game 0** : This game corresponds to the real security game of Sect. 2.1 where the bit used by the challenger to generate queried functional keys and ciphertexts is $c = 0$. More precisely, for $j = 1, \ldots, q_1$, the response to the j-th functional key query for vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ is created as $\text{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ such that

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \eta_j \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{d}_1^* + \eta_{j,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{5}$$

where $\gamma_j, \eta_j, \eta_{j,0} \xleftarrow{\$} \mathbb{Z}_p$. On the other hand, for $\ell = 1, \ldots, q_2$, the reply to the ℓ-th ciphertext query of $\mathcal{A}$ for vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ is generated as $\text{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ such that

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \xi_\ell \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}\right\} \tag{6}$$

where $\alpha_\ell, \xi_\ell, \xi_{\ell,0} \xleftarrow{\$} \mathbb{Z}_p$.

⟨II⟩ **Game 1 Sequence** [**Game 1-κ-1, . . . , Game 1-κ-4 ($\kappa = 1, \ldots, q_2$)**]

Game 1-κ-1: Game 1-0-4 coincides with Game 0. Game 1-κ-1 is the same as Game 1-$(\kappa - 1)$-4 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \boxed{\alpha''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{2n+i}} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \boxed{\alpha''_\kappa \vec{d}_3} + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{7}$$

where $\alpha_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 1-$(\kappa-1)$-4.

Game 1-κ-2: This game is identical to Game 1-κ-1 with the only exception that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha''_\kappa \sum_i \boxed{x_i^{(\kappa,1)}} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha''_\kappa \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{8}$$

where all the variables are generated as in Game 1-κ-1.

Game 1-κ-3: This game is analogous to Game 1-κ-2 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{2n+i} + \boxed{\alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i}} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha''_\kappa \vec{d}_3 + \boxed{\alpha'''_\kappa \vec{d}_4} + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{9}$$

where $\alpha'''_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 1-κ-2.

Game 1-κ-4: This game is the same as Game 1-κ-3 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{10}$$

where all the variables are generated as in Game 1-κ-3, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are modified from those in the last game by dropping the terms involving α''_κ in the exponent of g_1.

⟨III⟩ Game 2 Sequence [Game 2-ω-1, ..., Game 2-ω-6 ($\omega = 1, \ldots, q_1$)]

Game 2-ω-1: Game 2-0-6 coincides with Game 1-q_2-4. Game 2-ω-1 is the similar to Game 2-$(\omega-1)$-6 except that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \boxed{\gamma'_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{n+i}^* + \gamma''_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^*} + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \boxed{\gamma'_\omega \vec{d}_2^* + \gamma''_\omega \vec{d}_3^*} + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{11}$$

where $\gamma'_\omega, \gamma''_\omega \xleftarrow{\$} \mathbb{Z}_p$, and all the other variables are generated as in Game 2-$(\omega-1)$-6.

Sequence of Subgames of Game 2-ω-2[Game 2-ω-2-κ-1, ..., Game 2-ω-2-κ-5 ($\kappa = 1, \ldots, q_2$)]

Game 2-ω-2-κ-1: Game 2-ω-2-0-5 coincides with Game 2-ω-1. Game 2-ω-2-κ-1 is analogous to Game 2-ω-2-$(\kappa-1)$-5 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma'_\omega \sum_i \boxed{y_i^{(\omega,0)}} \vec{b}_{n+i}^* + \gamma''_\omega \sum_i \boxed{y_i^{(\omega,1)}} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*} \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma'_\omega \vec{d}_2^* + \gamma''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{12}$$

where all the variables are generated as in Game 2-ω-2-$(\kappa-1)$-5. Here a part of the exponent framed by a white box (respectively light gray box) indicates those terms which were changed in the transition from the previous game when $\kappa \geq 2$ (respectively $\kappa = 1$). More specifically, when $\kappa = 1$, $\boldsymbol{k}_1^{*(\omega)}$ in Eq. (12) is transformed from that in Eq. (11), which is the form of $\boldsymbol{k}_1^{*(\omega)}$ in Game 2-ω-2-0-5, by changing the portion of the exponent framed by a light gray box. On the other hand, when $\kappa \geq 2$, $\boldsymbol{k}_1^{*(\omega)}$ in Eq. (12) is obtained from that in Eq. (14), which is the form of $\boldsymbol{k}_1^{*(\omega)}$ in Game 2-ω-2-$(\kappa-1)$-5, by applying modification in the portion of the exponent framed by a white box.

Game 2-ω-2-κ-2: This game is identical to Game 2-ω-2-κ-1 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \boxed{\alpha'_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{n+i}} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}} \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \boxed{\alpha'_\kappa \vec{d}_2} + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{13}$$

where $\alpha'_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 2-ω-2-κ-1.

Game 2-ω-2-κ-3: This game is similar to Game 2-ω-2-κ-2 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma'_\omega \sum_i \boxed{y_i^{(\omega,1)}} \vec{b}_{n+i}^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*} \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma'_\omega \vec{d}_2^* + \gamma''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{14}$$

while the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha'_\kappa \sum_i \boxed{x_i^{(\kappa,1)}} \vec{b}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}} \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'_\kappa \vec{d}_2 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{15}$$

where all the variables are generated as in Game 2-ω-2-κ-2.

Game 2-ω-2-κ-4: This game is the same as Game 2-ω-2-κ-3 except that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$

are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\sum_i (\alpha_\kappa x_i^{(\kappa,0)} \vec{b}_i + \alpha'_\kappa x_i^{(\kappa,1)} \vec{b}_{n+i} + \boxed{\breve{\alpha}''_\kappa x_i^{(\kappa,1)} \vec{b}_{2n+i}} + \alpha'''_\kappa x_i^{(\kappa,1)} \vec{b}_{3n+i}) + \xi_\kappa \vec{b}_{4n+2}} \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'_\kappa \vec{d}_2 + \boxed{\breve{\alpha}''_\kappa \vec{d}_3} + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6} \end{aligned}\right\}, \tag{16}$$

where $\breve{\alpha}''_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 2-ω-2-κ-3.

Game 2-ω-2-κ-5: This game is analogous to Game 2-ω-2-κ-4 with the only exception that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \breve{\alpha}''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{2n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}} \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \breve{\alpha}''_\kappa \vec{d}_3 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6} \end{aligned}\right\}, \tag{17}$$

where all the variables are generated as in Game 2-ω-2-κ-4, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are transformed from those in the earlier game by removing the terms involving α'_κ in the exponent of g_1.

Game 2-ω-3: This game is identical to Game 2-ω-2-q_2-5 with the only exception that the components of the ω-th queried functional key for vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*} \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*} \end{aligned}\right\}, \tag{18}$$

where all the variables are generated as in Game 2-ω-2-q_2-5, i.e., in this game $\boldsymbol{k}_1^{*(\omega)}$ and $\boldsymbol{k}_2^{*(\omega)}$ are changed from those in the last game by deleting the terms involving γ'_ω in the exponent of g_2.

Game 2-ω-4: This game is the same as Game 2-ω-3 except that the components of the ω-th queried functional key for vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{2n+i}^* + \boxed{\gamma'''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{3n+i}^*} + \eta_\omega \vec{b}_{4n+1}^*} \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma''_\omega \vec{d}_3^* + \boxed{\gamma'''_\omega \vec{d}_4^*} + \eta_{\omega,0} \vec{d}_5^*} \end{aligned}\right\}, \tag{19}$$

where $\gamma'''_\omega \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 2-ω-3.

Game 2-ω-5: This game is similar to Game 2-ω-4 with the only exception that for $\ell = 1, \ldots, q_2$, the components of the ℓ-th queried ciphertext for vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}} \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6} \end{aligned}\right\}, \tag{20}$$

where all the variables are generated as in Game 2-ω-4, i.e., Eq. (20) resets $\boldsymbol{c}_1^{(\ell)}$ and $\boldsymbol{c}_2^{(\ell)}$, for $\ell = 1, \ldots, q_2$, as those in Eq. (10) by dropping the terms involving $\breve{\alpha}''_\ell$ in the exponent of g_1.

Game 2-ω-6: This game is the same as Game 2-ω-5 except that the components of the ω-th queried functional key for vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma'''_\omega \sum_i y_i^{(\omega,1)} \vec{b}^*_{3n+i} + \eta_\omega \vec{b}^*_{4n+1}}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma'''_\omega \vec{d}_4^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{21}$$

where all the variables are generated as in Game 2-ω-5, i.e., in this game $\boldsymbol{k}_1^{*(\omega)}$ and $\boldsymbol{k}_2^{*(\omega)}$ are changed from those in the earlier game by deleting the terms involving γ''_ω in the exponent of g_2.

⟨IV⟩ Game 3: This game is analogous to Game 2-q_1-6 except that for $j = 1, \ldots, q_1$, the components of the j-th queried functional key corresponding to vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i \boxed{y_i^{(j,1)}} \vec{b}_i^* + \gamma'''_j \sum_i \boxed{y_i^{(j,0)}} \vec{b}^*_{3n+i} + \eta_j \vec{b}^*_{4n+1}}, \\ \boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{d}_1^* + \gamma'''_j \vec{d}_4^* + \eta_{j,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{22}$$

while for $\ell = 1, \ldots, q_2$, the components of the ℓ-th queried ciphertext for vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i \boxed{x_i^{(\ell,1)}} \vec{b}_i + \alpha'''_\ell \sum_i \boxed{x_i^{(\ell,0)}} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}\right\} \tag{23}$$

where all the variables are generated as in Game 2-q_1-6.

⟨V⟩ Game 4 Sequence [Game 4-ω-1, ..., Game 4-ω-6 ($\omega = 1, \ldots, q_1$)]

Game 4-ω-1: Game 4-0-6 coincides with Game 3. Game 4-ω-1 is the same as Game 4-$(\omega-1)$-6 except that the components of the ω-th queried functional key for vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \boxed{\breve{\gamma}''_\omega \sum_i y_i^{(\omega,0)} \vec{b}^*_{2n+i}} + \gamma'''_\omega \sum_i y_i^{(\omega,0)} \vec{b}^*_{3n+i} + \eta_\omega \vec{b}^*_{4n+1}}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \boxed{\breve{\gamma}''_\omega \vec{d}_3^*} + \gamma'''_\omega \vec{d}_4^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{24}$$

where $\breve{\gamma}''_\omega \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 4-$(\omega-1)$-6.

Game 4-ω-2: This game is identical to Game 4-ω-1 with the only exception that for $\ell = 1, \ldots, q_2$, the components of the ℓ-th queried ciphertext corresponding

to vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,1)} \vec{b}_i + \boxed{\breve{\alpha}''_\ell \sum_i x_i^{(\ell,0)} \vec{b}_{2n+i}} + \alpha'''_\ell \sum_i x_i^{(\ell,0)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \boxed{\breve{\alpha}''_\ell \vec{d}_3} + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}\right\} \tag{25}$$

where $\breve{\alpha}''_\ell \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 4-ω-1.

Game 4-ω-3: This game is the same as Game 4-ω-2 with the only exception that the components of the ω-th queried functional key for vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \breve{\gamma}''_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \breve{\gamma}''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{26}$$

where all the variables are generated as in Game 4-ω-2, i.e., in this game $\boldsymbol{k}_1^{*(\omega)}$ and $\boldsymbol{k}_2^{*(\omega)}$ are transformed from those in the previous game by dropping the terms involving γ'''_ω in the exponent of g_2.

Game 4-ω-4: This game is analogous to Game 4-ω-3 except that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \boxed{\breve{\gamma}'_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{n+i}^*} + \breve{\gamma}''_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \boxed{\breve{\gamma}'_\omega \vec{d}_2^*} + \breve{\gamma}''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{27}$$

where $\breve{\gamma}'_\omega \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 4-ω-3.

Sequence of Subgames of Game 4-ω-5 [Game 4-ω-5-κ-1, ..., Game 4-ω-5-κ-5 ($\kappa = 1, \ldots, q_2$)]

Game 4-ω-5-κ-1: Game 4-ω-5-0-5 coincides with Game 4-ω-4. Game 4-ω-5-κ-1 is identical to Game 4-ω-5-($\kappa-1$)-5 except that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\sum_i (\alpha_\kappa x_i^{(\kappa,1)} \vec{b}_i + \boxed{\breve{\alpha}'_\kappa x_i^{(\kappa,0)} \vec{b}_{n+i}} + \breve{\alpha}''_\kappa x_i^{(\kappa,0)} \vec{b}_{2n+i} + \alpha'''_\kappa x_i^{(\kappa,0)} \vec{b}_{3n+i}) + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \boxed{\breve{\alpha}'_\kappa \vec{d}_2} + \breve{\alpha}''_\kappa \vec{d}_3 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{28}$$

where $\breve{\alpha}'_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 4-ω-5-($\kappa-1$)-5.

Game 4-ω-5-κ-2: This game is the same as Game 4-ω-5-κ-1 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \breve{\alpha}'_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \breve{\alpha}'_\kappa \vec{d}_2 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{29}$$

where all the variables are generated as in Game 4-ω-5-κ-1, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are changed from those in the last game by deleting the terms involving $\breve{\alpha}''_\kappa$ in the exponent of g_1.

Game 4-ω-5-κ-3: This game is similar to Game 4-ω-5-κ-2 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \breve{\gamma}'_\omega \sum_i \boxed{y_i^{(\omega,1)}} \vec{b}_{n+i}^* + \breve{\gamma}''_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \breve{\gamma}'_\omega \vec{d}_2^* + \breve{\gamma}''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{30}$$

while the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \breve{\alpha}'_\kappa \sum_i \boxed{x_i^{(\kappa,1)}} \vec{b}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \breve{\alpha}'_\kappa \vec{d}_2 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{31}$$

where all the variables are generated as in Game 4-ω-5-κ-2.

Game 4-ω-5-κ-4: This game is the same as Game 4-ω-5-κ-3 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \alpha'''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{32}$$

where all the variables are generated as in Game 4-ω-5-κ-3, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are transformed from those in the earlier game by removing the terms involving $\breve{\alpha}'_\kappa$ in the exponent of g_1.

Game 4-ω-5-κ-5: This game is analogous to Game 4-ω-5-κ-4 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\boldsymbol{k}_1^{*(\omega)} = \begin{cases} g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \breve{\gamma}'_\omega \sum_i \boxed{y_i^{(\omega,0)}} \vec{b}_{n+i}^* + \breve{\gamma}''_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, & \\ \qquad \text{if } \kappa \leq q_2 - 1 & \text{(33a)} \\ g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \breve{\gamma}'_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{n+i}^* + \breve{\gamma}''_\omega \sum_i \boxed{y_i^{(\omega,1)}} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, & \\ \qquad \text{if } \kappa = q_2 & \text{(33b)} \end{cases}$$

$$\boldsymbol{k}_2^{*(\omega)} = g_2^{\gamma_\omega \vec{d}_1^* + \breve{\gamma}'_\omega \vec{d}_2^* + \breve{\gamma}''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*} \tag{33c}$$

where all the variables are generated as in Game 4-ω-5-κ-4. Here a part of the exponent framed by a white box (respectively light gray box) indicates those terms which were changed from the previous game when $\kappa \leq q_2 - 1$ (respectively $\kappa = q_2$). More precisely, for $\kappa \leq q_2 - 1$, Eq. (33a) resets $\boldsymbol{k}_1^{*(\omega)}$ as in Eq. (27) by changing the portion of the exponent framed by a white box before executing the sequence of subgames Game 4-ω-5-κ-1 – Game 4-ω-5-κ-5 for the next value of κ. Equation (33b) modifies $\boldsymbol{k}_1^{*(\omega)}$ only once for $\kappa = q_2$ by applying change in the portion of the exponent framed by a light gray box and comes out of the sequence of subgames of Game 4-ω-5.

Game 4-ω-6: This game is the same as Game 4-ω-5-q_2-5 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,1)} \vec{b}_i^* + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{34}$$

where all the variables are generated as in Game 4-ω-5-q_2-5, i.e., in this game $\boldsymbol{k}_1^{*(\omega)}$ and $\boldsymbol{k}_2^{*(\omega)}$ are changed from those in the previous game by deleting the terms involving $\breve{\gamma}'_\omega$ and $\breve{\gamma}''_\omega$ in the exponent of g_2.

⟨VI⟩ Game 5 Sequence [Game 5-κ-1, ..., Game 5-κ-4 ($\kappa = 1, \ldots, q_2$)]

Game 5-κ-1: Game 5-0-4 coincides with Game 4-q_1-6. Game 5-κ-1 is similar to Game 5-$(\kappa-1)$-4 except that the components of the κ-th queried ciphertext for vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \boxed{\grave{\alpha}''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{2n+i}} + \alpha'''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \boxed{\grave{\alpha}''_\kappa \vec{d}_3} + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{35}$$

where $\grave{\alpha}''_\kappa \xleftarrow{\$} \mathbb{Z}_p$ and all the other variables are generated as in Game 5-$(\kappa-1)$-4.

Game 5-κ-2: This game is analogous to Game 5-κ-1 with the only exception that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \grave{\alpha}''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \grave{\alpha}''_\kappa \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{36}$$

where all the variables are generated as in Game 5-κ-1, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are modified from those in the last game by dropping the terms involving α'''_κ in the exponent of g_1.

Game 5-κ-3: This game is identical to Game 5-κ-2 except that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \grave{\alpha}''_\kappa \sum_i \boxed{x_i^{(\kappa,1)}} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \grave{\alpha}''_\kappa \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{37}$$

where all the variables are generated as in Game 5-κ-2.

Game 5-κ-4: This game is similar to Game 5-κ-3 with the only exception that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are computed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_i + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{38}$$

where all the variables are generated as in Game 5-κ-3, i.e., in this game $\boldsymbol{c}_1^{(\kappa)}$ and $\boldsymbol{c}_2^{(\kappa)}$ are changed from those in the earlier game by deleting the terms involving $\grave{\alpha}''_\kappa$ in the exponent of g_1. Note that in the final game, i.e., Game 5-q_2-4, all the queried functional keys $\text{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$, for $j = 1, \ldots, q_1$, and all the queried ciphertexts $\text{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$, for $\ell = 1, \ldots, q_2$, corresponds to functional keys and ciphertexts in the real security game of Sect. 2.1 where the bit used by the challenger is $c = 1$.

■ **Advantages of Adversary in Hybrid Games:** Denote $\mathsf{View}_{\mathcal{A}}^{(0)}$; $\mathsf{View}_{\mathcal{A}}^{(1\text{-}\kappa\text{-}h)}$, for $h = 1, \ldots, 4$; $\mathsf{View}_{\mathcal{A}}^{(2\text{-}\omega\text{-}h)}$, for $h = 1, 3, \ldots, 6$; $\mathsf{View}_{\mathcal{A}}^{(2\text{-}\omega\text{-}2\text{-}\kappa\text{-}h)}$, for $h = 1, \ldots, 5$; $\mathsf{View}_{\mathcal{A}}^{(3)}$; $\mathsf{View}_{\mathcal{A}}^{(4\text{-}\omega\text{-}h)}$, for $h = 1, \ldots, 4, 6$; $\mathsf{View}_{\mathcal{A}}^{(4\text{-}\omega\text{-}5\text{-}\kappa\text{-}h)}$, for $h = 1, \ldots, 5$; and $\mathsf{View}_{\mathcal{A}}^{(5\text{-}\kappa\text{-}h)}$, for $h = 1, \ldots, 4$ to be the views of the adversary $\mathcal{A}$ in Game 0; Game 1-κ-h, for $h = 1, \ldots, 4$; Game 2-ω-h, for $h = 1, 3, \ldots, 6$; Game 2-ω-2-κ-h, for $h = 1, \ldots, 5$; Game 3; Game 4-ω-h, for $h = 1, \ldots, 4, 6$; Game 4-ω-5-κ-h, for $h = 1, \ldots, 5$; and Game 5-κ-h, for $h = 1, \ldots, 4$ respectively. We define the advantage of $\mathcal{A}$ in Game ι as

$$\mathsf{Adv}_{\mathcal{A}}^{(\iota)}(\lambda) = \mathsf{Pr}\big[\mathcal{A}(\mathsf{View}_{\mathcal{A}}^{(\iota)}) = 1\big],$$

for $\iota \in \{0, 1\text{-}\kappa\text{-}h \ (h = 1, \ldots, 4), 2\text{-}\omega\text{-}h \ (h = 1, 3, \ldots, 6), 2\text{-}\omega\text{-}2\text{-}\kappa\text{-}h \ (h = 1, \ldots, 5), 3, \ 4\text{-}\omega\text{-}h \ (h = 1, \ldots, 4, 6), 4\text{-}\omega\text{-}5\text{-}\kappa\text{-}h \ (h = 1, \ldots, 5), 5\text{-}\kappa\text{-}h \ (h = 1, \ldots, 4)\}$.

To complete the proof of the theorem, we must show that the difference in the advantage of the adversary $\mathcal{A}$ between each pair of neighbouring games of the game sequence described above is at most negligible. Here, observe that the transition from Game 3 to Game 5-q_2-4 is actually the reverse of the transformation from Game 0 to Game 2-q_1-6 with the roles of $(\vec{x}_\ell^{(0)}, \vec{y}_j^{(0)})$ exchange with that of $(\vec{x}_\ell^{(1)}, \vec{y}_j^{(1)})$, for $j = 1, \ldots, q_1; \ell = 1, \ldots, q_2$. Therefore, it is sufficient to consider the transition from Game 0 to Game 3.

Indeed, in the full version of this paper [9] we have presented the complete sequence of arguments showing that the adversary $\mathcal{A}$ could experience at most a negligible difference in advantage between the neighbouring games from Game 0 to Game 3. Due to space consideration, in the next subsection we only provide those arguments which are significantly apart from one another and will demonstrate in detail our main technical ideas. Thus, it follows that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PKFP\text{-}IPE}}(\lambda) = \left|\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(5\text{-}q_2\text{-}4)}(\lambda)\right|$$

is negligible under the SXDH assumption. Hence the theorem. □

■ **Technically Distinguished Lemmas for Proof of Theorem 1:**

Lemma 1. *For any probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{C}_{1\text{-}1}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|Adv_{\mathcal{A}}^{(1\text{-}(\kappa-1)\text{-}4)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1\text{-}\kappa\text{-}1)}(\lambda)\right| \leq \mathsf{Adv}_{\mathcal{C}_{1\text{-}\kappa\text{-}1}}^{\mathsf{SXDH}}(\lambda)$, where $\mathcal{C}_{1\text{-}\kappa\text{-}1}(\cdot) = \mathcal{C}_{1\text{-}1}(\kappa, \cdot)$.*

Proof. Suppose that there is a probabilistic adversary $\mathcal{A}$ that achieves a non-negligible difference in advantage between Game 1-$(\kappa-1)$-4 and Game 1-κ-1. We construct a probabilistic algorithm $\mathcal{C}_{1\text{-}1}$ that attempts to decide the SXDH problem using $\mathcal{A}$ as a subroutine. $\mathcal{C}_{1\text{-}1}$ is given a positive integer κ and an instance of the SXDH problem $\varrho_\beta = \big((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e), g_1^{\mu}, g_1^{\nu}, \Re_\beta = g_1^{\mu\nu+r}\big)$, where $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$, and $r = 0$ or $r \xleftarrow{\$} \mathbb{Z}_p$ according as $\beta = 0$ or 1. $\mathcal{C}_{1\text{-}1}$ plays the role of the challenger in the security game of Sect. 2.1 and interacts with $\mathcal{A}$ as follows:

- $\mathcal{C}_{1\text{-}1}$ forms $(p, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, E) \xleftarrow{\$} \mathcal{G}_{\mathsf{DPVS}}\big(4n+2, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)\big)$ and $(p, \mathbb{V}'_1, \mathbb{V}'_2, \mathbb{G}_T, \mathbb{A}'_1, \mathbb{A}'_2, E') \xleftarrow{\$} \mathcal{G}_{\mathsf{DPVS}}\big(6, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)\big)$. Next, it samples dual orthonormal bases $\big(\mathbb{F} = \{\vec{f}_1, \ldots, \vec{f}_{4n+2}\}, \mathbb{F}^* = \{\vec{f}_1^*, \ldots, \vec{f}_{4n+2}^*\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ and $\big(\mathbb{H} = \{\vec{h}_1, \ldots, \vec{h}_6\}, \mathbb{H}^* = \{\vec{h}_1^*, \ldots, \vec{h}_6^*\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^6)$. It *implicitly* defines

$$\begin{array}{ll}
\vec{b}_i = \vec{f}_i + \mu\vec{f}_{2n+i}\,(i = 1, \ldots, n), & \vec{b}_i = \vec{f}_i\,(i = n+1, \ldots, 4n+2), \\
\vec{b}_{2n+i}^* = \vec{f}_{2n+i}^* - \mu\vec{f}_i^*\,(i = 1, \ldots, n), & \vec{b}_i^* = \vec{f}_i^*\,(i = 1, \ldots, 2n, 3n+1, \ldots, 4n+2), \\
\vec{d}_1 = \vec{h}_1 + \mu\vec{h}_3, & \vec{d}_i = \vec{h}_i\,(i = 2, \ldots, 6), \\
\vec{d}_3^* = \vec{h}_3^* - \mu\vec{h}_1^*, & \vec{d}_i^* = \vec{h}_i^*\,(i = 1, 2, 4, \ldots, 6).
\end{array}$$

It *implicitly* sets $\mathbb{B} = \{\vec{b}_1, \ldots, \vec{b}_{4n+2}\}, \mathbb{B}^* = \{\vec{b}_1^*, \ldots, \vec{b}_{4n+2}^*\}, \mathbb{D} = \{\vec{d}_1, \ldots, \vec{d}_6\}$, and $\mathbb{D}^* = \{\vec{d}_1^*, \ldots, \vec{d}_6^*\}$. Note that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ are dual orthonormal bases since those are obtained by applying an invertible linear transformation to the output of $\mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ and $\mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^6)$ respectively. For instance, observe that for $i = 1, \ldots, n$,

$$\langle\vec{b}_i, \vec{b}_{2n+i}^*\rangle = \overset{0}{\langle\vec{f}_i, \vec{f}_{2n+i}^*\rangle} - \mu\,\overset{1}{\langle\vec{f}_i, \vec{f}_i^*\rangle} + \mu\,\overset{1}{\langle\vec{f}_{2n+i}, \vec{f}_{2n+i}^*\rangle} - \mu^2\,\overset{0}{\langle\vec{f}_{2n+i}, \vec{f}_i^*\rangle} = 0,$$

$$\langle\vec{b}_i, \vec{b}_i^*\rangle = \overset{1}{\langle\vec{f}_i, \vec{f}_i^*\rangle} + \mu\,\overset{0}{\langle\vec{f}_{2n+i}, \vec{f}_i^*\rangle} = 1, \text{ etc.}$$

It hands the public parameters $\text{PP} = \big(p, \{\mathbb{V}_h, \mathbb{V}'_h\}_{h=1,2}, \mathbb{G}_T, \{\mathbb{A}_h, \mathbb{A}'_h\}_{h=1,2}, E, E'\big)$ to $\mathcal{A}$.

- In response to the j-th functional key query of $\mathcal{A}$ corresponding to vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$, for $j = 1, \ldots, q_1$, $\mathcal{C}_{1\text{-}1}$ chooses $\gamma_j, \eta_j, \eta_{j,0} \xleftarrow{\$} \mathbb{Z}_p$, computes

$$\begin{aligned}
\boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{f}_i^* + \eta_j \vec{f}_{4n+1}^*} = g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \eta_j \vec{b}_{4n+1}^*}, \\
\boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{h}_1^* + \eta_{j,0} \vec{h}_5^*} = g_2^{\gamma_j \vec{d}_1^* + \eta_{j,0} \vec{d}_5^*},
\end{aligned}$$

and gives the functional key $\text{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ to $\mathcal{A}$.

- In reply to $\mathcal{A}$'s ℓ-th ciphertext query corresponding to vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$, $\mathcal{C}_{1\text{-}1}$ proceeds as follows:

(a) $(\ell < \kappa)$ $\mathcal{C}_{1\text{-}1}$ picks $\alpha_\ell, \alpha_\ell''', \xi_\ell, \xi_{\ell,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{f}_i + \alpha_\ell''' \sum_i x_i^{(\ell,1)} \vec{f}_{3n+i} + \xi_\ell \vec{f}_{4n+2}} (g_1^{\mu})^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{f}_{2n+i}} \\
&= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha_\ell''' \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}}, \\
\boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{h}_1 + \alpha_\ell''' \vec{h}_4 + \xi_{\ell,0} \vec{h}_6} (g_1^{\mu})^{\alpha_\ell \vec{h}_3} = g_1^{\alpha_\ell \vec{d}_1 + \alpha_\ell''' \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}.
\end{aligned}$$

(b) $(\ell = \kappa)$ $\mathcal{C}_{1\text{-}1}$ selects $\xi_\kappa, \xi_{\kappa,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\boldsymbol{c}_1^{(\kappa)} &= (g_1^{\nu})^{\sum_i x_i^{(\kappa,0)} \vec{f}_i} (\Re_\beta)^{\sum_i x_i^{(\kappa,0)} \vec{f}_{2n+i}} g_1^{\xi_\kappa \vec{f}_{4n+2}} \\
&= g_1^{\nu \sum_i x_i^{(\kappa,0)} (\vec{f}_i + \mu \vec{f}_{2n+i}) + r \sum_i x_i^{(\kappa,0)} \vec{f}_{2n+i} + \xi_\kappa \vec{f}_{4n+2}} \\
&= g_1^{\nu \sum_i x_i^{(\kappa,0)} \vec{b}_i + r \sum_i x_i^{(\kappa,0)} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\
\boldsymbol{c}_2^{(\kappa)} &= (g_1^{\nu})^{\vec{h}_1} (\Re_\beta)^{\vec{h}_3} g_1^{\xi_{\kappa,0} \vec{h}_6} = g_1^{\nu(\vec{h}_1 + \mu \vec{h}_3) + r \vec{h}_3 + \xi_{\kappa,0} \vec{h}_6} = g_1^{\nu \vec{d}_1 + r \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}.
\end{aligned}$$

(c) $(\ell > \kappa)$ $\mathcal{C}_{1\text{-}1}$ chooses $\alpha_\ell, \xi_\ell, \xi_{\ell,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{f}_i + \xi_\ell \vec{f}_{4n+2}} (g_1^{\mu})^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{f}_{2n+i}} = g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \xi_\ell \vec{b}_{4n+2}}, \\
\boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{h}_1 + \xi_{\ell,0} \vec{h}_6} (g_1^{\mu})^{\alpha_\ell \vec{h}_3} = g_1^{\alpha_\ell \vec{d}_1 + \xi_{\ell,0} \vec{d}_6}.
\end{aligned}$$

$\mathcal{C}_{1\text{-}1}$ provides the ciphertext $\text{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ to $\mathcal{A}$.

- Finally, $\mathcal{A}$ outputs a bit c'. $\mathcal{C}_{1\text{-}1}$ outputs $\beta' = c'$.

Observe that if $\beta = 0$, i.e., $r = 0$, the κ-th answered ciphertext is of the form (Eq. 6), as in Game 1-$(\kappa - 1)$-4, where $\alpha_\kappa = \nu$. On the other hand, if $\beta = 1$, i.e., $r \xleftarrow{\$} \mathbb{Z}_p$, the κ-th answered ciphertext is of the form (Eq. 7), as in Game 1-κ-1, where $\alpha_\kappa = \nu$ and $\alpha''_\kappa = r$. Further, for $\ell < \kappa$, the ℓ-th answered ciphertext is of the form (Eq. 10) corresponding to Game 1-ℓ-4, which is its proper form in both Game 1-$(\kappa-1)$-4 and Game 1-κ-1 since the full sequence of transformations Game 1-ℓ-1 – Game 1-ℓ-4 has already been executed, whereas for $\ell > \kappa$, the ℓ-th

answered ciphertext is of the form (Eq. 6) corresponding to Game 0, which is its proper form since the sequence of transitions Game 1-ℓ-1 – Game 1-ℓ-4 has not yet been taken place. Additionally, for $j = 1, \ldots, q_1$, the j-th answered functional key is of the form (Eq. 5) corresponding to Game 0, which is its proper form since in the game transition so far no change is made in the form of the queried functional keys. Thus the view of $\mathcal{A}$ simulated by $\mathcal{C}_{1\text{-}1}$ is distributed as in Game 1-$(\kappa-1)$-4 or Game 1-κ-1 according as $\beta = 0$ or 1. This completes the proof of Lemma 1. □

Lemma 2. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ, $\mathsf{Adv}_{\mathcal{A}}^{(1\text{-}\kappa\text{-}1)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(1\text{-}\kappa\text{-}2)}(\lambda)$.*

Proof. In order to prove Lemma 2, we define an intermediate game, namely, Game 1-κ-1' as follows and show the equivalence of the distributions of the views of the adversary $\mathcal{A}$ in Game 1-κ-1 and that in Game 1-κ-1' (Claim 1) as well as those in Game 1-κ-2 and in Game 1-κ-1' (Claim 2).

Game 1-κ-1' ($\kappa = 1, \ldots, q_2$): This game is identical to Game 1-κ-1 with the only exception that the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha''_\kappa \sum_i \theta_i^{(\kappa)} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha''_\kappa \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{39}$$

where $\vec{\theta}^{(\kappa)} \xleftarrow{\$} \mathbb{Z}_p^n \backslash \{\vec{0}\}$ and all the other variables are generated as in Game 1-κ-1.

Claim 1. *The distribution of the view of the adversary $\mathcal{A}$ in* Game 1-κ-1 *and that in* Game 1-κ-1' *are equivalent.*

Proof. Consider the distribution of the view of $\mathcal{A}$ in Game 1-κ-1. We define new dual orthonormal bases $(\mathbb{U}, \mathbb{U}^*)$ of $\mathbb{Z}_p^{4n+2}$ using $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ below. We generate $\boldsymbol{M} \xleftarrow{\$} \mathsf{GL}(n, \mathbb{Z}_p)$ and define

$$\left.\begin{aligned} \begin{pmatrix} \vec{u}_{2n+1} \\ \vdots \\ \vec{u}_{3n} \end{pmatrix} &= \boldsymbol{M}^{-1} \cdot \begin{pmatrix} \vec{b}_{2n+1} \\ \vdots \\ \vec{b}_{3n} \end{pmatrix}, \quad \begin{pmatrix} \vec{u}^*_{2n+1} \\ \vdots \\ \vec{u}^*_{3n} \end{pmatrix} = \boldsymbol{M}^{\mathsf{T}} \cdot \begin{pmatrix} \vec{b}^*_{2n+1} \\ \vdots \\ \vec{b}^*_{3n} \end{pmatrix}, \\ \vec{u}_i &= \vec{b}_i, \qquad \vec{u}^*_i = \vec{b}^*_i, \\ & (i = 1, \ldots, 2n, 3n+1, \ldots, 4n+2). \end{aligned}\right\} \tag{40}$$

We set $\mathbb{U} = \{\vec{u}_1, \ldots, \vec{u}_{4n+2}\}, \mathbb{U}^* = \{\vec{u}^*_1, \ldots, \vec{u}^*_{4n+2}\}$. Note that $(\mathbb{U}, \mathbb{U}^*)$ are indeed dual orthonormal bases since those are obtained from the dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ by applying an invertible linear transformation. The components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are expressed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha''_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{2n+i} + \xi_\kappa \vec{b}_{4n+2}} \\ &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{u}_i + \alpha''_\kappa \sum_i \theta_i^{(\kappa)} \vec{u}_{2n+i} + \xi_\kappa \vec{u}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha''_\kappa \vec{d}_3 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{41}$$

where $\alpha_\kappa, \alpha''_\kappa, \xi_\kappa, \xi_{\kappa,0} \xleftarrow{\$} \mathbb{Z}_p$, and $\vec{\theta}^{(\kappa)} = \vec{x}^{(\kappa,0)} \cdot \boldsymbol{M}$.

Since $\vec{x}^{(\kappa,0)} \neq \vec{0}$ and $\boldsymbol{M}$ is uniformly selected from $\mathsf{GL}(n, \mathbb{Z}_p)$, $\vec{\theta}^{(\kappa)}$ is uniformly distributed in $\mathbb{Z}_p^n \backslash \{\vec{0}\}$ and it is independent from all the other variables. The components of any other ℓ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ are expressed as

(a) $(\ell < \kappa)$

$$\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}} \\ &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{u}_i + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{u}_{3n+i} + \xi_\ell \vec{u}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}$$

(b) $(\ell > \kappa)$

$$\boldsymbol{c}_1^{(\ell)} = g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \xi_\ell \vec{b}_{4n+2}} = g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{u}_i + \xi_\ell \vec{u}_{4n+2}}, \boldsymbol{c}_2^{(\ell)} = g_1^{\alpha_\ell \vec{d}_1 + \xi_{\ell,0} \vec{d}_6},$$

and for all $j = 1, \ldots, q_1$, the components of the j-th queried functional key for vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ are expressed as

$$\boldsymbol{k}_1^{*(j)} = g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \eta_j \vec{b}_{4n+1}^*} = g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{u}_i^* + \eta_j \vec{u}_{4n+1}^*}, \boldsymbol{k}_2^{*(j)} = g_2^{\gamma_j \vec{d}_1^* + \eta_{j,0} \vec{d}_5^*},$$

where all the variables are generated as in Game 1-κ-1.

Observe that in the light of the adversary $\mathcal{A}$'s view, both $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{U}, \mathbb{U}^*)$ are consistent with respect to PP. Also, this transformation of bases maintains the form (Eq. 5) of the j-th answered functional key $\mathrm{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ corresponding to Game 0, for $j = 1, \ldots, q_1$. Additionally, for $\ell < \kappa$, the ℓ-th answered ciphertext $\mathrm{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ preserves its form as in Eq. (10) corresponding to Game 1-ℓ-4 while for $\ell > \kappa$, $\mathrm{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ remains the same as in Eq. (6) corresponding to Game 0 under the basis transformation. Moreover, since the RHS of Eq. (41) and that of Eq. (39) are of the same form, the answered ciphertext $\mathrm{CT}^{(\kappa)} = (\boldsymbol{c}_1^{(\kappa)}, \boldsymbol{c}_2^{(\kappa)})$ is Game 1-κ-1 can be conceptually changed to that in Game 1-κ-1'. □

Claim 2. *The distribution of the view of adversary $\mathcal{A}$ in* Game 1-κ-2 *and that in* Game 1-κ-1' *are equivalent.*

Proof. Claim 2 is proven in a similar manner to Claim 1, using new dual orthonormal bases $(\mathbb{U}, \mathbb{U}^*)$ as in (Eq. 40). □

From Claims 1 and 2, it follows that adversary $\mathcal{A}$'s view in Game 1-κ-1 can be conceptually changed to that in Game 1-κ-2. This completes the proof of Lemma 2. □

Lemma 3. *For any probabilistic adversary $\mathcal{A}$, there exists a probabilistic algorithm $\mathcal{C}_{2\text{-}1}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $\left|\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(\omega-1)\text{-}6)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\omega\text{-}1)}(\lambda)\right| \leq \mathsf{Adv}_{\mathcal{C}_{2\text{-}\omega\text{-}1}}^{\mathsf{SXDH}}(\lambda)$, where $\mathcal{C}_{2\text{-}\omega\text{-}1}(\cdot) = \mathcal{C}_{2\text{-}1}(\omega, \cdot)$.*

Proof. Suppose that there is a probabilistic adversary $\mathcal{A}$ that achieves a non-negligible difference in advantage between Game 2-$(\omega-1)$-6 and Game 2-ω-1. We construct a probabilistic algorithm $\mathcal{C}_{2\text{-}1}$ that attempts to decide the SXDH problem using $\mathcal{A}$ as a subroutine. $\mathcal{C}_{2\text{-}1}$ is given a positive integer ω and an instance of the SXDH problem $\breve{\varrho}_\beta = \big((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e), g_2^{\breve{\mu}}, g_2^{\breve{\nu}}, \breve{\Re}_\beta = g_2^{\breve{\mu}\breve{\nu}+\breve{r}}\big)$, where $\breve{\mu}, \breve{\nu} \xleftarrow{\$} \mathbb{Z}_p$, and $\breve{r} = 0$ or $\breve{r} \xleftarrow{\$} \mathbb{Z}_p$ according as $\beta = 0$ or 1. $\mathcal{C}_{2\text{-}1}$ plays the role of the challenger in the security game of Sect. 2.1 and interacts with $\mathcal{A}$ as follows:

- The setup phase is executed by $\mathcal{C}_{2\text{-}1}$ in an analogous fashion as that performed by $\mathcal{C}_{1\text{-}1}$ in the proof of Lemma 1 except that $\mathcal{C}_{2\text{-}1}$ sets the dual orthonormal bases $\big(\mathbb{B} = \{\vec{b}_1, \ldots, \vec{b}_{4n+2}\}, \mathbb{B}^* = \{\vec{b}_1^*, \ldots, \vec{b}_{4n+2}^*\}\big)$ and $\big(\mathbb{D} = \{\vec{d}_1, \ldots, \vec{d}_6\}, \mathbb{D}^* = \{\vec{d}_1^*, \ldots, \vec{d}_6^*\}\big)$ *implicitly* from $\big(\mathbb{F} = \{\vec{f}_1, \ldots, \vec{f}_{4n+2}\}, \mathbb{F}^* = \{\vec{f}_1^*, \ldots, \vec{f}_{4n+2}^*\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ and $\big(\mathbb{H} = \{\vec{h}_1, \ldots, \vec{h}_6\}, \mathbb{H}^* = \{\vec{h}_1^*, \ldots, \vec{h}_6^*\}\big) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^6)$ respectively by selecting $\delta, \sigma \xleftarrow{\$} \mathbb{Z}_p$ and *implicitly* defining the following:

$$
\begin{array}{l}
\vec{b}_{n+i} = \vec{f}_{n+i} - \delta\breve{\mu}\vec{f}_i \,(i = 1, \ldots, n), \vec{b}_{2n+i} = \vec{f}_{2n+i} - \sigma\breve{\mu}\vec{f}_i \,(i = 1, \ldots, n), \\
\vec{b}_i = \vec{f}_i \,(i = 1, \ldots, n, 3n+1, \ldots, 4n+2), \\
\vec{b}_i^* = \vec{f}_i^* + \delta\breve{\mu}\vec{f}_{n+i}^* + \sigma\breve{\mu}\vec{f}_{2n+i}^* \,(i = 1, \ldots, n), \vec{b}_i^* = \vec{f}_i^* \,(i = n+1, \ldots, 4n+2), \\
\vec{d}_2 = \vec{h}_2 - \delta\breve{\mu}\vec{h}_1, \vec{d}_3 = \vec{h}_3 - \sigma\breve{\mu}\vec{h}_1, \vec{d}_i = \vec{h}_i \,(i = 1, 4, \ldots, 6), \\
\vec{d}_1^* = \vec{h}_1^* + \delta\breve{\mu}\vec{h}_2^* + \sigma\breve{\mu}\vec{h}_3^*, \vec{d}_i^* = \vec{h}_i^* \,(i = 2, \ldots, 6).
\end{array}
$$

- In response to the j-th functional key query of $\mathcal{A}$ corresponding to vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$, $\mathcal{C}_{2\text{-}1}$ proceeds as follows:

  (a) $(j < \omega)$ $\mathcal{C}_{2\text{-}1}$ picks $\gamma_j, \gamma_j''', \eta_j, \eta_{j,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$
\begin{aligned}
\boldsymbol{k}_1^{*(j)} &= g_2^{\sum_i (\gamma_j y_i^{(j,0)} \vec{f}_i^* + \gamma_j''' y_i^{(j,1)} \vec{f}_{3n+i}^*) + \eta_j \vec{f}_{4n+1}^*} \oplus \\
&\qquad (g_2^{\breve{\mu}})^{\sum_i (\delta\gamma_j y_i^{(j,0)} \vec{f}_{n+i}^* + \sigma\gamma_j y_i^{(j,0)} \vec{f}_{2n+i}^*)} \\
&= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \gamma_j''' \sum_i y_i^{(j,1)} \vec{b}_{3n+i}^* + \eta_j \vec{b}_{4n+1}^*}, \\
\boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{h}_1^* + \gamma_j''' \vec{h}_4^* + \eta_{j,0} \vec{h}_5^*} (g_2^{\breve{\mu}})^{\delta\gamma_j \vec{h}_2^* + \sigma\gamma_j \vec{h}_3^*} = g_2^{\gamma_j \vec{d}_1^* + \gamma_j''' \vec{d}_4^* + \eta_{j,0} \vec{d}_5^*}.
\end{aligned}
$$

  (b) $(j = \omega)$ $\mathcal{C}_{2\text{-}1}$ chooses $\eta_\omega, \eta_{\omega,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$
\begin{aligned}
\boldsymbol{k}_1^{*(\omega)} &= (g_2^{\breve{\nu}})^{\sum_i y_i^{(\omega,0)} \vec{f}_i^*} (\breve{\Re}_\beta)^{\sum_i (\delta y_i^{(\omega,0)} \vec{f}_{n+i}^* + \sigma y_i^{(\omega,0)} \vec{f}_{2n+i}^*)} g_2^{\eta_\omega \vec{f}_{4n+1}^*} \\
&= g_2^{\sum_i \left(\breve{\nu} y_i^{(\omega,0)} (\vec{f}_i^* + \delta\breve{\mu}\vec{f}_{n+i}^* + \sigma\breve{\mu}\vec{f}_{2n+i}^*) + \delta\breve{r} y_i^{(\omega,0)} \vec{f}_{n+i}^* + \sigma\breve{r} y_i^{(\omega,0)} \vec{f}_{2n+i}^*\right) + \eta_\omega \vec{f}_{4n+1}^*} \\
&= g_2^{\breve{\nu} \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \delta\breve{r} \sum_i y_i^{(\omega,0)} \vec{b}_{n+i}^* + \sigma\breve{r} \sum_i y_i^{(\omega,0)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, \\
\boldsymbol{k}_2^{*(\omega)} &= (g_2^{\breve{\nu}})^{\vec{h}_1^*} (\breve{\Re}_\beta)^{\delta\vec{h}_2^* + \sigma\vec{h}_3^*} g_2^{\eta_{\omega,0}\vec{h}_5^*} = g_2^{\breve{\nu}(\vec{h}_1^* + \delta\breve{\mu}\vec{h}_2^* + \sigma\breve{\mu}\vec{h}_3^*) + \delta\breve{r}\vec{h}_2^* + \sigma\breve{r}\vec{h}_3^* + \eta_{\omega,0}\vec{h}_5^*} \\
&= g_2^{\breve{\nu}\vec{d}_1^* + \delta\breve{r}\vec{d}_2^* + \sigma\breve{r}\vec{d}_3^* + \eta_{\omega,0}\vec{d}_5^*}.
\end{aligned}
$$

(c) $(j > \omega)$ $\mathcal{C}_{2\text{-}1}$ selects $\gamma_j, \eta_j, \eta_{j,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{f}_i^* + \eta_j \vec{f}_{4n+1}^*} (g_2^{\breve{\mu}})^{\delta\gamma_j \sum_i y_i^{(j,0)} \vec{f}_{n+i}^* + \sigma\gamma_j \sum_i y_i^{(j,0)} \vec{f}_{2n+i}^*} \\
&= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \eta_j \vec{b}_{4n+1}^*}, \\
\boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{h}_1^* + \eta_{j,0} \vec{h}_5^*} (g_2^{\breve{\mu}})^{\delta\gamma_j \vec{h}_2^* + \sigma\gamma_j \vec{h}_3^*} = g_2^{\gamma_j \vec{d}_1^* + \eta_{j,0} \vec{d}_5^*}.
\end{aligned}$$

$\mathcal{C}_{2\text{-}1}$ hands the functional key $\mathrm{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ to $\mathcal{A}$.

- In reply to the ℓ-th ciphertext query of $\mathcal{A}$ for vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$, for $\ell = 1, \ldots, q_2$, $\mathcal{C}_{2\text{-}1}$ selects $\alpha_\ell, \alpha_\ell''', \xi_\ell, \xi_{\ell,0} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}
\boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{f}_i + \alpha_\ell''' \sum_i x_i^{(\ell,1)} \vec{f}_{3n+i} + \xi_\ell \vec{f}_{4n+2}} \\
&= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha_\ell''' \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}}, \\
\boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{h}_1 + \alpha_\ell''' \vec{h}_4 + \xi_{\ell,0} \vec{h}_6} = g_1^{\alpha_\ell \vec{d}_1 + \alpha_\ell''' \vec{d}_4 + \xi_{\ell,0} \vec{d}_6},
\end{aligned}$$

and provides the ciphertext $\mathrm{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ to $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs a bit c'. $\mathcal{C}_{2\text{-}1}$ outputs $\beta' = c'$.

Observe that if $\beta = 0$, i.e., $\breve{r} = 0$, the ω-th answered functional key is of the form (Eq. 5), as in Game 2-$(\omega-1)$-6, where $\gamma_\omega = \breve{\nu}$. On the other hand, if $\beta = 1$, i.e., $\breve{r} \xleftarrow{\$} \mathbb{Z}_p$, the ω-th answered functional key is of the form (Eq. 11), as in Game 2-ω-1, where $\gamma_\omega = \breve{\nu}$, $\gamma_\omega' = \delta\breve{r}$, and $\gamma_\omega'' = \sigma\breve{r}$. Further, for $j < \omega$, the j-th answered functional key is of the form (Eq. 21) as in Game 2-j-6, which is its proper form in both Game 2-$(\omega-1)$-6 and Game 2-ω-1 since the sequence of transitions Game 2-j-1 – Game 2-j-6 has already been completed, whereas for $j > \omega$, the j-th answered functional key is of the form (Eq. 5) corresponding to Game 0, which is its proper form since during Game 1 sequence of transformations no change was made to the queried functional keys and the sequence of hybrids Game 2-j-1 – Game 2-j-6 has not yet been executed. Additionally, for $\ell = 1, \ldots, q_2$, the ℓ-th answered ciphertext is of the form (Eq. 10) as in Game 1-q_2-4, which is the proper form since for $\omega = 1$, no more alteration in the form of these ciphertexts has occurred after Game 1-q_2-4 and for $\omega \geq 2$, these ciphertexts have been reset to this form by Eq. (20) in Game 2-$(\omega-1)$-5. Thus the view of $\mathcal{A}$ simulated by $\mathcal{C}_{2\text{-}1}$ is distributed as in Game 2-$(\omega-1)$-6 or Game 2-ω-1 according as $\beta = 0$ or 1. This completes the proof of Lemma 3. □

Lemma 4. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ,* $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\omega\text{-}2\text{-}\kappa\text{-}2)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}\omega\text{-}2\text{-}\kappa\text{-}3)}(\lambda)$.

Proof. The proof of Lemma 4 utilizes the following result:

Lemma 5. (Lemma 3 in [13]). *For $\tau \in \mathbb{Z}_p$, let $\mathbb{S}_\tau = \{(\vec{\chi}, \vec{\vartheta}) \mid \langle \vec{\chi}, \vec{\vartheta} \rangle = \tau\} \subset \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, where p is a prime integer and n is some positive integer. For all $(\vec{\chi}, \vec{\vartheta}) \in \mathbb{S}_\tau$, for all $(\vec{\zeta}, \vec{\upsilon}) \in \mathbb{S}_\tau$,*

$$\mathsf{Pr}\left[\vec{\chi} \cdot \boldsymbol{F} = \vec{\zeta} \bigwedge \vec{\vartheta} \cdot \boldsymbol{F}^* = \vec{\upsilon}\right] = \mathsf{Pr}\left[\vec{\chi} \cdot \boldsymbol{F}^* = \vec{\zeta} \bigwedge \vec{\vartheta} \cdot \boldsymbol{F} = \vec{\upsilon}\right] = 1/\sharp\mathbb{S}_\tau,$$

where $\boldsymbol{F} \xleftarrow{\$} \mathsf{GL}(n, \mathbb{Z}_p)$, $\boldsymbol{F}^* = (\boldsymbol{F}^{\mathsf{T}})^{-1}$, *and for any set* A, $\sharp A$ *denotes the cardinality of the set* A.

In order to prove Lemma 4, we define an intermediate game, namely, Game 2-ω-2-κ-2' and show the equivalence of the distribution of the view of the adversary $\mathcal{A}$ in Game 2-ω-2-κ-2 and that in Game 2-ω-2-κ-2' (Claim 3) as well as those in Game 2-ω-2-κ-3 and in Game 2-ω-2-κ-2' (Claim 4).

Game 2-ω-2-κ-2' ($\omega = 1, \ldots, q_1$; $\kappa = 1, \ldots, q_2$): This game is similar to Game 2-ω-2-κ-2 with the only exception that the components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are formed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma'_\omega \sum_i \vartheta_i^{(\omega)} \vec{b}_{n+i}^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma'_\omega \vec{d}_2^* + \gamma''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{42}$$

while the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are created as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha'_\kappa \sum_i \chi_i^{(\kappa)} \vec{b}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'_\kappa \vec{d}_2 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{43}$$

such that $(\vec{\chi}^{(\kappa)}, \vec{\vartheta}^{(\omega)}) \xleftarrow{\$} \mathbb{S}_{\tau_{\omega,\kappa}} = \{(\vec{\chi}, \vec{\vartheta}) \mid \langle \vec{\chi}, \vec{\vartheta} \rangle = \tau_{\omega,\kappa}\} \subset \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, where $\tau_{\omega,\kappa} = \langle \vec{x}^{(\kappa,0)}, \vec{y}^{(\omega,0)} \rangle \big(= \langle \vec{x}^{(\kappa,1)}, \vec{y}^{(\omega,1)} \rangle$according to the restriction of the security game$\big)$, and all the other variables are generated as in Game 2-ω-2-κ-2.

Claim 3. *The distribution of the view of adversary* $\mathcal{A}$ *in* Game 2-ω-2-κ-2 *and that in* Game 2-ω-2-κ-2*' are equivalent.*

Proof. Consider the distribution of the view of $\mathcal{A}$ in Game 2-ω-2-κ-2. We define new dual orthonormal bases $(\mathbb{U}, \mathbb{U}^*)$ of $\mathbb{Z}_p^{4n+2}$ using $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ below. We generate $\boldsymbol{W} \xleftarrow{\$} \mathsf{GL}(n, \mathbb{Z}_p)$ and set

$$\left.\begin{aligned} \begin{pmatrix} \vec{u}_{n+1} \\ \vdots \\ \vec{u}_{2n} \end{pmatrix} &= \boldsymbol{W}^{-1} \cdot \begin{pmatrix} \vec{b}_{n+1} \\ \vdots \\ \vec{b}_{2n} \end{pmatrix}, \quad \begin{pmatrix} \vec{u}_{n+1}^* \\ \vdots \\ \vec{u}_{2n}^* \end{pmatrix} = \boldsymbol{W}^{\mathsf{T}} \cdot \begin{pmatrix} \vec{b}_{n+1}^* \\ \vdots \\ \vec{b}_{2n}^* \end{pmatrix}, \\ \vec{u}_i &= \vec{b}_i, \qquad \vec{u}_i^* = \vec{b}_i^*, \\ & (i = 1, \ldots, n, 2n+1, \ldots, 4n+2). \end{aligned}\right\} \tag{44}$$

We define $\mathbb{U} = \{\vec{u}_1, \ldots, \vec{u}_{4n+2}\}, \mathbb{U}^* = \{\vec{u}_1^*, \ldots, \vec{u}_{4n+2}^*\}$. Note that $(\mathbb{U}, \mathbb{U}^*)$ are indeed dual orthonormal bases since those are obtained from the dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ by applying an invertible linear transformation. The components of the ω-th queried functional key corresponding to vectors $(\vec{y}^{(\omega,0)}, \vec{y}^{(\omega,1)})$ are expressed as

$$\left.\begin{aligned} \boldsymbol{k}_1^{*(\omega)} &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{b}_i^* + \gamma'_\omega \sum_i y_i^{(\omega,0)} \vec{b}_{n+i}^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{b}_{2n+i}^* + \eta_\omega \vec{b}_{4n+1}^*} \\ &= g_2^{\gamma_\omega \sum_i y_i^{(\omega,0)} \vec{u}_i^* + \gamma'_\omega \sum_i \vartheta_i^{(\omega)} \vec{u}_{n+i}^* + \gamma''_\omega \sum_i y_i^{(\omega,1)} \vec{u}_{2n+i}^* + \eta_\omega \vec{u}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(\omega)} &= g_2^{\gamma_\omega \vec{d}_1^* + \gamma'_\omega \vec{d}_2^* + \gamma''_\omega \vec{d}_3^* + \eta_{\omega,0} \vec{d}_5^*}, \end{aligned}\right\} \tag{45}$$

while the components of the κ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\kappa,0)}, \vec{x}^{(\kappa,1)})$ are expressed as

$$\left.\begin{aligned} \boldsymbol{c}_1^{(\kappa)} &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_i + \alpha'_\kappa \sum_i x_i^{(\kappa,0)} \vec{b}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{b}_{3n+i} + \xi_\kappa \vec{b}_{4n+2}} \\ &= g_1^{\alpha_\kappa \sum_i x_i^{(\kappa,0)} \vec{u}_i + \alpha'_\kappa \sum_i \chi_i^{(\kappa)} \vec{u}_{n+i} + \alpha'''_\kappa \sum_i x_i^{(\kappa,1)} \vec{u}_{3n+i} + \xi_\kappa \vec{u}_{4n+2}}, \\ \boldsymbol{c}_2^{(\kappa)} &= g_1^{\alpha_\kappa \vec{d}_1 + \alpha'_\kappa \vec{d}_2 + \alpha'''_\kappa \vec{d}_4 + \xi_{\kappa,0} \vec{d}_6}, \end{aligned}\right\} \tag{46}$$

where $\gamma_\omega, \gamma'_\omega, \gamma''_\omega, \eta_\omega, \eta_{\omega,0}, \alpha_\kappa, \alpha'_\kappa, \alpha'''_\kappa, \xi_\kappa, \xi_{\kappa,0} \xleftarrow{\$} \mathbb{Z}_p$, and $\vec{\vartheta}_\omega = \vec{y}^{(\omega,0)} \cdot (\boldsymbol{W}^\intercal)^{-1}$, $\vec{\chi}^{(\kappa)} = \vec{x}^{(\kappa,0)} \cdot \boldsymbol{W}$.

From Lemma 5 it follows that $(\vec{\chi}^{(\kappa)}, \vec{\vartheta}^{(\omega)})$ are uniformly distributed in $\mathbb{S}_{\tau_{\omega,\kappa}}$, where $\langle \vec{x}^{(\kappa,0)}, \vec{y}^{(\omega,0)} \rangle = \tau_{\omega,\kappa}$, and are independent from all the other variables.

The components of any other j-th queried functional key corresponding to vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ are expressed as follows

(a) $(j < \omega)$

$$\begin{aligned} \boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \gamma'''_j \sum_i y_i^{(j,1)} \vec{b}_{3n+i}^* + \eta_j \vec{b}_{4n+1}^*} \\ &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{u}_i^* + \gamma'''_j \sum_i y_i^{(j,1)} \vec{u}_{3n+i}^* + \eta_j \vec{u}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{d}_1^* + \gamma'''_j \vec{d}_4^* + \eta_{j,0} \vec{d}_5^*}, \end{aligned}$$

(b) $(j > \omega)$

$$\begin{aligned} \boldsymbol{k}_1^{*(j)} &= g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \eta_j \vec{b}_{4n+1}^*} = g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{u}_i^* + \eta_j \vec{u}_{4n+1}^*}, \\ \boldsymbol{k}_2^{*(j)} &= g_2^{\gamma_j \vec{d}_1^* + \eta_{j,0} \vec{d}_5^*}, \end{aligned}$$

while the components of any other ℓ-th queried ciphertext corresponding to vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ are expressed as

(a) $(\ell < \kappa)$

$$\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \breve{\alpha}''_\ell \sum_i x_i^{(\ell,1)} \vec{b}_{2n+i} + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}} \\ &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{u}_i + \breve{\alpha}''_\ell \sum_i x_i^{(\ell,1)} \vec{u}_{2n+i} + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{u}_{3n+i} + \xi_\ell \vec{u}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \breve{\alpha}''_\ell \vec{d}_3 + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}$$

(b) $(\ell > \kappa)$

$$\begin{aligned} \boldsymbol{c}_1^{(\ell)} &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}} \\ &= g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{u}_i + \alpha'''_\ell \sum_i x_i^{(\ell,1)} \vec{u}_{3n+i} + \xi_\ell \vec{u}_{4n+2}}, \\ \boldsymbol{c}_2^{(\ell)} &= g_1^{\alpha_\ell \vec{d}_1 + \alpha'''_\ell \vec{d}_4 + \xi_{\ell,0} \vec{d}_6}, \end{aligned}$$

where all the variables are generated as in Game 2-ω-2-κ-2.

Observe that in the light of the adversary $\mathcal{A}$'s view, both $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{U}, \mathbb{U}^*)$ are consistent with respect to PP. Also, for $j < \omega$, the j-th answered functional key $\text{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ preserves its form as in Eq. (21) corresponding to Game 2-j-6 while for $j > \omega$, $\text{SK}^{(j)} = (\boldsymbol{k}_1^{*(j)}, \boldsymbol{k}_2^{*(j)})$ remains the same as in Eq. (5)

corresponding to Game 0, and for $\ell < \kappa$, the ℓ-th answered ciphertext $\mathrm{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ preserves its form as in Eq. (17) corresponding to Game 2-ω-2-ℓ-5 while for $\ell > \kappa$, $\mathrm{CT}^{(\ell)} = (\boldsymbol{c}_1^{(\ell)}, \boldsymbol{c}_2^{(\ell)})$ remains the same as in Eq. (10) corresponding to Game 1-q_2-4 (or equivalently of the form (Eq. 20) as in Game 2-$(\omega-1)$-5, for $\omega \geq 2$) under the basis transformation. Moreover, since the RHS of Eq. (45) (respectively Eq. (46)) and that of Eq. (42) (respectively Eq. (43)) are of the same form, the ω-th queried functional key $\mathrm{SK}^{(\omega)} = (\boldsymbol{k}_1^{*(\omega)}, \boldsymbol{k}_2^{*(\omega)})$ and the κ-th queried ciphertext $\mathrm{CT}^{(\kappa)} = (\boldsymbol{c}_1^{(\kappa)}, \boldsymbol{c}_2^{(\kappa)})$ in Game 2-ω-2-κ-2 can be conceptually changed to those in Game 2-ω-2-κ-2'. □

Claim 4. *The distribution of the view of the adversary $\mathcal{A}$ in* Game 2-ω-2-κ-3 *and that in* Game 2-ω-2-κ-2*' are equivalent.*

Proof. Claim 4 is proven in an analogous manner to Claim 3 using new dual orthonormal bases $(\mathbb{U}, \mathbb{U}^*)$ as in Eq. (44). □

From Claims 3 and 4 it follows that adversary $\mathcal{A}$'s view in Game 2-ω-2-κ-2 can be conceptually changed to that in Game 2-ω-2-κ-3. This completes the proof of Lemma 4. □

Lemma 6. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ,* $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}q_1\text{-}6)}(\lambda) = \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$.

Proof. In Game 2-q_1-6, for $j = 1, \ldots, q_1$, the components of the j-th queried functional key corresponding to vectors $(\vec{y}^{(j,0)}, \vec{y}^{(j,1)})$ have the form

$$\boldsymbol{k}_1^{*(j)} = g_2^{\gamma_j \sum_i y_i^{(j,0)} \vec{b}_i^* + \gamma_j''' \sum_i y_i^{(j,1)} \vec{b}_{3n+i}^* + \eta_j \vec{b}_{4n+1}^*}, \boldsymbol{k}_2^{*(j)} = g_2^{\gamma_j \vec{d}_1^* + \gamma_j''' \vec{d}_4^* + \eta_{j,0} \vec{d}_5^*},$$

as in Eq. (21), where $\gamma_j, \gamma_j''', \eta_j, \eta_{j,0} \xleftarrow{\$} \mathbb{Z}_p$, while for $\ell = 1, \ldots, q_2$, the components of the ℓ-th queried ciphertext for vectors $(\vec{x}^{(\ell,0)}, \vec{x}^{(\ell,1)})$ of the form

$$\boldsymbol{c}_1^{(\ell)} = g_1^{\alpha_\ell \sum_i x_i^{(\ell,0)} \vec{b}_i + \alpha_\ell''' \sum_i x_i^{(\ell,1)} \vec{b}_{3n+i} + \xi_\ell \vec{b}_{4n+2}}, \boldsymbol{c}_2^{(\ell)} = g_1^{\alpha_\ell \vec{d}_1 + \alpha_\ell''' \vec{d}_4 + \xi_{\ell,0} \vec{d}_6},$$

as in Eq. (20), where $\alpha_\ell, \alpha_\ell''', \xi_\ell, \xi_{\ell,0} \xleftarrow{\$} \mathbb{Z}_p$.

Therefore, by swapping the components of the dual orthonormal bases $\big(\mathbb{B} = \{\vec{b}_1, \ldots, \vec{b}_{4n+2}\}, \mathbb{B}^* = \{\vec{b}_1^*, \ldots, \vec{b}_{4n+2}^*\}\big)$ (respectively $\big(\mathbb{D} = \{\vec{d}_1, \ldots, \vec{d}_6\}, \mathbb{D}^* = \{\vec{d}_1^*, \ldots, \vec{d}_6^*\}\big)$) in the first block, i.e., in the range $i = 1, \ldots, n$ (respectively $i = 1$) and in the fourth block, i.e., in the range $i = 3n+1, \ldots, 4n$ (respectively $i = 4$), we obtain the distribution in Game 3. More precisely, we define new dual orthonormal bases $(\mathbb{U}, \mathbb{U}^*)$ of $\mathbb{Z}_p^{4n+2}$ and $(\mathbb{W}, \mathbb{W}^*)$ of $\mathbb{Z}_p^6$ using $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^{4n+2})$ and $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\mathbb{Z}_p^6)$ as follows: We set

$$\begin{aligned}
\vec{u}_{3n+i} &= \vec{b}_i, & \vec{u}_{3n+i}^* &= \vec{b}_i^* \ (i = 1, \ldots, n),\\
\vec{u}_i &= \vec{b}_{3n+i}, & \vec{u}_i^* &= \vec{b}_{3n+i}^* \ (i = 1, \ldots n),\\
\vec{u}_i &= \vec{b}_i, & \vec{u}_i^* &= \vec{b}_i^* \ (i = n+1, \ldots, 3n, 4n+1, 4n+2),\\
\vec{w}_4 &= \vec{d}_1, & \vec{w}_4^* &= \vec{d}_1^*, \qquad \vec{w}_1 = \vec{d}_4, \qquad \vec{w}_1^* = \vec{d}_4^*,\\
\vec{w}_i &= \vec{d}_i, & \vec{w}_i^* &= \vec{d}_i^* \ (i = 2, 3, 5, 6).
\end{aligned}$$

We define $\mathbb{U} = \{\vec{u}_1, \ldots, \vec{u}_{4n+2}\}, \mathbb{U}^* = \{\vec{u}^*_1, \ldots, \vec{u}^*_{4n+2}\}, \mathbb{W} = \{\vec{w}_1, \ldots, \vec{w}_6\}$, $\mathbb{W}^* = \{\vec{w}^*_1, \ldots, \vec{w}^*_6\}$. It is clear that $(\mathbb{U}, \mathbb{U}^*)$ and $(\mathbb{W}, \mathbb{W}^*)$ are indeed dual orthonormal bases since those are obtained from the dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ respectively by means of invertible linear transformations.

Observe that in light of the adversary $\mathcal{A}$'s view, both $(\mathbb{B}, \mathbb{B}^*)$ (respectively $(\mathbb{D}, \mathbb{D}^*)$) and $(\mathbb{U}, \mathbb{U}^*)$ (respectively $(\mathbb{W}, \mathbb{W}^*)$) are consistent with respect to PP. Moreover, it readily follows that the components of the queried functional keys and ciphertexts in $\mathsf{Game}\ 2\text{-}q_1\text{-}6$ over bases $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ are expressed as those in Eqs. (22) and (23) of $\mathsf{Game}\ 3$ over bases $(\mathbb{U}, \mathbb{U}^*)$ and $(\mathbb{W}, \mathbb{W}^*)$. This completes the proof of Lemma 6. □

5 Conclusion

In this paper, we have presented the *first non-generic* private key FE scheme for the inner product functionality achieving the *strongest indistinguishability-based* notion of function privacy, namely, the full-hiding security [2,8]. Our construction has utilized the standard asymmetric bilinear pairing group of prime order and has derived its security from the SXDH assumption. A significant future direction of research in this area would be to explore *simulation-based* notion of function privacy [2] in the context of IPE in the private key setting.

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015)
2. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: Function private functional encryption and property preserving encryption: new definitions and positive results. Technical report, Cryptology ePrint Archive, Report 2013/744 (2013)
3. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. Technical report, Cryptology ePrint Archive, Report 2015/672 (2015)
4. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013)
5. Boneh, D., Raghunathan, A., Segev, G.: Function-private subspace-membership encryption and its applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 255–275. Springer, Heidelberg (2013)
6. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
7. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014)
8. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015)

9. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. Technical report, Cryptology ePrint Archive, Report 2015/1255 (2015)
10. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. Technical report, Cryptology ePrint Archive, Report 2014/666 (2014)
11. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 40–49. IEEE (2013)
12. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 555–564. ACM (2013)
13. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
14. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
15. O'Neill, A.: Definitional issues in functional encryption. Technical report, Cryptology ePrint Archive, Report 2010/556 (2010)
16. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)

Deniable Functional Encryption

Angelo De Caro[1(✉)], Vincenzo Iovino[2], and Adam O'Neill[3]

[1] IBM Research Zurich, Rüschlikon, Switzerland
angelo.decaro@gmail.com
[2] University of Luxembourg, Luxembourg, Luxembourg
vinciovino@gmail.com
[3] Georgetown University, Washington, D.C., USA
amoneill@gmail.com

Abstract. *Deniable encryption*, first introduced by Canetti *et al.* [14], allows a sender and/or receiver of encrypted communication to produce fake but authentic-looking coins and/or secret keys that "open" the communication to a different message. Here we initiate its study for the more general case of *functional encryption* (FE), as introduced by Boneh *et al.* [12], wherein a receiver in possession of a key k can compute from any encryption of a message x the value $F(k, x)$ according to the scheme's functionality F. Our results are summarized as follows: We put forth and motivate the concept of deniable FE, for which we consider two models. In the first model, as previously considered by O'Neill *et al.* [31] in the case of identity-based encryption, a receiver gets assistance from the master authority to generate a fake secret key. In the second model, there are "normal" and "deniable" secret keys, and a receiver in possession of a deniable secret key can produce a fake but authentic-looking normal key on its own. This parallels the "multi-distributional" model of deniability previously considered for public-key encryption.

In the first model, we show that any FE scheme for the general circuit functionality (as several recent candidate construction achieve) can be converted into an FE scheme having receiver deniability, without introducing any additional assumptions. In addition we show an efficient receiver deniable FE for Boolean Formulae from bilinear maps. In the second (multi-distributional) model, we show a specific FE scheme for the general circuit functionality having receiver deniability. This result additionally assumes differing-inputs obfuscation and relies on a new technique we call *delayed trapdoor circuits*. To our knowledge, a scheme in the multi-distributional model was not previously known even in the simpler case of identity-based encryption.

Finally, we show that receiver deniability for FE implies some form of simulation security, further motivating study of the latter and implying optimality of our results.

Keywords: Deniable encryption · Functional encryption · Simulation security

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 196–222, 2016.
DOI: 10.1007/978-3-662-49384-7_8

1 Introduction

Encryption schemes meeting standard security notions (*e.g.*, semantic security [22]) may be *committing* in the sense that they tie the sender and receiver to having communicated a particular message. This is potentially damaging in the context of coercion, whereby for example the receiver's secret key becomes revealed (say under subpoena). Deniable encryption, formalized by Canetti *et al.* in 1997 [14], mitigates this threat by allowing the sender and/or receiver, after having already exchanged an encrypted message, to produce fake but authentic-looking random coins that "open" the ciphertext to a different message. That is, they can produce random coins (from which in particular a secret key can be computed) that make it look like they communicated some other message. The study of deniable encryption has seen a renewed interest. In particular, O'Neill *et al.* [31] construct "bideniable" public-key encryption schemes, namely where the sender and receiver can simultaneously equivocate without coordination, albeit in a relaxed, "multidistributional" model where there are special "deniable" algorithms that the sender and receiver must run in order to later be able to do so (in which case it looks like they ran the "normal" prescribed algorithms all along). Following [14,31], we call schemes where only the sender can equivocate "sender deniable," where only the receiver can equivocate "receiver deniable," and where both can equivocate "bideniable". Bendlin *et al.* [8] show that (non-interactive) receiver-deniable public-key encryption is *impossible* unless one works in the multidistributional model. Finally, a recent breakthrough work of Sahai and Waters [32] constructs sender-deniable public-key encryption *without* relying on the multidistributional model.

Deniability for Functional Encryption. In this paper, we initiate the study of deniability for much more advanced cryptosystems, as captured under the umbrella concept of *functional encryption* (FE) [12]. (Deniability for identity-based encryption was also previously considered by [31].) Whereas in traditional public-key encryption decryption is an all-or-nothing affair (*i.e.*, a receiver is either able to recover the entire message using its key, or nothing), in FE is possible to finely control the amount of information that is revealed by a ciphertext to a given receiver. Somewhat more precisely, in a functional encryption scheme for functionality F, each secret key (generated by a master authority) is associated with some value k. Anyone can encrypt via the public parameters. When a ciphertext Ct_x that encrypts x is decrypted using a secret key Sk_k for value k, the result is $F(k, x)$. Intuitively, security requires that a receiver in possession of Sk_k learns nothing beyond this. We contend that deniability is an important property to consider in the context of FE. For example, consider a large organization using FE in which members have different keys. Suppose the police coerces one of the members of the organization into revealing its key or requesting a key for some value k. A deniable FE scheme in the sense we consider would allow this member to provide the police with a key $\widetilde{SK}_k$ that "opens" a ciphertext Ct_x as above to any apparent value of $F(k, x)$ it likes. Another interesting application would be an encrypted email server that uses FE, where the server is in

possession of keys that allow it to do searches, spam filtering, targeted advertising, *etc.* If the government coerces the email server to provide its keys or requests additional keys from the client, the server can do so in a way that again reveals any apparent values it likes. As another scenario, consider a secure routing protocol implemented with FE, where any node receives an encrypted packet and using a secret key corresponding to its routing table can forward the packet to the right port without knowing the next destinations of the packet. The NSA could coerce the nodes to reveal their respective routing tables to trace the final destinations of the packet. If the FE system is receiver deniable, there is no reason for the NSA to coerce them as the nodes could reveal a fake secret key.

Model and Definitions. More specifically, we propose the concept of *receiver-deniable FE*. For intuition, suppose the coercer has observed Ct_x as above. Informally, receiver-deniable FE allows the sender to produce "fake" secret key Sk'_k (we assume the coercer knows k) that makes equivocate Ct_x as encryption of any other x' so that the secret key decrypts to $F(k, x')$. But this intuition for the definition hides several points. First, what if the coercer is able to coerce many receivers, thus seeing many secret keys? In the case of identity-based encryption, it was previously observed by O'Neill *et al.* [31] that this case is equivalent via a hybrid argument to equivocation of a single ciphertext and secret key. However, this hybrid argument fails in our more general setting. Therefore, in our modeling we consider what we call (n_c, n_k)-*receiver-deniability*, meaning the coercer requests n_c challenge ciphertexts (for which no underlying randomness is revealed) and n_k secret keys (i.e., receiver-coerce queries) adaptively. Second, and more interesting, O'Neill *et al.* [31] noted that an impossibility result of [8] implies that, even in the simpler case of identity-based encryption, "full" receiver deniability inherently requires that a receiver get assistance from the master authority to produce a fake secret key. While this may seem like the strongest possible model (indeed, [31] conveys the intuition that it is necessary for deniability), we propose an alternative, "multi-distributional" model as well, where there are "normal" and "deniable" secret keys, and a receiver in possession of a deniable secret key can produce a fake but authentic-looking normal one without any assistance. Here we envision that a user tells the authority initially whether it wants a normal or deniable secret key, and can later claim to a coercer that it requested a normal one even if it did not. We consider both models for deniable FE in this work. Note that the models are incomparable: on the one hand, the first ("full") model requires the receiver to get assistance from the master authority, but a receiver does not have to choose one or the other type of key to request initially as in the second ("multi-distributional") model. Getting assistance from the master authority to equivocate may not be feasible in many cases, making the second model particularly compelling, especially in light of the arguments of [31] for the meaningfulness of the multi-distributional model in the basic case of public-key encryption.

"Full" Receiver Deniability from Trapdoor Circuits. Next we show how to transform any "IND-secure" FE scheme for general circuits (*i.e.*, where its functionality F computes general boolean circuits on some input length) into a

FE for the same functionality that is (n_c, poly)-receiver-deniable in the full model (but where the receiver gets assistance from the master authority to equivocate) without introducing any additional assumption. In particular, recent works [13,19,21,34] show IND-secure FE for general circuits whose security is based either on indistinguishable obfuscation and its variants or polynomial hardness of simple assumptions on multi-linear maps. We can use any of these schemes in our construction. We present a direct black-box transformation, making use of the "trapdoor circuits" technique, introduced by De Caro *et al.* [17] to show how to bootstrap IND-secure for circuits to the stronger notion of simulation security (SIM-security). The idea of the trapdoor mechanism is to replace the original circuit C with a trapdoor circuit $\mathsf{Trap}[C]$ that the *receiver faking algorithm* can then use to program the output in some way.

To give some intuition, let us consider for simplicity the case of equivocating a single ciphertext and secret key. Then, a plaintext will have two slots where the first slot will be the actual message x. The second slot will be a random string s, some sort of *tag* used to identify the ciphertext. On the other hand, $\mathsf{Trap}[C]$, where for simplicity we restrict C to be one-bit output circuit, will have two slots embedded in it, let us call them trapdoor values. Both the slots will be random strings r_1, r_2 used as formal variables to represent Boolean values 0 and 1. Now, if it happens that $s = r_1$ then $\mathsf{Trap}[C]$ returns 0, if $s = r_2$ then it returns 1, otherwise $\mathsf{Trap}[C]$ returns $C(x)$. Notice that, when s, r_1 and r_2 are chosen uniformly and independently at the random then the above events happen with negligible probability thus this trapdoor mechanism does not influence the correctness of the scheme. On the other hand, it is easy to see how the receiver faking algorithm works by setting r_1 or r_2 to s depending on the expected faked output. Clearly, the receiver needs the master authority to generate a new secret key, corresponding to circuit $\mathsf{Trap}[C]$, with tailored embedded r_1 and r_2. Moreover, the above solution fails when more secret keys have to be equivocated. In fact, s then would appear in all the faked secret keys and this would be easily recognizable by the adversary. A trivial fix is to put in the ciphertexts as many different s's as the number of secret keys to be faked but this will create an unnecessary dependence that can be removed by using a PRF as a compact source of randomness. In Sect. 3 we present the result in full details.

Efficient Receiver Deniable FE for Boolean Formulae. We explore the possibility of achieving receiver deniability for weaker classes of functionalities that still support some form of trapdoor mechanism and for which a functional encryption scheme can be constructed assuming standard assumptions. We show how to do this for Boolean formulae, namely we show how to transform any IND-secure FE scheme for Boolean formulate into one that is (n_c, n_d)-receiver deniable. Note that Katz, Sahai and Waters [27] show how to construct an FE scheme for Boolean formulae given an FE scheme for the inner-product predicate whose security, by the result of Okamoto and Takashima [29], can be based on the Decisional Linear Assumption in bilinear groups. An interesting point, however, is that these schemes for boolean formulae allow polynomials in t variables with degree at most d in each variable, as long as d^t is polynomial

in the security parameter. This will mean that in order for our scheme to be efficient the trapdoor mechanism will have a non-negligible probability of being activated by an honest encryption (i.e., completeness is non-negligible). We fix this issue by using parallel repetition. The resulting scheme is (n_c, n_k)-receiver deniable but we do not know how to achieve $n_k = \mathsf{poly}$. The result is presented in Sect. 4.

"Multi-distributional" Receiver Deniability from "Delayed Trapdoor Circuits". In our first result, the receiver crucially relies on the assistance of the master authority to generate a new secret key with tailored embedded r_1 and r_2, the trapdoor values. To avoid this, we need to find a way for the central authority to release a fake key that allows the receiver to modify the trapdoor values later on when required. This is solved using the new technique of *delayed trapdoor circuits*. Instead of embedding directly the trapdoor values in the $\mathsf{Trap}[C]$, they are externalised. The trapdoor values are encrypted using an IND-CCA encryption scheme to avoid that the adversary can maul those values and learn something it should not learn. The resulting ciphertext, let us call it Ct', is then linked to the corresponding $\mathsf{Trap}[C]$ by using a one-way function f in this way: a fresh random value z in the domain of f will be encrypted together with the trapdoor values, $t = f(z)$ will be embedded in trapdoor circuit. $\mathsf{Trap}[C]$ then will take in input also Ct' and verify that it encrypts a pre-image of t, before proceeding more. It is easy to see then that the fake key we were looking for is z. Knowing z allows to generate a new Ct' for different trapdoor values. Our construction starts from that of Garg *et al.* [19] but departs from it in many technicalities needed to face the challenges met in the hybrid experiments. Namely, a ciphertext of the functional encryption scheme for x corresponds to a double encryption, à la Naor-Yung [28], of x, using a statistical simulation-soundness NIZK. A secret key for circuit C is the differing-input obfuscation [2,4,13] of a trapdoor circuit $\mathsf{Trap}[C]$ that takes in input the double encryption of x and the double encryption of the trapdoor values related to $\mathsf{Trap}[C]$. Intuitively, differing-input obfuscation is required because there are certain Ct' that allows to understand, for example, which secret key $\mathsf{Trap}[C]$ is using to decrypt the double encryption of x. The actual construction is much more complicated, and is presented in full details in Sect. 3. We point out that in a concurrent work Apon *et al.* [3] construct a bi-deniable FE scheme for the inner-product predicate in the multidistributional model from LWE.

Relation to Simulation-Based Security. As observed by [31], in the case of PKE, deniability implies a scheme is also *non-committing* [15,16] and *secure under key-revealing selective-opening attacks* (SOA-K) [6,18]. On the other hand, it was recently observed by [7] that the notion of simulation-based (SIM) security for FE implicitly incorporates SOA-K. SIM-security is a stronger notion of security for FE than IND-security and has been the subject of multiple recent works [1,5,7,12,17,23,30]. Very roughly, in both notions the adversary makes key-derivation queries, then queries for challenge ciphertexts, then again makes key-derivation queries. SIM-security asks that the "view" of the adversary can be simulated by a simulator given neither ciphertexts nor keys but only the

corresponding outputs of the functionality on the underlying plaintexts, whereas IND-security only asks that it cannot distinguish the encryptions of messages that it cannot trivially distinguish using its requested keys. This leads to the interesting result that a receiver-deniable FE scheme necessarily achieves some form of SIM-security. To formalize it, recall from [17] that (q_1, ℓ, q_2)-SIM security denotes SIM-security where the adversary is allowed to make at most q_1 non-adaptive key queries, ℓ encryption queries (challenge ciphertexts), and q_2 adaptive key queries. We show that an (n_c, n_k)-receiver deniable FE scheme is also $(0, n_c, n_k)$-SIM-secure (see Appendix A for a formal theorem and proof). On the other hand we stress deniability is *stronger* in the respect that equivocable ciphertexts and keys must decrypt correctly in the real system. Our results on receiver deniability can be seen as showing that the techniques of [17] are sufficient not just for achieving SIM-security but for deniability as well. Moreover, this implication implies that known impossibility results for SIM-secure FE [1,7,12,17] mean that in the receiver deniable case (n_c, poly)-deniability (which we achieve assuming IND-secure FE for the circuit functionality) is in fact *optimal*. These impossibility results hold only in the standard model and not in the (programmable) RO model [26], but in the case of deniability it is unclear how programmable ROs could help since programmability only helps in a simulation whereas deniability refers to the behaviour of the real system.

What About Sender Deniability? In this work we choose to focus on receiver deniability rather than sender deniability. Receiver deniability is arguably more important than sender deniability in practice — it is plausible that the sender erases its coins, but not that the receiver erases its key. We believe sender deniability can also be added to our schemes, however, by applying the techniques of Sahai and Waters [32] used to achieve sender-deniable public-key encryption. We investigated sender deniability for FE but we did not include the results in this version.

2 Definitions

We start by giving formal definition of *functional encryption* [12,20], and its security, and *deniable functional encryption* and its security. Due to space constraints we defer to [2,4,13] for definitions of differing-inputs obfuscation, and to Garg *et al.* [19] for the definition of statistical simulation-sound non-interactive zero-knowledge proofs (SSS-NIZK, in short).

Functional Encryption. We define the primitive and its security following Boneh *et al.* [12] notation.

Definition 1. [Functionality] A *functionality* $F = \{F_n\}_{n>0}$ is a family of functions $F_n : K_n \times X_n \to \Sigma$ where K_n is the *key space* for parameter n, X_n is the *message space* for parameter n and Σ is the *output space*. Sometimes we will refer to functionality F as a function from $F : K \times X \to \Sigma$ with $K = \cup_n K_n$ and $X = \cup_n X_n$.

Notice that, when $\boldsymbol{x} = (x_1, \ldots, x_\ell)$ is a vector of messages, for any $k \in K$, we denote by $F(k, \boldsymbol{x})$ the vector of evaluations $(F(k, x_1), \ldots, F(k, x_\ell))$.

Definition 2. [Functional Encryption Scheme] A *functional encryption* scheme for functionality F defined over (K, X) is a tuple $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of 4 algorithms with the following syntax:

1. $\mathsf{Setup}(1^\lambda, 1^n)$ outputs *public* and *master secret* keys $(\mathsf{Mpk}, \mathsf{Msk})$ for *security parameter* λ and *length parameter* n that are polynomially related.
2. $\mathsf{KeyGen}(\mathsf{Msk}, k)$, on input Msk and $k \in K_n$ outputs *secret key* Sk.
3. $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input Mpk and $x \in X_n$ outputs *ciphertext* Ct;
4. $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk})$ outputs $y \in \Sigma \cup \{\perp\}$.

In addition we make the following *correctness* requirement: for all $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, all $k \in K_n$ and $x \in X_n$, for $\mathsf{Sk} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x)$, we have that $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}) = F(k, x)$ whenever $F(k, x) \neq \perp$, except with negligible probability. (See [7] for a discussion about this condition.)

Definition 3. [$\mathsf{Circuit}$ Functionality] The $\mathsf{Circuit}$ functionality has key space K_n equals to the set of all n-input Boolean circuits and message space X_n the set $\{0,1\}^n$ of n-bit strings. For $C \in K_n$ and $x \in X_n$, we have $\mathsf{Circuit}(C, x) = C(x)$, that is, the output of circuit C on input x.

Indistinguishability-Based Security. The indistinguishability-based notion of security for functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for functionality F defined over (K, X) is formalized by means of the following game $\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}$ between an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a *challenger* $\mathcal{C}$.

$\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{C}$ generates $(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and runs $\mathcal{A}_0$ on input Mpk;
2. $\mathcal{A}_0$, during its computation, issues q_1 *non-adaptive key-generation queries*. $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to $\mathcal{A}_0$.
   When $\mathcal{A}_0$ stops, it outputs two *challenge messages vectors*, of length ℓ, $\boldsymbol{x}_0, \boldsymbol{x}_1 \in X^\ell$ and its internal state $\mathtt{st}$.
3. $\mathcal{C}$ picks $b \in \{0,1\}$ at random, and, for $i \in \ell$, computes the *challenge ciphertexts* $\mathsf{Ct}_i = \mathsf{Enc}(\mathsf{Mpk}, x_b[i])$. Then $\mathcal{C}$ sends $(\mathsf{Ct}_i)_{i \in [\ell]}$ to $\mathcal{A}_1$ that resumes its computation from state $\mathtt{st}$.
4. $\mathcal{A}_1$, during its computation, issues q_2 *adaptive key-generation queries*. $\mathcal{C}$ on input key $k \in K$ computes $\mathsf{Sk} = \mathsf{KeyGen}(\mathsf{Msk}, k)$ and sends it to $\mathcal{A}_1$.
5. When $\mathcal{A}_1$ stops, it outputs b'.
6. **Output:** if $b = b'$, for each $i \in [\ell]$, $|x_0^i| = |x_1^i|$, and $F(k, \boldsymbol{x}_0) = F(k, \boldsymbol{x}_1)$ for each k for which $\mathcal{A}$ has issued a key-generation query, then output 1 else output 0.

The advantage of adversary $\mathcal{A}$ in the above game is defined as

$$\mathsf{Adv}^{\mathsf{FE},\mathsf{IND}}_{\mathcal{A}}(1^\lambda) = \mathrm{Prob}[\mathsf{IND}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda) = 1] - 1/2$$

Definition 4. We say that FE is (q_1, ℓ, q_2)-*indistinguishably secure* ((q_1, ℓ, q_2)-IND-Secure, for short) where $q_1 = q_1(\lambda), \ell = \ell(\lambda) q_2 = q_2(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, if all probabilistic polynomial-time adversaries $\mathcal{A}$ issuing at most q_1 non-adaptive key queries, q_2 adaptive key queries and output challenge message vectors of length and most ℓ, have at most negligible advantage in the above game. Notice that, in the case that a parameter is an unbounded polynomial we use the notation poly. If a parameter is not specified then it assumed to be poly.

Receiver-Deniable Functional Encryption Scheme. We define the primitive and its security in the following way:

Definition 5. [Receiver-Deniable Functional Encryption Scheme] A (n_c, n_k)-*receiver-deniable functional encryption scheme for functionality F defined over* (K, X), where $n_c = n_c(\lambda), n_k = n_k(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, is made up of the algorithms $\mathsf{RecDenFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ of a standard FE scheme for F (Definition 2) and in addition the following algorithm:

- $\mathsf{RecFake}(\mathsf{Msk}, k, \mathbf{Ct}, \mathbf{x})$. The *receiver faking algorithm*, on input the master secret key Msk, a key k, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_{n_c})$ and messages $\mathbf{x} = (x_1, \ldots, x_{n_c})$, outputs faked secret key Sk_C.

Correctness is defined as in Definition 2 and indistinguishability as in Definition 4.

Definition 6. [Receiver-Deniability] We require that for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, issuing at most n_k receiver-coerce oracle queries, the following two experiments are computationally indistinguishable.

$\mathsf{RealRecDenExp}_{\mathcal{A}}^{\mathsf{RecDenFE}}(1^\lambda)$	$\mathsf{FakeRecDenExp}_{\mathcal{A}}^{\mathsf{RecDenFE}}(1^\lambda)$
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda);$	$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda);$
$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathcal{O}_1, \mathcal{O}_2}(\mathsf{Mpk});$	$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathcal{O}_1, \mathcal{O}_2}(\mathsf{Mpk});$
$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x_i; r_i))_{i \in [n_c]};$	$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y_i; r_i))_{i \in [n_c]};$
Output: $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_1(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathbf{Ct}^\star, \mathtt{st})$	**Output:** $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_2(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathbf{Ct}^\star, \mathtt{st})$

where $\mathbf{x}^\star = (x_1^\star, \ldots, x_{n_c}^\star)$, $\mathbf{y}^\star = (y_1^\star, \ldots, y_{n_c}^\star)$, and $\mathbf{Ct}^\star = (\mathsf{Ct}_1^\star, \ldots, \mathsf{Ct}_{n_c}^\star)$. $(\mathcal{K}_1, \mathcal{K}_2)$ are the receiver-coerce oracles.

All the oracles declared above are defined as follows:

$\mathcal{K}_1(k, \mathbf{Ct}, \mathbf{x})$	$\mathcal{K}_2(k, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k);$	$\mathsf{Sk}_k \leftarrow \mathsf{RecFake}(\mathsf{Msk}, k, \mathbf{Ct}, \boldsymbol{x});$
Output: Sk_k	**Output:** Sk_k

$\mathcal{O}_1(k, x, y)$	$\mathcal{O}_2(k, x, y)$
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x; r)$;	$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y; r)$;
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$\mathsf{Sk}_k \leftarrow \mathsf{RecFake}(\mathsf{Msk}, k, \mathsf{Ct}, x)$;
Output: $(\mathsf{Ct}, \mathsf{Sk}_k)$	**Output:** $(\mathsf{Ct}, \mathsf{Sk}_k)$

In the above experiments, we require the following:

1. There is no query (k, x, y) issued to $\mathcal{O}_1$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_1$ and there is no query (k, x, y) issued to $\mathcal{O}_2$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_2$, where we consider all queries issued during the entire course of the experiment; i.e., when counting all the queries made by $\mathcal{A}_0$ and $\mathcal{A}_1$ together.
2. For any query issued by $\mathcal{A}_1$ to its oracle $\mathcal{K}_1$ or $\mathcal{K}_2$ oracle for key $k^\star$, neither $\mathcal{A}_0$ nor $\mathcal{A}_1$ query $k^\star$ to either of their oracles $\mathcal{O}_1, \mathcal{O}_2$; i.e., they do not make any query $(k^\star, x, y)$ for any x, y to $\mathcal{O}_1$ or $\mathcal{O}_2$.
3. For each key k different from any of the challenge keys $k_i^\star$ queried by $\mathcal{A}_0$ and $\mathcal{A}_1$ to oracles $\mathcal{O}_1$ or $\mathcal{O}_2$, it holds that $F(k, \mathbf{x}^\star) = F(k, \mathbf{y}^\star)$.

2.1 Multi-distributional Receiver-Deniable Functional Encryption Scheme

Definition 7. [Multi-Distributional Receiver-Deniable FE] A (n_c, n_k)-multi-distributional receiver-deniable functional encryption scheme for functionality F defined over (K, X), where $n_c = n_c(\lambda), n_k = n_k(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, is made up of the algorithms $\mathsf{MDRecDenFE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ of a standard FE scheme for F (Definition 2) and in addition the following two algorithms:

- $\mathsf{DenKeyGen}(\mathsf{Msk}, k)$. The *deniable key generation algorithm*, on input the master secret key Msk, and key k, outputs secret key Sk_k and fake key Fk_k.
- $\mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathbf{Ct}, \mathbf{x})$. The *receiver faking algorithm*, on input secret key and fake key $\mathsf{Sk}_k, \mathsf{Fk}_k$ for key k, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_{n_c})$ and messages $\mathbf{x} = (x_1, \ldots, x_{n_c})$, outputs faked secret key Sk'_k.

Correctness is defined as in Definition 2 and indistinguishability as in Definition 4. We also require the following security property.

Definition 8. [Multi-Distributional Receiver Deniability] We require that for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, issuing at most n_k receiver-coerce oracle queries, the following two experiments are computationally indistinguishable.

$\mathsf{RealMDRecDenExp}_{\mathcal{A}}^{\mathsf{RecDenFE}}(1^\lambda)$	$\mathsf{FakeMDRecDenExp}_{\mathcal{A}}^{\mathsf{RecDenFE}}(1^\lambda)$
$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0(1^\lambda)$;	$(\mathbf{x}^\star, \mathbf{y}^\star, \mathtt{st}) \leftarrow \mathcal{A}_0(1^\lambda)$;
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;	$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x_i; r_i))_{i \in [n_c]}$;	$(\mathsf{Ct}_i^\star \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y_i; r_i))_{i \in [n_c]}$;
Output: $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_1(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathsf{Mpk}, \mathbf{Ct}^\star, \mathtt{st})$	**Output:** $\mathcal{A}_1^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{K}_2(\cdot, \mathbf{Ct}^\star, \mathbf{x}^\star)}(\mathsf{Mpk}, \mathbf{Ct}^\star, \mathtt{st})$

where $\mathbf{x}^\star = (x_1^\star, \ldots, x_{n_c}^\star)$, $\mathbf{y}^\star = (y_1^\star, \ldots, y_{n_c}^\star)$, and $\mathbf{Ct}^\star = (\mathsf{Ct}_1^\star, \ldots, \mathsf{Ct}_{n_c}^\star)$. $(\mathcal{K}_1, \mathcal{K}_2)$ are the receiver-coerce oracles.

All the oracle declared above are defined as follows:

$\mathcal{K}_1(k, \mathbf{Ct}, \mathbf{x})$	$\mathcal{K}_2(k, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$(\mathsf{Sk}_k, \mathsf{Fk}_k) \leftarrow \mathsf{DenKeyGen}(\mathsf{Msk}, k)$;
Output: Sk_k	$\mathsf{Sk}'_k \leftarrow \mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathbf{Ct}, \mathbf{x})$;
	Output: Sk'_k

$\mathcal{O}_1(k, x, y)$	$\mathcal{O}_2(k, x, y)$
$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, x; r)$;	$\mathsf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, y; r)$;
$\mathsf{Sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, k)$;	$(\mathsf{Sk}_k, \mathsf{Fk}_k) \leftarrow \mathsf{DenKeyGen}(\mathsf{Msk}, k)$;
Output: $(\mathsf{Ct}, \mathsf{Sk}_k)$	$\mathsf{Sk}'_k \leftarrow \mathsf{RecFake}(\mathsf{Sk}_k, \mathsf{Fk}_k, \mathsf{Ct}, x)$;
	Output: $(\mathsf{Ct}, \mathsf{Sk}'_k)$

In the above experiments, we require the following:

1. There is no query (k, x, y) issued to $\mathcal{O}_1$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_1$ and there is no query (k, x, y) issued to $\mathcal{O}_2$ and at same time a query $(k, \mathbf{Ct}^\star, \mathbf{x})$ for some $\mathbf{x}$ issued to $\mathcal{K}_2$, where we consider all queries issued during the entire course of the experiment; i.e., when counting all the queries made by $\mathcal{A}_0$ and $\mathcal{A}_1$ together.
2. For any query issued by $\mathcal{A}_1$ to its oracle $\mathcal{K}_1$ or $\mathcal{K}_2$ for key $k^\star$, neither $\mathcal{A}_0$ nor $\mathcal{A}_1$ query $k^\star$ to either of their oracles $\mathcal{O}_1, \mathcal{O}_2$; i.e., they do not make any query $(k^\star, x, y)$ for any x, y to $\mathcal{O}_1$ or $\mathcal{O}_2$.
3. For each key k different from any of the challenge keys $k_i^\star$ queried by $\mathcal{A}$ to oracles $\mathcal{O}_1$ or $\mathcal{O}_2$, it holds that $F(k, \mathbf{x}^\star) = F(k, \mathbf{y}^\star)$.

Remark 9. Our security notion is *selective*, in that the adversary *commits* to (x, y) before it sees Mpk. It is possible to bootstrap selectively-secure scheme to full security using standard complexity leveraging arguments [10,24] at the price of a $2^{|x|}$ loss in the security reduction.

3 Receiver Deniable FE from Trapdoor Circuits

In this section, we present a construction of a (n_c, poly)-*receiver deniable functional encryption scheme for* $\mathsf{Circuit}$, $\mathsf{RecDenFE}$.

Overview. To construct our $\mathsf{RecDenFE}$ scheme, we start from an IND-Secure FE scheme for $\mathsf{Circuit}$. During the key generation, we replace the original circuit with a *trapdoor* one that the *receiver faking algorithm* can then use to program the output in some way. More specifically, we put additional "slots" in the plaintexts and secret keys that will be critically used by the receiver faking algorithm. A plaintext will have two slots where the first slot will be the actual message x. The second slot will be a random string s, some sort of *tag* used to identify the

ciphertext that will serve as seed of a PRF. On the other hand, a secret key for circuit C has $4 \cdot n_c$ slots consisting of random strings t_i, z_i, t'_i, z'_i, for $i \in [n_c]$, used as formal variables to represent Boolean values 0 and 1. Specifically:

Definition 10. [Trapdoor Circuit] Let C be a Boolean circuit on n-bits and 1-bit output, $\mathcal{F} = \{f_s : s \in \{0,1\}^\lambda\}_{\lambda \in \mathbb{N}}$ be a $(l(\lambda), L(\lambda))$-pseudo-random function family. For any $\mathbf{t} = (t_i \in \{0,1\}^{l(\lambda)})_{i \in [n_c]}, \mathbf{z} = (z_i \in \{0,1\}^{L(\lambda)})_{i \in [n_c]}$ and $\mathbf{t}' = (t'_i \in \{0,1\}^{l(\lambda)})_{i \in [n_c]}, \mathbf{z}' = (z'_i \in \{0,1\}^{L(\lambda)})_{i \in [n_c]}$, define the corresponding *trapdoor circuit* $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}$ on $(n+\lambda)$-bit inputs and 1-bit output as follows:

Circuit $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x')$
$(x, s) \leftarrow x'$
If $f_s(t_i) = z_i$ for some $i \in [n_c]$ Then Return 1
Else If $f_s(t'_i) = z'_i$ for some $i \in [n_c]$ Then Return 0
Else Return $C(x)$

We are now ready to present our RecDenFE scheme.

Construction 11. [Receiver Deniable Functional Encryption] Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ be a functional encryption scheme for the functionality Circuit and $\mathcal{F} = \{f_s : s \in \{0,1\}^\lambda\}_{\lambda \in \mathbb{N}}$ be a $(l(\lambda), L(\lambda))$-pseudo-random function family. We define our *receiver deniable functional encryption scheme*$\mathsf{RecDenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{RecFake})$ for Circuit as follows.

- $\mathsf{Setup}(1^\lambda, 1^n)$ runs $\mathsf{FE.Setup}(1^\lambda, 1^{n+\lambda})$ to get the pair $(\mathsf{FE.Mpk}, \mathsf{FE.Msk})$. Then, the master public key is $\mathsf{Mpk} = \mathsf{FE.Mpk}$ and the master secret key is $\mathsf{Msk} = \mathsf{FE.Msk}$. The algorithm returns the pair $(\mathsf{Mpk}, \mathsf{Msk})$.
- $\mathsf{Enc}(\mathsf{Mpk}, x)$ on input master public key $\mathsf{Mpk} = \mathsf{FE.Mpk}$, and message $x \in \{0,1\}^n$, chooses a random $s \in \{0,1\}^\lambda$ and sets $x' = (x, s)$. Then the algorithm computes and returns the ciphertext $\mathsf{Ct} = \mathsf{FE.Enc}(\mathsf{FE.Mpk}, x')$.
- $\mathsf{KeyGen}(\mathsf{Msk}, C)$ on input master secret key $\mathsf{Msk} = \mathsf{FE.Msk}$ and a n-input Boolean circuit C, chooses, for $i \in [n_c]$, random strings $t_i, t'_i \in \{0,1\}^{l(\lambda)}$, $z_i, z'_i \in \{0,1\}^{L(\lambda)}$ and computes $\mathsf{FE.Sk}_C = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$. The algorithm returns the secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$.
- $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}_C)$ on input master public key $\mathsf{Mpk} = \mathsf{FE.Mpk}$, Ct and secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$ for circuit C, returns the output of $\mathsf{FE.Dec}(\mathsf{FE.Mpk}, \mathsf{Ct}, \mathsf{FE.Sk}_C)$.
- $\mathsf{RecFake}(\mathsf{Msk}, C, \mathbf{Ct}, \mathbf{x})$ on input the master secret key $\mathsf{Msk} = \mathsf{FE.Msk}$, a Boolean circuit C on n-bits input and 1-bit output, at most n_c ciphertexts $\mathbf{Ct} = (\mathsf{Ct}_1, \ldots, \mathsf{Ct}_\ell)$ and messages $\mathbf{x} = (x_1, \ldots, x_\ell)$, extracts s_i from each ciphertext Ct_i by using $\mathsf{FE.Msk}$. Then, for each $i \in [\ell]$, RecFake chooses random t_i and t'_i in $\{0,1\}^{l(\lambda)}$ and distinguishes between the following two case:
  - If $C(x_i) = 1$, it sets $z_i = f_{s_i}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$.
  - If $C(x_i) = 0$, it sets $z'_i = f_{s_i}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.

  Finally, RecFake computes $\mathsf{FE.Sk}_C = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$.

Correctness of our RecDenFE scheme follows from the correctness of FE and from the observation that, for randomly chosen $\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}'$ and s and for all x, $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x, s) = C(x)$ except with negligible probability.

Security. The proof of security can be found in Appendix B.

4 Receiver Deniable FE for Boolean Formulae

We have seen in the previous section how to construct a receiver deniable functional encryption scheme for circuits assuming the existence of an IND-Secure FE scheme for the same functionality. To the best of our knowledge, the only way to construct an IND-Secure FE for circuits is by using obfuscation and its variants.

In this section we explore the possibility of achieving receiver deniability for weaker classes of functionalities that still support some form of trapdoor mechanism and for which a functional encryption scheme can be constructed assuming standard assumptions. Namely, we are interested in constructing a receiver deniable FE for Boolean formulae. In [27], Katz *et al*, show how to construct a functional encryption scheme for Boolean formulae given a functional encryption scheme for the inner-product whose security, by the result of Okamoto and Takashima [29], can be based on the Decisional Linear Assumption in bilinear groups. To construct a functional encryption scheme for Boolean formulae, [27] first shows how to construct functional encryption schemes for predicates corresponding to univariate polynomials whose degree d is polynomial in the security parameter. This can be generalized to the case of polynomials in t variables, and degree at most d in each variable, as long as d^t is polynomial in the security parameter. Given the polynomial-based construction, [27] shows that for Boolean variables it is possible to handle arbitrary CNF or DNF formulas by noting that the predicate OR_{I_1,I_2}, where $\mathsf{OR}_{I_1,I_2}(x_1, x_2) = 1$ iff either $x_1 = I_1$ or $x_2 = I_2$, can be encoded as the bivariate polynomial $p(x_1, x_2) = (x_1 - I_1) \cdot (x_2 - I_2)$ and the predicate AND_{I_1,I_2}, where $\mathsf{AND}_{I_1,I_2}(x_1, x_2) = 1$ if both $x_1 = I_1$ and $x_2 = I_2$, correspond to the polynomial $p(x_1, x_2) = (x_1 - I_1) + (x_2 - I_2)$. (Notice that, for non-Boolean variables it is not known how to directly handle negation.) The complexity of the resulting scheme depends polynomially on d^t, where t is the number of variables and d is the maximum degree of the resulting polynomial in each variable. This bound will critically influence our construction of receiver deniable scheme as we will show in the next section. Specifically, the length of the additional slots used in trapdoor mechanism of the previous section will be fixed and independent of the security parameter to avoid the exponential blowup of the complexity of the resulting scheme. As a consequence, the trapdoor mechanism has a non-negligible probability of being active in the real scheme thus influencing the decryption error probability. Parallel repetition will fix this issue.

4.1 Our Construction

Overview. The trapdoor formula will follow the same design lines of the trapdoor circuit we used in the previous section with the main difference being the length

of the slots which will be here constant and independent from the security parameter to avoid the exponential blowup in the [27] construction. Thus, as in the previous section, we will have additional slots in the plaintexts and secret keys that will be critically used by the receiver faking algorithm. The plaintext will have two slots where the first slot will be the actual message x. The second slot will be a random string s. On the other hand, a secret key for Boolean formula f will also have two slots to represent Boolean values 0 and 1. Specifically:

Construction 12. [Trapdoor Boolean Formula] Let f be a Boolean formula on n-bits. For any two strings $r_0, r_1 \in \{0,1\}^\ell$, define the corresponding *trapdoor boolean formula* $\mathsf{Trap}[f]^{r_0,r_1}$ on $(n+\ell)$-bit inputs as follows: $\mathbf{Formula}\mathsf{Trap}[f]^{r_0,r_1}(x,s) := (s = r_1) \vee [f(x) \wedge \neg(s = r_0)]$, where the expression $(s = r)$ is the comparison bit-a-bit.

We are now ready to present our RecDenFE scheme.

Construction 13. [Receiver Deniable Functional Encryption for Boolean Formulae] Let $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Eval})$ be the functional encryption scheme for the functionality Boolean Formulae. For any constant $\ell > 3$, we define our *receiver deniable functional encryption scheme* $\mathsf{RecDenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{RecFake})$ for Boolean formulae as follows.

- $\mathsf{Setup}(1^\lambda, 1^n, 1^m)$, for each $i \in [m]$, runs $\mathsf{FE.Setup}(1^\lambda, 1^{n+\ell})$ to get the pair $(\mathsf{FE.Mpk}_i, \mathsf{FE.Msk}_i)$. Then, the master public key is $\mathsf{Mpk} = (\mathsf{FE.Mpk})_{i\in[m]}$ and the master secret key is $\mathsf{Msk} = (\mathsf{FE.Msk})_{i\in[m]}$. The algorithm returns the pair $(\mathsf{Mpk}, \mathsf{Msk})$.
- $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input master public key $\mathsf{Mpk} = (\mathsf{FE.Mpk}_i)_{i\in[m]}$ and message $x \in \{0,1\}^n$, for each $i \in [m]$, chooses a random $s_i \in \{0,1\}^\ell$ and sets $\mathsf{Ct}_i = \mathsf{FE.Enc}(\mathsf{FE.Mpk}_i, (x, s_i))$. The algorithm returns the ciphertext $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$.
- $\mathsf{KeyGen}(\mathsf{Msk}, f)$, on input master secret key $\mathsf{Msk} = (\mathsf{FE.Msk})_{i\in[m]}$ and a n-input Boolean formula f, for each $i \in [m]$, chooses two random strings $r_0^i, r_1^i \in \{0,1\}^\ell$, such that $r_0^i \neq r_1^i$, and computes secret key $\mathsf{FE.Sk}_f^i = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}_i, \mathsf{Trap}[f]^{r_0^i, r_1^i})$. The algorithm returns the secret key $\mathsf{Sk}_f = (r_0^i, r_1^i, \mathsf{FE.Sk}_f^i)_{i\in[m]}$.
- $\mathsf{Dec}(\mathsf{Mpk}, \mathsf{Ct}, \mathsf{Sk}_f)$, on input master public key $\mathsf{Mpk} = (\mathsf{FE.Mpk})_{i\in[m]}$, $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$ and secret key $\mathsf{Sk}_f = (r_0^i, r_1^i, \mathsf{FE.Sk}_f^i)_{i\in[m]}$ for Boolean formula f, for $i \in [m]$, computes Boolean value $b_i = \mathsf{FE.Eval}(\mathsf{FE.Mpk}_i, \mathsf{Ct}_i, \mathsf{FE.Sk}_f^i)$, and returns as output the Boolean value on which the majority of b_i's have agreed on.
- $\mathsf{RecFake}(\mathsf{Msk}, f, \mathsf{Ct}, x')$, on input the master secret key $\mathsf{Msk} = (\mathsf{FE.Msk}_i)_{i\in[m]}$, an n-input Boolean formula f, ciphertext $\mathsf{Ct} = (\mathsf{Ct}_i)_{i\in[m]}$ and message x', for all $i \in [m]$, the algorithm extracts s_i from Ct_i by using $\mathsf{FE.Msk}_i$. Now, RecFake chooses the r_0^i's and r_1^i's by following a binomial distribution with number of trials equals to m and success probability $p = (1-2^{-\ell})$. Specifically, RecFake distinguishes between the following two cases. Let $b' = f(x')$, for each $i \in [m]$, if there is a success in the i-th trial then RecFake sets $r_{b'}^i = s_i$ and

r^i_{1-b} to a random value different from s_i. Otherwise, $\mathsf{RecFake}$ sets $r^i_{1-b'} = s_i$ and r_b to a random value different from s_i. Finally, $\mathsf{RecFake}$ computes secret key $\mathsf{FE.Sk}^i_f = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}_i, \mathsf{Trap}[f]^{r^i_0, r^i_1})$, and returns the secret key $\mathsf{Sk}_f = (r^i_0, r^i_1, \mathsf{FE.Sk}^i_f)_{i\in[m]}$ as faking key.

Correctness. Notice that for any $i \in [m]$, the probability that $s_i = r^i_0 \vee s_i = r^i_1$ is at most $2^{-\ell+1}$. Thus the output of the decryption is correct, i.e. $\mathsf{Trap}[f]^{r^i_0, r^i_1}(x, s_i) = f(x)$, with probability at least $1 - 2^{-\ell+1}$.

Thus on average, an $(1 - 2^{-\ell+1})$ fraction of the ciphertexts will be decrypted to the correct value and for large enough m, the Chernoff bound guarantees that the correctness of $\mathsf{RecDenFE}$ hold with overwhelming probability.

Security. The proof that $\mathsf{RecDenFE}$ is a $(1, 1)$-receiver deniable functional encryption scheme for Boolean formulae is essentially that of Theorem 20 and we omit further details.

We note that one can extend the scheme to (n_c, n_k)-receiver deniability in a simple way but we cannot achieve $n_k = \mathsf{poly}$ in this case, however, because we cannot use symmetric encryption (at least in a straightforward way).

5 Multi-distributional Receiver Deniable FE from di$\mathcal{O}$

In order to avoid to overburden the notation and to make the presentation easy to follow, we present a construction of a $(1, 1)$-*multi-distributional receiver deniable functional encryption scheme for* $\mathsf{Circuit}$.

Overview. Our construction resembles that of Garg *et al.* [19]. Namely, a ciphertext of the functional encryption scheme for x corresponds to a double encryption, à la Naor-Yung [28], of x, using a statistical simulation-soundness NIZK. A secret key for circuit C is the differing-input obfuscation of a trapdoor circuit $\mathsf{Trap}[C]$ that takes in input the double encryption of x and the double encryption of the trapdoor values.

Construction 14. [Multi-Distributional Receiver Deniable FE] Given an IND-CPA PKE system $\mathcal{E} = (\mathcal{E}.\mathsf{Setup}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ with perfect correctness, a differing-inputs obfuscator $\mathsf{di}\mathcal{O}$, an SSS-NIZK proof system $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify}, \mathsf{NIZK.Sim})$ and a one-way function f, we define our multi-distributional receiver deniable functional encryption $\mathsf{MDRecDenFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{DenKeyGen}, \mathsf{RecFake}, \mathsf{Dec})$ as follows:

1. $\mathsf{Setup}(1^\lambda)$ takes in input the *security parameter* λ and computes the following: For $i \in [4]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\lambda)$. Then, $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. The algorithm sets $\mathsf{Mpk} = ((\mathsf{pk}_i)_{i\in[4]}, f, \mathsf{crs})$, $\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$.
2. $\mathsf{KeyGen}(\mathsf{Msk}, C)$ takes in input *master secret key* $\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$, and *circuit* C, and does the following: Computes common reference string $\mathsf{crs}' \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Then, sample random z in the domain of f and set $t = f(z)$ and compute $\mathsf{Ct}' := (\mathsf{ct}_3, \mathsf{ct}_4, \pi_2)$, where $\mathsf{ct}_3 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, 0^n, 0^\lambda); r_3)$ and

$\mathsf{ct}_4 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, 0^n, 0^\lambda); r_4)$, and π_2 is a NIZK proof of Eq. 2. Finally, the algorithm computes a differing-input obfuscation $\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}$ for the trapdoor circuit $\mathsf{Trap}_{1,3}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_i, \mathsf{sk}_j, f, t]$. The algorithm outputs *secret key* for the circuit C, $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$.

3. $\mathsf{DenKeyGen}(\mathsf{Msk}, C)$ takes in input$\mathsf{Msk} = ((\mathsf{sk}_i)_{i\in[4]})$ and *circuit* C, and computes $\mathsf{Sk}_C = \mathsf{KeyGen}(\mathsf{Msk}, C)$. The algorithm outputs Sk_C as the *secret key* for the circuit C and $\mathsf{Fk}_C = z$.
4. $\mathsf{Enc}(\mathsf{Mpk}, x)$, on input *master public key* $\mathsf{Mpk} = ((\mathsf{pk}_i)_{i\in[4]}, f, \mathsf{crs})$ and *messages* $x \in X_n$, computes $\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1)$, where $\mathsf{ct}_1 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1)$ and $\mathsf{ct}_2 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2)$ and π_1 is a NIZK proof of Eq. 1. The algorithm outputs *ciphertext* Ct.
5. $\mathsf{Dec}(\mathsf{Sk}_C, \mathsf{Ct})$ on input *secret key* $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$ and *ciphertext* Ct, the algorithm outputs $\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}(\mathsf{Ct}, \mathsf{Ct}')$.
6. $\mathsf{RecFake}(\mathsf{Sk}_C, \mathsf{Fk}_C, \mathsf{Ct}, x)$ on input *secret key* $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \mathsf{Ct}')$, *fake key* $\mathsf{Fk}_C = z$, where $t = f(z)$, *ciphertext* $\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1)$ and *message* x, does the following: Compute $\hat{\mathsf{Ct}} := (\hat{\mathsf{ct}}_3, \hat{\mathsf{ct}}_4, \hat{\pi}_2)$, where $\hat{\mathsf{ct}}_3 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, \mathsf{Ct}, x); r_3)$ and $\hat{\mathsf{ct}}_4 \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, \mathsf{Ct}, x); r_4)$ and $\hat{\pi}_2$ is a NIZK proof of Eq. 2. The new secret key for circuit C is $\mathsf{Sk}_C = (\mathsf{di}\mathcal{O}_{\mathsf{Trap}_{1,3}}, t, \hat{\mathsf{Ct}})$.

Correctness follows immediately from the correctness of the $\mathsf{di}\mathcal{O}$, PKE, SSS-NIZK, and the description of the trapdoor circuits described below.

$\mathsf{Trap}_{i,j}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_i, \mathsf{sk}_j, f, t](\mathsf{Ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \pi_1), \mathsf{Ct}' = (\mathsf{ct}_3, \mathsf{ct}_4, \pi_2))$

The algorithm does the following:

1. Check that π_1 is valid NIZK proof (using the $\mathsf{NIZK.Verify}$ algorithm and crs) for the NP-statement

$$\begin{aligned} &\exists x, r_1, r_2 : \\ &\mathsf{ct}_1 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1) \text{ and } \mathsf{ct}_2 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2) \end{aligned} \tag{1}$$

2. Check that π_2 is valid NIZK proof (using the $\mathsf{NIZK.Verify}$ algorithm and crs') for the NP-statement

$$\begin{aligned} &\exists z, c, x, r_3, r_4 : \\ &\mathsf{ct}_3 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, c, x); r_3) \text{ and } \mathsf{ct}_4 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, c, x); r_4) \text{ and } f(z) = t \end{aligned} \tag{2}$$

3. If any checks fail output 0.
4. $(z', c', x') \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j)$
5. if $c' = \mathsf{Ct}$ then output $C(x')$; otherwise output $C(\mathcal{E}.\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct}_i))$.

Remark 15. To allow a secret key to support faking against n_c ciphertexts, like we do in Sect. 3, we attach to a secret key n_c ciphertexts Ct'_i each being the double encryption of $(z_i, 0^n, 0^\lambda)$ for different z_i's. Then, $\mathsf{Trap}_{i,j}$ will contain the images under f of all z_i's.

Proof of Security. We prove the following main theorem.

Theorem 16. If $\mathsf{di}\mathcal{O}$ is an differing-input obfuscator, $\mathcal{E}$ is IND-CPA and f is a one-way function then $\mathsf{MDRecDenFE}$ is a $(1,1)$-*multi-distributional receiver deniable* in the sense of Definition 6.

To prove the above theorem, we prove the indistinguishability of the following hybrid experiments. Recall that, for simplicity, we prove $(1,1)$-security. Extending the proof of security to n_c being any constant and $n_k = \mathsf{poly}$ resorts to add more hybrids to switch the challenge ciphertexts and the secret keys generated by the receiver-coerce oracle to the target distribution.

Hybrid H_1. This is the $\mathsf{RealMDRecDenExp}$ experiment where the receiver-coerce oracle is $\mathcal{K}_1$.

Hybrid H_2. It is identical to H_1 except that: (1) $(\mathsf{crs}, \pi_1^\star)$ is simulated as $(\mathsf{crs}, \pi_1^\star) \leftarrow \mathsf{NIZK.Sim}(1^\lambda, \exists x, r_1, r_2 : \mathsf{ct}_1^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1)$ and $\mathsf{ct}_2^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2))$ where $\mathsf{ct}_1^\star, \mathsf{ct}_2^\star$ is part of the challenge ciphertext. Note that, in the selective security game the challenge ciphertext can be given out simultaneously with the public parameters. (2) (crs', π_2), generated by the receiver-coerce oracle, is simulated as $(\mathsf{crs}', \pi_2) \leftarrow \mathsf{NIZK.Sim}(1^\lambda, \exists z, c, x, r_3, r_4 : \mathsf{ct}_3 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z, c, x); r_3) \wedge \mathsf{ct}_4 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z, c, x); r_4) \wedge f(z) = t)$. Notice that the crs of the secret keys generated by $\mathcal{O}_1$ and $\mathcal{O}_2$ are not simulated. The indistinguishability of H_2 from H_1 follows from the ZK property of the NIZK system and by a standard hybrid argument.

Hybrid H_3. It is identical to H_2 except that $\mathsf{ct}_2^\star$ encrypts y. The NIZK's are still simulated. The indistinguishability of H_3 from H_2 follows from the IND-CPA security of $(\mathsf{pk}_2, \mathsf{sk}_2)$.

Hybrid H_4. It is identical to H_3 except that ct_4, generated by the receiver-coerce oracle, encrypts $(0^k, \mathsf{Ct}^\star, x)$. The NIZK's are still simulated. The indistinguishability of H_4 from H_3 follows from the IND-CPA security of $(\mathsf{pk}_4, \mathsf{sk}_4)$.

Hybrid H_5. It is identical to H_4 except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{1,4}$. The indistinguishability of H_5 from H_4 follows from the security of $\mathsf{di}\mathcal{O}$ noticing that $\mathsf{Trap}_{1,3}$ and $\mathsf{Trap}_{1,4}$ compute the same function. This follows: (1) by the statistical simulation-soundness of the NIZK system that guarantees that Ct', as generated by the receiver-coerce oracle, used in both experiments, is the only one to contain a NIZK proof for a false statement accepted by the verifier and (2) by the fact that by definition of $\mathsf{Trap}_{1,3}$, $\mathsf{Trap}_{1,4}$ and Ct', it holds that: $\mathsf{Trap}_{1,3}(\mathsf{Ct}^\star, \mathsf{Ct}') = C(\mathcal{E}.\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ct}_1)) = C(x) = C(\mathcal{E}.\mathsf{Dec}(\mathsf{sk}_4, \mathsf{ct}_4)) = \mathsf{Trap}_{1,4}(\mathsf{Ct}^\star, \mathsf{Ct}')$.

Hybrid H_6. It is identical to H_5 except that the secret keys, generated by $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{1,4}$. The indistinguishability of H_6 from H_5 follows from the security of $\mathsf{di}\mathcal{O}$ noticing that $\mathsf{Trap}_{1,3}$ and $\mathsf{Trap}_{1,4}$ compute the same function. The statistical simulation-soundness of the NIZK system guarantees that there is no Ct' for a false statement that the verifier accepts.

Hybrid H_7. It is identical to H_6 except that ct_3, generated by the receiver-coerce oracle, encrypts $(0^k, \mathsf{Ct}^\star, x)$. The NIZK's are still simulated. The indistinguishability of H_7 from H_6 follows from the IND-CPA security of $(\mathsf{pk}_3, \mathsf{sk}_3)$.

Hybrid H_8. It is identical to H_7 except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{1,3}$. The indistinguishability of H_8 from H_7 is symmetrical to that of H_5 from H_4.

Hybrid H_9. It is identical to H_8 except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,3}$.

Overview: by the statistical simulation-soundness of the NIZK proof system and by definition of $\mathsf{Trap}_{2,3}$, the inputs that distinguish $\mathsf{Trap}_{2,3}$ from $\mathsf{Trap}_{1,3}$ have the form $(\mathsf{Ct}^\star, \hat{\mathsf{Ct}})$ where: (1) $\hat{\mathsf{Ct}} = (\hat{\mathsf{ct}}_3, \hat{\mathsf{ct}}_4, \hat{\pi}_2) \neq \mathsf{Ct}'$. (2) $\hat{\mathsf{ct}}_3$ and $\hat{\mathsf{ct}}_4$ encrypt the same value (this follows by the statistical simulation-soundness of the NIZK system and from the fact that, being Ct' the only false statement with accepting proof, it must hold that $\hat{\mathsf{Ct}} \neq \mathsf{Ct}'$), and (3) $\hat{\mathsf{ct}}_3$ encrypts a string of the form (z', c', x') with $f(z') = t$.

To prove the indistinguishability of H_8 from H_7, we proceed in two steps: Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be any multi-distributional receiver deniability adversary, in the first step we show that there exists a sampling algorithm $\mathsf{Sampler}^{\mathcal{A}}$ that samples a circuit family $\mathcal{C}$ (containing $\mathsf{Trap}_{2,3}$ and $\mathsf{Trap}_{1,3}$), that it is differing-inputs under the one-wayness of f. In the second step, we show that if $\mathcal{A}$ can distinguish the two experiments, then it is possible to construct a distinguisher that breaks the security of $\mathsf{di}\mathcal{O}$.

First Step: We define a circuit family $\mathcal{C}$ associated with a PPT $\mathsf{Sampler}^{\mathcal{A}}$ and show that it is differing-inputs under the one-wayness of f. $\mathsf{Sampler}^{\mathcal{A}}$ takes in input the security parameter, the description of the OWF f and the challenge of the one-way security game $t^\star$, and does the following:

1. Runs $\mathcal{A}_0$ on input 1^λ to obtain $(x^\star, y^\star, \mathtt{st}_{\mathcal{A}})$.
2. Computes, for $i \in [4]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\lambda)$, and $\mathsf{ct}_1^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x^\star)$ and $\mathsf{ct}_2^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, y^\star)$ and $(\mathsf{crs}, \pi_1^\star) \leftarrow \mathsf{NIZK.Sim}(1^\lambda, \exists x, r_1, r_2 : \mathsf{ct}_1^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_1, x; r_1)$ and $\mathsf{ct}_2^\star = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_2, x; r_2))$. Sets the master public key and the challenge ciphertext as $\mathsf{Mpk} = ((\mathsf{pk}_i)_{i\in[4]}, f, \mathsf{crs})$, $\mathsf{Ct}^\star = (\mathsf{ct}_1^\star, \mathsf{ct}_2^\star, \pi_1^\star)$. The master secret key is then $\mathsf{Msk} = (\mathsf{sk}_i)$.

   Finally, runs $\mathcal{A}_1$ on input $(\mathsf{Mpk}, \mathsf{Ct}^\star, \mathtt{st}_{\mathcal{A}})$, simulating oracles $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{K}(\cdot, \mathsf{Ct}^\star, x^\star)$ in the following way:

1. $\mathcal{O}_1(k,x,y), \mathcal{O}_2(k,x,y)$: Given Mpk and Msk, the output distributions of these oracles is easy to generate.
2. $\mathcal{K}(C, \mathsf{Ct}^\star, x^\star)$: $\mathsf{Sampler}^{\mathcal{A}}$ computes $\mathsf{ct}_3 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (0^\lambda, \mathsf{Ct}^\star, x^\star))$ and $\mathsf{ct}_4 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (0^\lambda, \mathsf{Ct}^\star, x^\star))$, and $(\mathsf{crs}', \pi_2) \leftarrow \mathsf{NIZK.Sim}(\exists z, c, x, r_3, r_4 : \mathsf{ct}_3 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_3, (z,c,x); r_3)$ and $\mathsf{ct}_4 = \mathcal{E}.\mathsf{Enc}(\mathsf{pk}_4, (z,c,x); r_4)$ and $f(z) = t^\star)$. At this point the algorithm interrupts the execution of $\mathcal{A}$ and returns $(\mathsf{Trap}_{1,3}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_1, \mathsf{sk}_3, f, t], \mathsf{Trap}_{2,3}[C, \mathsf{crs}, \mathsf{crs}', \mathsf{sk}_2, \mathsf{sk}_3, f, t], \mathtt{st})$, where $\mathtt{st}$ contains its entire computation.

This terminates the description of $\mathsf{Sampler}^{\mathcal{A}}$. Now, suppose there exists an adversary $\mathcal{B}$ that takes in input $(1^\lambda, \mathsf{Trap}_{1,3}, \mathsf{Trap}_{2,3}, \mathtt{st})$ and finds an input on which $\mathsf{Trap}_{1,3}, \mathsf{Trap}_{2,3}$ are different, meaning $\mathcal{B}$ finds a $\hat{\mathsf{Ct}} = (\hat{\mathsf{ct}}_3, \hat{\mathsf{ct}}_4, \hat{\pi}_2)$ as defined above. Then, by using sk_3 in $\mathtt{st}$, $\mathcal{B}$ can decrypt $\hat{\mathsf{ct}}_3$ and extract a pre-image of $t^\star$.

Second Step: Suppose that $\mathcal{A}$ distinguishes H_9 and H_8 with non-negligible advantage then we can construct a distinguisher $\mathcal{D}^{\mathcal{A}}$, for the differing-input circuit family defined above, that breaks the security of $\mathsf{di}\mathcal{O}$. The distinguisher $\mathcal{D}^{\mathcal{A}}$ taks in $(C, \mathtt{st})$ where C is the differing-input obfuscation of either $\mathsf{Trap}_{1,3}$ or $\mathsf{Trap}_{2,3}$ and does the following: (1) Restart from the position where $\mathsf{Sampler}^{\mathcal{A}}$ stopped and uses C to generate the output of the receiver-coerce oracle. Specifically, $\mathcal{D}$ returns $(C, t^\star, \mathsf{Ct}')$, where $\mathsf{Ct}' = (\mathsf{ct}_3, \mathsf{ct}_4, \pi_2)$. (2) $\mathcal{D}$ continues to respond to the oracle invocations as $\mathsf{Sampler}^{\mathcal{A}}$ does. (3) Finally, when $\mathcal{A}$ has completed its execution, it returns a bit that become $\mathcal{D}$'s output.

This terminates the description of $\mathcal{D}^{\mathcal{A}}$. Now, if C is the differing-input obfuscation of $\mathsf{Trap}_{1,3}$ then $(\mathsf{Sampler}^{\mathcal{A}}, \mathcal{D}^{\mathcal{A}})$ have simulated H_8. On the other hand, if C is the differing-input obfuscation of $\mathsf{Trap}_{2,3}$ then $(\mathsf{Sampler}^{\mathcal{A}}, \mathcal{D}^{\mathcal{A}})$ have simulated H_9.

Hybrid H_{10}. It is identical to H_9 except that the secret keys, generated by the $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,4}$. The indistinguishability of H_{10} from H_9 follows from the security of $\mathsf{di}\mathcal{O}$ noticing that $\mathsf{Trap}_{1,4}$ and $\mathsf{Trap}_{2,4}$ compute the same function. The statistical simulation-soundness of the NIZK system guarantees that there is no Ct for a false statement that the verifier accepts. Moreover when considering $\mathsf{Ct}^\star$, the security game constraints guarantee that the secret keys asked to $\mathcal{O}_1$ and $\mathcal{O}_2$ evaluate to the same value on the challenge messages.

Hybrid H_{11}. It is identical to H_{10} except that $\mathsf{ct}_1^\star$ encrypts y, The indistinguishability of H_{11} from H_{10} follows from the IND-CPA security of $(\mathsf{pk}_1, \mathsf{sk}_1)$.

Hybrid H_{12}. It is identical to H_{11} except that the secret keys, generated by the $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,3}$. The indistinguishability of H_{12} from H_{11} is symmetrical to that of H_6 from H_5.

Hybrid H_{13}. It is identical to H_{12} except that ct_4, generated by the receiver-coerce oracle, encrypts $(z, \mathsf{Ct}^\star, x)$, The indistinguishability of H_{13} from H_{12} follows from the IND-CPA security of $(\mathsf{pk}_4, \mathsf{sk}_4)$.

Hybrid H_{14}. It is identical to H_{13} except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,4}$. The indistinguishability of H_{14} from H_{13} is symmetrical to that of H_5 from H_4.

Hybrid H_{15}. It is identical to H_{14} except that the secret keys, generated by the $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,4}$. The indistinguishability of H_{15} from H_{14} is symmetrical to that of H_6 from H_5.

Hybrid H_{16}. It is identical to H_{15} except that ct_3, generated by the receiver-coerce oracle, encrypts $(z, \mathsf{Ct}^\star, x)$. The indistinguishability of H_{16} from H_{15} follows from the IND-CPA security of $(\mathsf{pk}_3, \mathsf{sk}_3)$.

Hybrid H_{17}. It is identical to H_{16} except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,3}$. The indistinguishability of H_{17} from H_{16} is symmetrical to that of H_5 from H_4.

Hybrid H_{18}. It is identical to H_{17} except that the secret keys, generated by the receiver-coerce oracle, contain the differing-input obfuscation of the program $\mathsf{Trap}_{1,3}$. The indistinguishability of H_{18} from H_{17} is symmetrical to that of H_5 from H_4.

Hybrid H_{19}. It is identical to H_{18} except that the secret keys, generated by the $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{2,3}$. The indistinguishability of H_{19} from H_{18} is symmetrical to that of H_6 from H_5.

Hybrid H_{20}. It is identical to H_{19} except that the secret keys, generated by the $\mathcal{O}_1$ and $\mathcal{O}_2$, contain the differing-input obfuscation of the program $\mathsf{Trap}_{1,3}$. The indistinguishability of H_{20} from H_{19} is symmetrical to that of H_{10} from H_9.

Hybrid H_{21}. It is identical to H_{20} except that all crs's are honestly generated. The indistinguishability of H_{21} from H_{20} follows from the zero-knowledge of the NIZK system and by a standard hybrid argument. It remains to notice that H_{21} corresponds to $\mathsf{FakeDenExp}$ where the receiver-coerce oracle is $\mathcal{K}_2$.

6 Open Problems and Future Work

Our work leaves open the problem of a construction of a multidistributional deniable FE for general functionalities that avoid the use of diO. It is also worthy to investigate whether our techniques can be used to add deniability to other flavors of FE, e.g., [9,11,25,33].

Acknowledgments. Vincenzo Iovino is supported by the National Research Fund of Luxembourg and made part of this work while was at the University of Warsaw

supported by the WELCOME/2010-4/2 grant founded within the framework of the EU Innovative Economy Operational Programme.

Angelo de Caro's work was partly supported by the EU H2020 TREDISEC project, funded by the European Commission under grant agreement no. 644412.

We thank Anna Sorrentino and the anonymous reviewers for useful comments.

A Simulation-Based Security for FE and Its Relation to Receiver-Deniability

Definition 17. [De Caro *et al.* [17] Simulation-Based Definition] A FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for functionality F defined over (K, X) is (q_1, ℓ, q_2)-*simulation-secure* ((q_1, ℓ, q_2)-SIM-Secure, for short), where $q_1 = q_1(\lambda), \ell = \ell(\lambda), q_2 = q_2(\lambda)$ are polynomials in the security parameter λ that are fixed a priori, if there exists a PPT *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for all PPT *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, issuing at most q_1 non-adaptive key queries, q_2 adaptive key queries and output challenge message vector of length and most ℓ, the outputs of the following two experiments are computationally indistinguishable.

$\mathsf{RealExp}^{\mathsf{FE},\mathcal{A}}(1^\lambda, 1^n)$	$\mathsf{IdealExp}^{\mathsf{FE},\mathcal{A}}_{\mathsf{Sim}}(1^\lambda, 1^n)$
$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n);$	$(\mathsf{Mpk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n);$
$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk});$	$(\boldsymbol{x}, \mathtt{st}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk});$
$\mathbf{Ct} \leftarrow \mathsf{Enc}(\mathsf{Mpk}, \boldsymbol{x});$	$(\mathbf{Ct}, \mathtt{st}') \leftarrow \mathsf{Sim}_0(\mathsf{Mpk}, \lvert\boldsymbol{x}\rvert, (k_i, \mathsf{Sk}_{k_i}, F(k_i, \boldsymbol{x})));$
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{Msk},\cdot)}(\mathsf{Mpk}, \mathbf{Ct}, \mathtt{st});$	$\alpha \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot)}(\mathsf{Mpk}, \mathsf{Ct}, \mathtt{st});$
Output: $(\mathsf{Mpk}, \boldsymbol{x}, \alpha)$	**Output:** $(\mathsf{Mpk}, \boldsymbol{x}, \alpha)$

Here, the (k_i)'s correspond to the key-generation queries of the adversary. Further, oracle $\mathcal{O}(\cdot)$ is the second stage of the simulator, namely algorithm $\mathsf{Sim}_1(\mathsf{Msk}, \mathtt{st}', \cdot, \cdot)$. Algorithm Sim_1 receives as third argument a key k_j for which the adversary queries a secret key, and as fourth argument the output value $F(k_j, \boldsymbol{x})$. Further, note that the simulator algorithm Sim_1 is stateful in that after each invocation, it updates the state $\mathtt{st}'$ which is carried over to its next invocation. (Notice that, in the case that a parameter is an unbounded polynomial we use the notation poly.)

(n_c, n_k)-receiver-deniability $\implies$ $(0, n_c, n_k)$-SIM-Security.

Theorem 18. Suppose that RecDenFE is a (n_c, n_k)-receiver-deniable functional encryption scheme for functionality F defined over (K, X) then RecDenFE is $(0, n_c, n_k)$-SIM-Secure (Definition 17) as well.

Proof. We start by constructing the simulator required by the SIM-Security. $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ is defined as follow. Sim_0 on input master public key Mpk, remember that non-adaptive key generation queries are not allowed in the setting we are considering, chooses random vector $\boldsymbol{x}^\star$ of n_c messages and, for $i \in n_c$,

generated ciphertext $\mathsf{Ct}_i^\star = \mathsf{Enc}(\mathsf{Mpk}, x_i^\star)$ and returns $(\mathbf{Ct}^\star, \mathtt{st} = (\mathbf{Ct}^\star))$. Sim_1 on input master secret key Msk, status $\mathtt{st}'$, and tuple $(k, \mathsf{Sk}_C, F(k, \boldsymbol{x})))$ does the following. Sim_1 invokes the receiver faking algorithm to generate the secret key for k, namely $\mathsf{Sk}_k = \mathsf{RecFake}(\mathsf{Msk}, k, \mathbf{Ct}^\star, F(k, \boldsymbol{x}))$. Finally, Sim_1 returns Sk_k as its own output.

Now, for the sake of contradiction, let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and $\mathcal{D}$ be an adversary and a distinguisher that break the $(0, n_c, n_k)$-SIM-Security of $\mathsf{RecDenFE}$, meaning that $(\mathcal{A}, \mathcal{D})$ can distinguish between $\mathsf{RealExp}$ and $\mathsf{IdealExp}$. Then, we construct and adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ and distinguisher $\mathcal{D}'$ that break the (n_c, n_k)-receiver-deniable security of $\mathsf{RecDenFE}$. Specifically, $\mathcal{B}$ is defined as follows: $\mathcal{B}_0$ on input master public key Mpk runs $\mathcal{A}_0$ on input Mpk. Notice that, $\mathcal{A}_0$ does not issue any key generation query. At some point $\mathcal{A}_0$ returns challenge messages $\boldsymbol{x}$ and status $\mathtt{st}_{\mathcal{A}}$. $\mathcal{B}_0$ chooses random messages $\boldsymbol{x}'$ and returns $(\boldsymbol{x}, \boldsymbol{x}', \mathtt{st} = (\mathtt{st}_{\mathcal{A}}, \boldsymbol{x}))$.

$\mathcal{B}_1$ on input $\mathbf{Ct}^\star$ and status $\mathtt{st}$ (notice that in this case $\boldsymbol{r}_S$ is a zero-length vector) runs $\mathcal{A}_1$ on input $\mathbf{Ct}^\star$ and status $\mathtt{st}_{\mathcal{A}}$. When $\mathcal{A}_1$ issue a key-generation query for key k, $\mathcal{B}$ invokes its $\mathcal{O}^K$ oracle on input k, $\mathbf{Ct}^\star$ and $F(k, \boldsymbol{x})$ to obtain Sk_k that is given back to $\mathcal{A}_1$. At some point A_1 returns some output α and $\mathcal{B}_1$ returns α as its own output.

On the other hand, the distinguisher $\mathcal{D}'$ is exactly $\mathcal{D}$.

Now notice that, if $\mathcal{B}$ is playing the $\mathsf{RealRecDenExp}$ experiment then $\mathcal{A}$ is playing the $\mathsf{RealExp}$. On the other side, if $\mathcal{B}$ is playing the $\mathsf{FakeRecDenExp}$ experiment then $\mathcal{A}$ is playing the $\mathsf{IdealExp}$. This concludes the proof.

B Proof of Security of Construction 11

In this section, we prove the following main theorems

Theorem 19. If FE is IND-Secure, then $\mathsf{RecDenFE}$ is IND-Secure as well.

The proof of Theorem 19 is straightforward and we omit it.

Theorem 20. If FE is $(\mathsf{poly}, 1, \mathsf{poly})$-$\mathsf{IND}$-Secure then $\mathsf{RecDenFE}$ is a (n_c, poly)-*receiver deniable* in the sense of Definition 6, for any constant n_c.

Proof. We prove security via a sequence of hybrid experiments. To do so, we will make use of the following simulation receiver faking algorithm.

$\mathsf{Sim.RecFake}^{\mathsf{FE.KeyGen}(\mathsf{FE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s})$

The algorithm takes in input a circuit C, messages $\mathbf{x} = (x_1, \ldots, x_\ell)$, strings $\mathbf{s} = (s_1, \ldots, s_\ell)$ each in $\{0,1\}^\lambda$, and oracle access to the FE key generation algorithm. Then, for each $i \in [\ell]$, the algorithm chooses random t_i and t_i' in $\{0,1\}^{l(\lambda)}$ and distinguishes between the following two case:

- If $C(x_i) = 1$, it sets $z_i = f_{s_i}(t_i)$ and chooses random $z_i' \in \{0,1\}^{L(\lambda)}$.
- If $C(x_i) = 0$, it sets $z_i' = f_{s_i}(t_i')$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.

Finally, the algorithm computes $\mathsf{FE.Sk}_C = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$.

We are now ready to describe the hybrids. The change between the presented hybrid and the previous will be denoted by boxing the modified parts.

Hybrid H_1: Consider the following two oracles:

$\mathcal{E}_1^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(\mathsf{Ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{Mpk}, (x_i, s_i)))_{i\in[n_c]}$
Output: $((\mathsf{Ct}_i), \emptyset)$

$\mathcal{K}_1^\star(C, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_C \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, C)$;
Output: Sk_k

Then, hybrid H_1 is the real experiment $\mathsf{RealDenExp}$ where the challenge ciphertexts are created by oracle $\mathcal{E}_1^\star$, and the receiver-coerce oracle is $\mathcal{K}_1^\star$.

Notice that $\mathcal{E}_1^\star$ is exactly $\mathcal{E}_1$ with the only difference that we have unrolled the call to the $\mathsf{RecDenFE}$ encryption algorithm for the sake of clarity, and $\mathcal{K}_1^\star = \mathcal{K}_1$.

Hybrid H_2: Consider the following oracles:

$\mathcal{E}_2^\star(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$\boxed{(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}}$
$(\mathsf{Ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{Mpk}, (x_i, s_i)))_{i\in[n_c]}$
Output: $((\mathsf{Ct}_i), \emptyset)$

$\mathcal{K}_2^\star(C, \mathbf{Ct}, \mathbf{x})$
$\boxed{\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}^{\mathsf{FE.KeyGen}(\mathsf{FE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s}')}$
Output: Sk_C

where $\mathbf{s}' = (s'_1, \ldots, s'_{n_c})$ is the randomness sampled by $\mathcal{E}_2^\star$.

Then, experiment H_2 is the same as H_1 except that the oracle $\mathcal{E}_1^\star$ is replaced by $\mathcal{E}_2^\star$ and receiver-coerce oracle is modified as above.

Hybrid H_3: Consider the following oracles:

$\mathcal{E}_3^\star(\mathbf{x}, \mathbf{y})$
$(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$\boxed{(\mathsf{Ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{Mpk}, (y_i, s'_i)))_{i\in[n_c]}}$
Output: $((\mathsf{Ct}_i), \emptyset)$

$\mathcal{K}_3^\star(C, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}^{\mathsf{FE.KeyGen}(\mathsf{FE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s}')$
Output: Sk_C

Then, experiment H_3 is the same as H_2 except that the oracle $\mathcal{E}_2^\star$ is replaced by $\mathcal{E}_3^\star$ and receiver-coerce oracle is modified as above.

Finally, notice that H_3 is exactly the faking experiment $\mathsf{FakeDenExp}$ where $\mathcal{E}_2 = \mathcal{E}_3^\star$ and $\mathcal{K}_2 = \mathcal{K}_3^\star$.

We now show that the relevant distinguishing probabilities between adjacent hybrids are negligible, which completes the proof.

Indistinguishability of H_1 and H_2: To prove indistinguishability we use the following sequence of hybrid experiments.

- Hybrid $H_{1,j}$, for $1 \le j \le n_c + 1$: This is the same as H_1 except that $\mathcal{E}_2^\star$ is used instead of $\mathcal{E}_1^\star$ and the following new receiver-coercer oracle is used:

$\mathcal{K}^{\star}_{1,j}(C, \mathbf{Ct}, \mathbf{x})$
$\mathsf{Sk}_C \leftarrow \mathsf{Sim.RecFake}_{2,j}^{\mathsf{FE.KeyGen}(\mathsf{FE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s}')$
Output: Sk_C

where $\mathbf{s}'$ is chosen by oracle $\mathcal{E}^{\star}_1$ and $\mathsf{Sim.RecFake}_2$ is defined as follow:

$\mathsf{Sim.RecFake}_{2,j}^{\mathsf{FE.KeyGen}(\mathsf{FE.Msk},\cdot)}(C, \mathbf{x}, \mathbf{s})$

The algorithm takes in input a circuit C, messages $\mathbf{x} = (x_1, \ldots, x_\ell)$, strings $\mathbf{s} = (s_1, \ldots, s_\ell)$ each in $\{0,1\}^\lambda$, and oracle access to the FE key generation algorithm. Then, for each $i \in [\ell]$, the algorithm chooses random t_i and t'_i in $\{0,1\}^{l(\lambda)}$.
Now we have two cases:

1. $i < j$: then the algorithm distinguishes between the following two cases:
   - If $C(x_i) = 1$, it sets $z_i = f_{s_i}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$.
   - If $C(x_i) = 0$, it sets $z'_i = f_{s_i}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.
2. $i \geq j$: then the algorithm chooses random $z_i, z'_i \in \{0,1\}^{L(\lambda)}$.

Finally, the algorithm computes $\mathsf{FE.Sk}_C = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$.

Then, notice that $H_1 = H_{1,1}$ and $H_2 = H_{1,n_c+1}$. Thus, it is sufficient to prove that $H_{1,k}$ is computational indistinguishable from $H_{1,k+1}$. This can be reduced to the security of the family of pseudo-random functions. In fact, notice that $H_{1,k+1}$ is the same as $H_{1,k}$ except that to set z_k (or z'_k deeding on $C(x_k)$), the PRF is used on input seed s'_k which is never used in any other part of the simulation.

More formally, suppose there exists a distinguisher $\mathcal{D}$ and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ for which $H_{1,k}$ and $H_{1,k+1}$ are not computationally indistinguishable. Then $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful adversary $\mathcal{B}$ for the pseudo-randomness of $\mathcal{F}$.

Specifically, $\mathcal{B}$ on input the security parameter λ an having oracle access to a function $\hat{f}$ which is either f_s for random seed $s \leftarrow \{0,1\}^\lambda$ or $F \leftarrow \mathcal{R}(l(\lambda), L(\lambda))$ where $\mathcal{R}(l(\lambda), L(\lambda))$ is the space of all possible functions $F : \{0,1\}^{l(\lambda)} \rightarrow \{0,1\}^{L(\lambda)}$, does the following.

- $\mathcal{B}$, generates $(\mathsf{Mpk}, \mathsf{Msk})$ by invoking the setup algorithm of FE. $\mathcal{B}$ runs $\mathcal{A}_0$ on input master public key Mpk and answers $\mathcal{A}_0$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ by using $(\mathsf{Mpk}, \mathsf{Msk})$. Eventually, $\mathcal{A}_0$ outputs $\mathbf{x}^\star = (x^\star_1, \ldots, x^\star_{n_c}), \mathbf{y}^\star = (y^\star_1, \ldots, y^\star_{n_c})$ and its state $\mathtt{st}$. Then $\mathcal{B}$, generates the challenge ciphertexts $\mathsf{Ct}^\star_1, \ldots, \mathsf{Ct}^\star_{n_c}$ by using Mpk, encrypting $\mathbf{x}$ or $\mathbf{y}$ depending on a chosen random bit b. Finally, $\mathcal{B}$ runs $\mathcal{A}_1$ on input challenge ciphertexts $\mathsf{Ct}^\star_1, \ldots, \mathsf{Ct}^\star_{n_c}$ and answers $\mathcal{A}_1$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ by using $(\mathsf{Mpk}, \mathsf{Msk})$. To answer receiver-coerce oracle queries, $\mathcal{B}$ first chooses $(s'_i \leftarrow \{0,1\}^\lambda)_{i \in [n_c] \setminus \{k\}}$, then on input a receiver-coerce query of the form $(C, \mathbf{Ct}, \mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_{n_c})$, $\mathcal{B}$ does the following: For each $i \in [n_c] \setminus \{k\}$, $\mathcal{B}$ chooses random t_i and t'_i in $\{0,1\}^{l(\lambda)}$. Then,

1. $i < k$: $\mathcal{B}$ distinguishes between the following two cases: (a) If $C(x_i) = 1$, it sets $z_i = f_{s'_i}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$. (b) If $C(x_i) = 0$, it sets $z'_i = f_{s'_i}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.
2. $i = k$: $\mathcal{B}$ uses its own oracle $\hat{f}$ as follows:
   - If $C(x_i) = 1$, it sets $z_i = \hat{f}(t_i)$ and chooses random $z'_i \in \{0,1\}^{L(\lambda)}$.
   - If $C(x_i) = 0$, it sets $z'_i = \hat{f}(t'_i)$ and chooses random $z_i \in \{0,1\}^{L(\lambda)}$.
3. $i \geq k$: $\mathcal{B}$ chooses random $z_i, z'_i \in \{0,1\}^{L(\lambda)}$.

Finally, $\mathcal{B}$ computes $\mathsf{FE.Sk}_C = \mathsf{FE.KeyGen}(\mathsf{FE.Msk}, \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'})$, and returns secret key $\mathsf{Sk}_C = (\mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}', \mathsf{FE.Sk}_C)$ as the answer of oracle $\mathcal{E}_1^\star$. Eventually, $\mathcal{A}_1$ returns its output and $\mathcal{B}$ passes it to the distinguisher $\mathcal{D}$ and returns $\mathcal{D}$'s output as its own output.

Now notice that if $\hat{f} = F$ then $\mathcal{B}$ is simulating Game $H_{1,k}$. On the other hand, if $\hat{f} = f_s$ then $\mathcal{B}$ is simulating Game $H_{1,k+1}$

Indistinguishability of H_2 and H_3: To prove indistinguishability we use the following sequence of hybrid experiments.

- Hybrid $H_{2,j}$, for $1 \leq j \leq n_c + 1$: This is the same as H_2 except that the following new oracle $\mathcal{E}^\star_{2,j}$ is used:

$\mathcal{E}^\star_{2,j}(\mathbf{x}, \mathbf{y})$
$(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
$(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$
Then, for $i \in [n_c]$, we have two cases:

1. $i < j$: $\mathsf{Ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{Mpk}, (y_i, s'_i))$,
2. $i \geq j$: $\mathsf{Ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{Mpk}, (x_i, s_i))$.

Output: $((\mathsf{Ct}_i), \emptyset)$

Then, notice then that $H_2 = H_{2,1}$ and $H_3 = H_{2,n_c+1}$. Thus, it is sufficient to prove that $H_{2,k}$ is computational indistinguishable from $H_{2,k+1}$. This follows from the assumed IND-Security of FE. In fact, notice that $H_{2,k+1}$ is the same as $H_{2,k}$ except that the k-th challenge ciphertext is for message (y_k, s'_k) instead of (x_k, s_k). Moreover, notice that for all the faked secret keys generated using the algorithm $\mathsf{Sim.RecFake}_{2,n_c+1}$ it holds than $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x_k, s_k) = C(x_k) = \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(y_k, s'_k)$.

More formally, suppose there exists a distinguisher $\mathcal{D}$ and adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ for which $H_{2,k}$ and $H_{2,k+1}$ are not computationally indistinguishable. Then $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful IND adversary $\mathcal{B}$ for FE. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input FE master public key Mpk and having oracle access to the FE key generation algorithm, runs $\mathcal{A}_0$ on input master public key Mpk and answers $\mathcal{A}_0$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ by using Mpk and its key generation oracle. Eventually, $\mathcal{A}_0$ outputs $\mathbf{x}^\star = (x^\star_1, \ldots, x^\star_{n_c}), \mathbf{y}^\star = (y^\star_1, \ldots, y^\star_{n_c})$ and its state st. Them $\mathcal{B}_0$, chooses $(s_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$ and $(s'_i \leftarrow \{0,1\}^\lambda)_{i\in[n_c]}$, and returns as its challenge messages $(y^\star_k, s'_k)$ and $(x^\star_k, s_k)$ and put in its state the state of $\mathcal{A}_0$ and its entire computation.

- $\mathcal{B}_1$ on input ciphertext $\mathsf{Ct}^\star$, which is the encryption of $(y_k^\star, s_k')$ or $(x_k^\star, s_k)$, and having oracle access to the FE key generation algorithm, generates challenge ciphertexts in the following way: For $j < k$, $\mathcal{B}_1$ sets $\mathsf{Ct}_j^\star = \mathsf{Encrypt}(\mathsf{Mpk}, (y_j^\star, s_j'))$, for $j > k$, $\mathcal{B}_1$ sets $\mathsf{Ct}_j^\star = \mathsf{Encrypt}(\mathsf{Mpk}, (x_j^\star, s_j))$, and for $j = k$, $\mathcal{B}_0$ set $\mathsf{Ct}_k^\star = \mathsf{Ct}^\star$. Finally, $\mathcal{B}_1$ runs $\mathcal{A}_1$ on input challenge ciphertexts $\mathsf{Ct}_1^\star, \ldots, \mathsf{Ct}_{n_c}^\star$ and answers $\mathcal{A}_1$'s queries to $\mathcal{O}_1$ and $\mathcal{O}_2$ and to the receiver-coerce oracle $\mathcal{K}$, which is implemented at this stage by $\mathsf{Sim.RecFake}_{2,n_c+1}$, by using Mpk and its own key generation oracle. Eventually, $\mathcal{A}_1$ returns its output and $\mathcal{B}_1$ passes it to the distinguisher $\mathcal{D}$ and returns $\mathcal{D}$'s output as its own output.

It remains to verify that $\mathcal{B}$ is valid IND adversary, meaning that all the key queries issued by $\mathcal{B}$ satisfy the game constraints with the respect to the challenge messages $(y_k^\star, s_k')$ and $(x_k^\star, s_k)$. We have the following two cases: (1) For query made by $\mathcal{A}$ to $\mathcal{O}_1$ or $\mathcal{O}_2$ of the form (C, x, y), $\mathcal{B}$ generates a ciphertext with the respect to a freshly chosen seed $\hat{s}$ and issues a secret key query to its oracle for circuit $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}$, where $\mathbf{z}$ and $\mathbf{z}'$ are related to $\hat{s}$ It holds then, under the constraints of the receiver deniable security game, $C(y_k^\star) = C(x_k^\star)$ then with overwhelming probability $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}((y_k^\star, s_k')) = C(y_k^\star) = C(x_k^\star) = \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}((x_k^\star, s_k))$, by definition of the trapdoor circuit and by noting that $s_k, s_k', \mathbf{t}, \mathbf{z}, \mathbf{t}', \mathbf{z}'$ are uncorrelated. (2) For a query issued to the receiver-coerce oracle for a circuit C, the corresponding secret key is generated by the algorithm $\mathsf{Sim.RecFake}_{2,n_c+1}$. By definition of this algorithm it holds that $\mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(x_k^\star, s_k) = C(x_k) = \mathsf{Trap}[C, \mathcal{F}]^{\mathbf{t},\mathbf{z},\mathbf{t}',\mathbf{z}'}(y_k^\star, s_k')$. This concludes the proof.

References

1. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013)
2. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013). http://eprint.iacr.org/2013/689
3. Apon, D., Fan, X., Liu, F.: Bi-deniable inner product encryption from LWE. IACR Cryptology ePrint Archive 2015:993 (2015)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 1. Springer, Heidelberg (2001)
5. Barbosa, M., Farshim, P.: On the semantic security of functional encryption schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 143–161. Springer, Heidelberg (2013)
6. 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)
7. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: possibility results, impossibility results and the quest for a general definition. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 218–234. Springer, Heidelberg (2013)

8. Bendlin, R., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: Lower and upper bounds for deniable public-key encryption. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 125–142. Springer, Heidelberg (2011)
9. Blundo, C., Iovino, V., Persiano, G.: Predicate encryption with partial public keys. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 298–313. Springer, Heidelberg (2010)
10. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
11. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013)
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
13. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014)
14. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
15. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th Annual ACM Symposium on Theory of Computing, pp. 639–648. ACM Press, May 1996
16. Damgård, I.B., 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)
17. De Caro, A., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 519–535. Springer, Heidelberg (2013)
18. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. In: 40th Annual Symposium on Foundations of Computer Science, pp. 523–534. IEEE Computer Society Press, October 1999
19. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual Symposium on Foundations of Computer Science, pp. 40–49. IEEE Computer Society Press, October 2013
20. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)
21. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure attribute based encryption from multilinear maps. Cryptology ePrint Archive, Report 2014/622 (2014). http://eprint.iacr.org/2014/622
22. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
23. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)

24. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing, pp. 545–554. ACM Press, June 2013
25. Iovino, V., Tang, Q., Żebrowski, K.: On the power of public-key functional encryption with function privacy. IACR Cryptology ePrint Archive 2015:470 (2015)
26. Iovino, V., Żebroski, K.: Simulation-based secure functional encryption in the random oracle model. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) LatinCrypt 2015. LNCS, vol. 9230, pp. 21–39. Springer, Heidelberg (2015)
27. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
28. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd Annual ACM Symposium on Theory of Computing, pp. 427–437. ACM Press, May 1990
29. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (Hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
30. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010). http://eprint.iacr.org/
31. O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 525–542. Springer, Heidelberg (2011)
32. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing, pp. 475–484. ACM Press, May/June 2014
33. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
34. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (2015)

Identity-Based Encryption

Identity-Based Cryptosystems and Quadratic Residuosity

Marc Joye(✉)

Technicolor, 175 S. San Antonio Road, Los Altos, CA 94022, USA
marc.joye@technicolor.com

Abstract. Three approaches are currently used for devising identity-based encryption schemes. They respectively build on pairings, quadratic residues (QR), and lattices. Among them, the QR-based scheme proposed by Cocks in 2001 is notable in that it works in standard RSA groups: its security relies on the standard quadratic residuosity assumption. But it has also a number of deficiencies, some of them have been subsequently addressed in follow-up works. Currently, one of the main limitations of Cocks' scheme resides in its apparent lack of structure. This considerably restricts the range of possible applications. For example, given two Cocks ciphertexts, it is unknown how to evaluate of a function thereof.

Cocks' scheme is believed to be non-homomorphic. This paper disproves this conjecture and proposes a constructive method for computing over Cocks ciphertexts. The discovery of the hidden algebraic structure behind Cocks encryption is at the core of the method. It offers a better understanding of Cocks' scheme. As a further illustration of the importance of the knowledge of the underlying structure, this paper shows how to anonymize Cocks ciphertexts without increasing their size or sacrificing the security.

Finally and of independent interest, this paper presents a simplified version of the abstract identity-based cryptosystem with short ciphertexts of Boneh, Gentry, and Hamburg.

Keywords: Public-key cryptography · Identity-based encryption · Cocks' scheme · Homomorphic encryption · Anonymous encryption · Public-key encryption with keyword search · Quadratic residuosity

1 Introduction

Identity-based cryptography is an extension of the public-key paradigm which was first put forward by Shamir [25]. As discussed in [20, Chapter 1], a major issue with public-key cryptography is the management of trust. Another issue to be dealt with is to recover the public key and accompanying certificate, verify it, and then only encrypt and send messages. Identity-based cryptography aims at solving these practical issues by simplifying the key management.

While identity-based signature schemes were quickly proposed (already in his 1984 paper, Shamir presented such a scheme), identity-based encryption (IBE)

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 225–254, 2016.
DOI: 10.1007/978-3-662-49384-7_9

schemes seem harder to develop and only came later. The first implementation of an IBE scheme was proposed by Desmedt and Quisquater [13] in 1986. It is however non-standard in the sense that it requires tamper-proof hardware for its security. The realization of a truly practical IBE scheme remained elusive until a breakthrough paper by Boneh and Franklin [7] in 2001, and concurrently by Sakai *et al.* [24]. The Boneh-Franklin IBE scheme makes use of bilinear maps. Its publication was quickly followed up by a large number of works. More recently, lattices were considered as a building block for constructing IBE schemes [16]. Again this gave rise to a number of follow-up works.

A totally different approach was described back in 2001 by Cocks in a short 4-page paper [12]. Cocks' IBE scheme only requires elementary mathematics. Encryption merely involves a couple of operations modulo an RSA modulus and the evaluation of Jacobi symbols. Its security rests on the standard quadratic residuosity assumption in the random oracle model. Despite its simplicity, Cocks' scheme received less attention from the research community, compared to the pairing-based or lattice-based constructions. We believe that this is mainly due to the apparent lack of structure behind Cocks' scheme. In this paper, we identify Cocks ciphertexts as elements of a certain algebraic group. This makes Cocks' scheme amenable to applications that were previously not possible. In particular, it can now be used in applications where computing over ciphertexts is required. Typical applications include electronic voting, auction systems, private information retrieval, or cloud computing.

Related Work. Since it appeared in 2001, a handful of variants of Cocks' IBE scheme have been proposed in the literature, aiming at enhancing certain features of the original scheme or offering extra properties.

Cocks' scheme is known not to be anonymous. The ciphertexts leak information about their recipient's identity. The anonymity aspect in Cocks-like cryptosystems was first considered by Di Crescenzo and Saraswat in [14] (and incidentally in [8]). Subsequently, Ateniese and Gasti [2] and, more recently, Clear *et al.* [11] proposed concurrent anonymous cryptosystems derived from Cocks' scheme. Table 1 in [11] gives a comparison of these two latter schemes. The scheme by Clear *et al.* features the best encryption and decryption times (i.e., 79 ms and 27 ms for a 128-bit message with a key-size of 1024 bits in their setting). The Ateniese-Gasti scheme is slower but has a smaller ciphertext expansion.

Cocks' scheme is mostly attractive when used with the hybrid encryption paradigm encrypting a short session key. Indeed messages are encrypted in a bit-by-bit fashion with Cocks' IBE scheme. It therefore looses its practicability when long messages need to be encrypted. The ciphertext expansion issue was addressed by Boneh, Gentry, and Hamburg [8]. They propose a space-efficient, anonymous IBE scheme based on the quadratic residuosity. However, the encryption in their scheme is time-consuming. Encryption time is quartic in the security parameter per message bit. Several possible trade-offs are discussed in [8, Sect. 5.3] and [19].

The work closest to ours is a recent paper by Clear, Hughes, and Tewari [10]. They develop a XOR-homomorphic variant of Cocks' scheme. This is elegantly achieved by seeing ciphertexts as elements (of the multiplicative group) in a certain quotient ring. The homomorphism property then naturally pops up as an application of the corresponding multiplication operation. A similar scheme was independently found —and generalized to higher power residue symbols— by Boneh, LaVigne, and Sabin [9]. As explained in [9], these schemes are however less efficient, bandwidth-wise, than the original Cocks' scheme.

Our Contributions. Motivated by the work of [10], we were interested in finding the exact algebraic structure behind Cocks' scheme. We quote from [23]:

> "We believe that studying and understanding the mathematics that underlies the associated cryptosystems is a useful aid to better understand their properties and their security."

Interestingly, the results of Rubin and Silverberg [23] (appearing earlier in [22]) were instrumental in our work. In a nutshell, we consider the torus $\mathbb{T}_1(\mathbb{F}_p) = \mathbb{F}_p^\times$ viewed as a 'degenerate' representation of the torus $\mathbb{T}_2(\mathbb{F}_p)$ where $\mathbb{F}_{p^2}$ is replaced with $\mathbb{F}_p(\delta)$ where $\delta \in \mathbb{F}_p^\times$. We then extend the setting modulo an RSA composite through Chinese remaindering. We show that the Cocks ciphertexts are squares in the so-obtained algebraic structure and form a quasi-group. The underlying group law yields the sought-after homomorphism. Compared to the approach in [10] there is no ciphertext expansion —the ciphertexts in [10] are twice longer. More importantly, it directly applies to Cocks' scheme. This is somewhat surprising. It points out that the original Cocks' IBE scheme is *inherently* homomorphic. In this regard, it shares similarities with the Goldwasser-Micali [public-key] encryption scheme [17]. We note that both schemes are semantically secure under the quadratic residuosity assumption and have comparable performance.

Another contribution is an anonymous variant of Cocks's scheme. Anonymous identity-based schemes are important cryptographic tools as they constitute the central building block for public-key encryption with keyword search (PEKS). The companion PEKS scheme derived from our anonymous IBE scheme is detailed in Appendix D. Compared to the earlier QR-based[1] scheme by Di Crescenzo and Saraswat [14], it reduces the size of the searchable ciphertexts by a typical factor of 2 without sacrificing the security.

Of independent interest, we present in Appendix E a simplified version of the abstract IBE system with short ciphertexts of Boneh, Gentry, and Hamburg [8].

2 Definitions and Notation

In this section, we review the classical notions of semantic security and of anonymity for identity-based encryption. We also formally present the quadratic residuosity assumption and a variant thereof.

[1] We note that the new security assumption introduced in [14] was later shown in [2] to be equivalent to the QR assumption.

2.1 Identity-Based Encryption

An *identity-based encryption scheme* [7] (or IBE in short) is defined as a tuple of four polynomial-time algorithms $(\mathtt{SETUP}, \mathtt{EXTRACT}, \mathtt{ENCRYPT}, \mathtt{DECRYPT})$:

Setup. The setup algorithm SETUP is a randomized algorithm that, taking a security parameter 1^κ as input, outputs the system parameters mpk together with the master secret key msk: $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{R} \mathtt{SETUP}(1^\kappa)$. The message space is denoted by $\mathcal{M}$.

Key derivation. The key derivation algorithm EXTRACT takes as input an identity id and, using the master secret key msk, returns a secret key for the user with identity id: $\mathsf{usk} \leftarrow \mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$.

Encryption. The encryption algorithm ENCRYPT is a randomized algorithm that takes as input an identity id and a plaintext $m \in \mathcal{M}$, and returns a ciphertext C. We write $C \leftarrow \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}, m)$.

Decryption. The decryption algorithm DECRYPT takes as input secret key usk (corresponding to identity id) and a ciphertext C and returns the corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $m \leftarrow \mathtt{DECRYPT}_{\mathsf{usk}}(C)$ if C is a valid ciphertext and $\perp \leftarrow \mathtt{DECRYPT}_{\mathsf{usk}}(C)$ if it is not.

It is required that $\mathtt{DECRYPT}_{\mathsf{usk}}(\mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}, m)) = m$ for any identity id and all messages $m \in \mathcal{M}$, where $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{R} \mathtt{SETUP}(1^\kappa)$ and $\mathsf{usk} \leftarrow \mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$.

2.2 Security Notions

Semantic Security. The notion of *indistinguishability of encryptions* [17] captures a strong notion of data-privacy: The adversary should not learn anything about a plaintext given its encryption, beyond the length of the plaintext. The definitions for the public-key setting naturally extend to the identity-based paradigm. The standard definition is strengthened by allowing the adversary to issue chosen private-key extraction queries [7].

We view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. This corresponds to adversary $\mathcal{A}$ running in two stages. Upon receiving the system parameters mpk, in the "find" stage, algorithm $\mathcal{A}_1$ issues private-key extraction queries $\mathsf{id}_1, \ldots, \mathsf{id}_{n_1}$ and receives back the private key usk_i corresponding to identity id_i: $\mathsf{usk}_i \leftarrow \mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id}_i)$. The queries may be asked adaptively. Once the adversary decides not to make further oracle queries, it outputs a challenge identity id^* (with $\mathsf{id}^* \neq \mathsf{id}_i$, $1 \leq i \leq n_1$), two (different) equal-size messages m_0 and $m_1 \in \mathcal{M}$ (where $\mathcal{M}$ denotes the message space), and some state information s. In the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext C which is the encryption of m_b for identity id^* where b is chosen uniformly at random in $\{0, 1\}$. Algorithm $\mathcal{A}_2$ can issue more private-key extraction queries $\mathsf{id}_{n_1+1}, \ldots, \mathsf{id}_{n_2}$; the only restriction is that $\mathsf{id}_i \neq \mathsf{id}^*$, $n_1 < i \leq n_2$. The goal of $\mathcal{A}_2$ is to guess the value of b from s and C. Formally, a public-key encryption

scheme is said *indistinguishable* (or *semantically secure*) if

$$\Pr\left[\begin{array}{l}(\mathsf{mpk},\mathsf{msk}) \stackrel{R}{\leftarrow} \mathtt{SETUP}(1^\kappa),\\ (\mathsf{id}^*, m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}),\ : \mathcal{A}_2^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(s, C) = b\\ b \stackrel{R}{\leftarrow} \{0,1\}, C \leftarrow \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}^*, m_b)\end{array}\right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by SETUP and over the random coins of the adversary.

Adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ can encrypt any message of its choice, for any identity of its choice. In other words, the adversary can mount chosen-identity, chosen-plaintext attacks (IND-ID-CPA). Hence, we write IND-ID-CPAthe security notion achieved by a semantically secure identity-based encryption scheme.

Remark 1. When the message space is $\mathcal{M} = \{0,1\}$, the previous probability simplifies to

$$\Pr\left[\begin{array}{ll}(\mathsf{mpk},\mathsf{msk}) \stackrel{R}{\leftarrow} \mathtt{SETUP}(1^\kappa),\\ (\mathsf{id}^*, s) \leftarrow \mathcal{A}_1^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}), & : \mathcal{A}_2^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(s, C) = b\\ b \stackrel{R}{\leftarrow} \{0,1\}, C \leftarrow \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}^*, b)\end{array}\right] - \frac{1}{2}.$$

Anonymity. Analogously, the notion of *anonymity* captures a strong requirement about privacy: a ciphertext should not reveal the identity of the recipient. More formally, it is defined as a straightforward adaptation of key privacy [4] to the identity-based paradigm [1].

As before, we view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. In the "find" stage, algorithm $\mathcal{A}_1$ issues private-key extraction queries $\mathsf{id}_1, \ldots, \mathsf{id}_{n_1}$ and receives back the private key usk_i corresponding to identity id_i: $\mathsf{usk}_i \leftarrow \mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id}_i)$. The queries may be asked adaptively. Once the adversary decides not to make further oracle queries, it outputs two (different) challenge identities id_0^* and id_1^* (with $\mathsf{id}_0^*, \mathsf{id}_1^* \neq \mathsf{id}_i$, $1 \leq i \leq n_1$), a message $m \in \mathcal{M}$, and some state information s. In the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext C which is the encryption of m for identity id_b^* where b is chosen uniformly at random in $\{0,1\}$. Algorithm $\mathcal{A}_2$ can issue more private-key extraction queries $\mathsf{id}_{n_1+1}, \ldots, \mathsf{id}_{n_2}$; the only restriction is that $\mathsf{id}_i \neq \mathsf{id}_0^*, \mathsf{id}_1^*$, $n_1 < i \leq n_2$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and C.

An IBE scheme is said to be *anonymous* if

$$\Pr\left[\begin{array}{l}(\mathsf{mpk},\mathsf{msk}) \stackrel{R}{\leftarrow} \mathtt{SETUP}(1^\kappa),\\ (\mathsf{id}_0^*, \mathsf{id}_1^*, m, s) \leftarrow \mathcal{A}_1^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}),\ : \mathcal{A}_2^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(s, C) = b\\ b \stackrel{R}{\leftarrow} \{0,1\}, C \leftarrow \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}_b^*, m)\end{array}\right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by SETUP and over the random coins of the adversary. We write ANO-ID-CPA the corresponding security notion achieved by an anonymous IBE scheme.

Semantic Security and Anonymity. Of course, the goals of indistinguishability and anonymity can be combined to give rise to the ANO-IND-ID-CPA security notion. Halevi's sufficient condition [18] was extended to IBE schemes in [1]. Namely, an IBE scheme is ANO-IND-ID-CPA if it is IND-ID-CPA and if

$$\Pr\left[\begin{array}{l}(\mathsf{mpk},\mathsf{msk}) \xleftarrow{R} \mathtt{SETUP}(1^\kappa),\\ (\mathsf{id}_0^*,\mathsf{id}_1^*,m,s) \leftarrow \mathcal{A}_1^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(\mathsf{mpk}),\\ b \xleftarrow{R} \{0,1\}, r \xleftarrow{R} \mathcal{M} \text{ and } |r|_2 = |m|_2,\\ C \leftarrow \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}_b^*, r)\end{array} : \mathcal{A}_2^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)}(s,C) = b\right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by SETUP and over the random coins of the adversary. The difference is that a *random* message r is encrypted as opposed to the message m chosen by $\mathcal{A}$; the only restriction being that r and m must be of equal length.

2.3 Complexity Assumptions

It is useful to introduce some notation. Let $N = pq$ be the product of two primes p and q. The set of integers whose Jacobi symbol is 1 is denoted by $\mathbb{J}_N$, $\mathbb{J}_N = \left\{a \in \mathbb{Z}_N^* \mid \left(\frac{a}{N}\right) = 1\right\}$; the set of quadratic residues is denoted by $\mathbb{QR}_N$, $\mathbb{QR}_N = \left\{a \in \mathbb{Z}_N^* \mid \left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1\right\}$. Notice that $\mathbb{QR}_N$ is a subset of $\mathbb{J}_N$.

This leads to the following computational assumption [17]. Basically, it says that quadratic residues cannot be distinguished from quadratic non-residues modulo an RSA composite $N = pq$.

Definition 1 (Quadratic Residuosity Assumption). *Let* RSAgen *be a probabilistic algorithm which, given a security parameter κ, outputs primes p and q and their product $N = pq$. The* Quadratic Residuosity (QR) assumption *relative to* RSAgen *asserts that the success probability defined as the distance*

$$\left|Pr[\mathcal{D}(x,N) = 1 \mid x \xleftarrow{R} \mathbb{QR}_N] - Pr[\mathcal{D}(x,N) = 1 \mid x \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N]\right|$$

is negligible for any probabilistic polynomial-time distinguisher $\mathcal{D}$; the probabilities are taken over the experiment of running $(N,p,q) \leftarrow \mathsf{RSAgen}(1^\kappa)$ and choosing at random $x \in \mathbb{QR}_N$ and $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$.

A stronger assumption is introduced in [8]. It says that the QR assumption holds in the presence of a hash square-root oracle. More formally, the assumption is defined as follows.

Definition 2 (Interactive Quadratic Residuosity Assumption). *Again let* RSAgen *be a probabilistic algorithm which, given a security parameter κ, outputs primes p and q and their product $N = pq$. Let also $\mathcal{H}$ be a hash function that on input an arbitrary bit-string returns an element in $\mathbb{J}_N$ and let $\mathcal{O}$ be a hash square-root oracle that maps an input pair (N,s) to one of $\mathcal{H}(s)^{1/2} \bmod$*

N or $(u\mathcal{H}(x))^{1/2} \bmod N$, for some quadratic non-residue $u \in \mathbb{J}_N$. The Interactive Quadratic Residuosity (IQR) assumption *asserts that the success probability defined as the distance*

$$\left| Pr[\mathcal{D}^{\mathcal{O}}(x, N) = 1 \mid x \xleftarrow{R} \mathbb{QR}_N] - Pr[\mathcal{D}^{\mathcal{O}}(x, N) = 1 \mid x \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N] \right|$$

is negligible for any probabilistic polynomial-time distinguisher $\mathcal{D}$; the probabilities are taken over the experiment of running $(N, p, q) \leftarrow \mathsf{RSAgen}(1^\kappa)$, choosing at random oracle $\mathcal{O}$, and choosing at random $x \in \mathbb{QR}_N$ and $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$.

Remark 2. As noted in [8] the IQR assumption is equivalent to the QR assumption in the random oracle model [5].

3 Review of Cocks' Scheme

In 2001, Cocks published an identity-based encryption scheme that does not rely on pairings over elliptic curves [12]. Cocks' scheme works in standard RSA groups and its security relies on the quadratic residuosity assumption (in the random oracle model). The encryption processes one bit at a time. To simplify the presentation, we assume that messages being encrypted are in the set $\{-1, 1\}$. For example, the map $\mu : \{0, 1\} \to \{-1, 1\},\ b \mapsto m = (-1)^b$ maps a bit b to a message $m \in \mathcal{M} = \{\pm 1\}$. The inverse map is given by $\mu^{-1}(m) = (1 - m)/2$.

3.1 Description

Cocks' scheme proceeds as follows.

$\mathtt{SETUP}(1^\kappa)$. Given a security parameter κ, SETUP generates an RSA modulus $N = pq$ where p and q are prime. It also selects an element $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$. The system parameters are $\mathsf{mpk} = \{N, u, \mathcal{H}\}$ where $\mathcal{H}$ is a cryptographic hash function mapping bit-strings to $\mathbb{J}_N$. The master secret key is $\mathsf{msk} = \{p, q\}$.

$\mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$. Using hash function $\mathcal{H}$, EXTRACT sets $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. If $R_{\mathsf{id}} \in \mathbb{QR}_N$ it computes $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$; otherwise it computes $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2}\ N$. EXTRACT returns user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

$\mathtt{ENCRYPT}(\mathsf{id}, m)$ To encrypt a message $m \in \{\pm 1\}$ for user with identity id, ENCRYPT chooses at random $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$ such that $\left(\frac{t}{N}\right) = \left(\frac{\bar{t}}{N}\right) = m$. It then computes

$$c = t + \frac{R_{\mathsf{id}}}{t} \bmod N \quad \text{and} \quad \bar{c} = \bar{t} + \frac{uR_{\mathsf{id}}}{\bar{t}} \bmod \mathrm{N}$$

where $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. The returned ciphertext is $C = (c, \bar{c})$.

$\mathtt{DECRYPT}_{\mathsf{usk}}(C)$. From $\mathsf{usk} = \{r_{\mathsf{id}}\}$ and $C = (c, \bar{c})$, if ${r_{\mathsf{id}}}^2 \equiv \mathcal{H}(\mathsf{id}) (\mathrm{mod}\, N)$, DECRYPT sets $\gamma = c$; otherwise it sets $\gamma = \bar{c}$. Plaintext m is then recovered as

$$m = \left(\frac{\gamma + 2r_{\mathsf{id}}}{N}\right).$$

Remark 3. The above description is a generalization of the original scheme. In [12], Cocks considers Blum integers; namely, RSA moduli $N = pq$ with $p, q \equiv 3 \pmod 4$. Doing so, it follows that $\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = -1$ and therefore $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$. The original scheme corresponds to the choice $u = -1$. The above description also slightly generalizes the one offered in [8, Appendix A] in that parameter u is not necessarily chosen as a *random* quadratic non-residue in $\mathbb{J}_N$.

Remark 4. Alternatively, the decryption algorithm can recover plaintext m as $m = \left(\frac{\gamma - 2r_{\mathsf{id}}}{N}\right)$. The correctness of the decryption follows by remarking that when ${r_{\mathsf{id}}}^2 \equiv \mathcal{H}(\mathsf{id}) \pmod N$, $\gamma \pm 2r_{\mathsf{id}} \equiv t(1 \pm \frac{r_{\mathsf{id}}}{t})^2 \pmod N$ yielding $\left(\frac{\gamma \pm 2r_{\mathsf{id}}}{N}\right) = \left(\frac{t}{N}\right) = m$. Likewise, when ${r_{\mathsf{id}}}^2 \equiv u\mathcal{H}(\mathsf{id}) \pmod N$, $\gamma \pm 2r_{\mathsf{id}} \equiv \bar{t}(1 \pm \frac{r_{\mathsf{id}}}{t})^2 \pmod N$ and thus $\left(\frac{\gamma \pm 2r_{\mathsf{id}}}{N}\right) = \left(\frac{\bar{t}}{N}\right) = m$.

3.2 Security Analysis

The next proposition shows that the generalized Cocks' scheme is semantically secure under the QR assumption in the random oracle model. Equivalently, as mentioned in Remark 2, the scheme is semantically secure under the IQR assumption in the standard model.

Proposition 1. *The scheme of Sect. 3.1 is* IND-ID-CPA *under the quadratic residuosity assumption in the random oracle model.*

Proof. The proof can be found in Appendix A. □

4 A Useful Representation

Let $\mathbb{F}_q$ denote the finite field with q elements, where $q = p^r$ is a prime power. The order of the multiplicative group $\mathbb{F}_{p^r}^\times = \mathbb{F}_{p^r} \setminus \{0\}$ is $p^r - 1$. Note that $p^r - 1 = \prod_{d|r} \Phi_d(p)$ where $\Phi_d(x)$ represents the d-th cyclotomic polynomial. We let $\mathsf{G}_{p,r} \subseteq \mathbb{F}_{p^r}^\times$ denote the cyclic subgroup of order $\Phi_r(p)$. In [22], Rubin and Silverberg identify $\mathsf{G}_{p,r}$ with the $\mathbb{F}_p$-points of an algebraic torus. Namely, they consider

$$\mathbb{T}_r(\mathbb{F}_p) = \{\alpha \in \mathbb{F}_{p^r}^\times \mid N_{\mathbb{F}_{p^r}/F}(\alpha) = 1 \text{ whenever } \mathbb{F}_p \subseteq F \subsetneq \mathbb{F}_{p^r}\},$$

that is, the elements of $\mathbb{F}_{p^r}^\times$ whose norm is one down to every intermediate subfield F. Their key observation is that $\mathbb{T}_r(\mathbb{F}_p)$ forms a group whose elements can be represented with only $\phi(r)$ elements of $\mathbb{F}_p$, where ϕ denotes Euler's totient function. The compression factor is thus of $r/\phi(r)$ over the field representation [15,22].

4.1 Parametrization of $\mathbb{T}_2(\mathbb{F}_p)$

This corresponds to the case $r = 2$. We review the explicit representation for $\mathbb{T}_2(\mathbb{F}_p)$ presented in [22, Sect. 5.2].

For simplicity, we assume that p is an *odd* prime. Let $\Delta = \delta^2 \in \mathbb{F}_p^\times$ with $\delta \notin \mathbb{F}_p$. Then $\mathbb{T}_2(\mathbb{F}_p)$ is the multiplicative group given by

$$\mathbb{T}_2(\mathbb{F}_p) = \big\{x + \delta y \mid x, y \in \mathbb{F}_p \text{ and } x^2 - \Delta y^2 = 1\big\}.$$

Define the map $\psi : \mathbb{F}_p \to \mathbb{T}_2(\mathbb{F}_p),\ u \mapsto \frac{u+\delta}{u-delta} = \frac{u^2+\Delta}{u^2-\Delta} + \delta\,\frac{2u}{u^2-\Delta}$. The inverse map is given by $\psi^{-1} : \mathbb{T}_2(\mathbb{F}_p) \setminus \{1\} \to \mathbb{F}_p,\ v \mapsto \frac{\delta(v+1)}{v-1}$. By augmenting $\mathbb{F}_p$ with a special symbol ∞ and defining $\psi(\infty) = 1$, maps ψ and ψ^{-1} extend naturally to give an isomorphism $T_2(\mathbb{F}_p) \xrightarrow{\sim} \mathbb{F}_p \cup \{\infty\}$.

4.2 An Alternative Representation For $(\mathbb{Z}/N\mathbb{Z})^\times$

One may wonder what happens if Δ is chosen as a quadratic residue in the Rubin-Silverberg representation for $\mathbb{T}_2(\mathbb{F}_p)$. As will become apparent in Sect. 4.3, this seemingly useless setting has practical consequences. We start with the multiplicative group $\mathbb{F}_p^\times$ and then extend our results to $(\mathbb{Z}/N\mathbb{Z})^\times$ through Chinese remaindering.

The Group $\mathcal{F}_{p,\Delta}$. Let p be an odd prime. We henceforth assume that $\delta \in \mathbb{F}_p^\times$ and thus that $\Delta = \delta^2 \in \mathbb{QR}_p$. In this case, the torus $\mathbb{T}_2(\mathbb{F}_p)$ becomes isomorphic to $\mathbb{T}_1(\mathbb{F}_p) = \mathbb{F}_p^\times$. Note also that map ψ as given in Sect. 4.1 is no longer defined at $u = \delta$. Moreover, when $\delta \in \mathbb{F}_p^\times$, $\psi(-\delta) = 0$ cannot be expressed as $\psi(-\delta) = x+\delta y$ for some $x, y \in \mathbb{F}_p$ with $x^2 - \Delta y^2 = 1$. So, we define the set

$$\mathcal{F}_{p,\Delta} = (\mathbb{F}_p \setminus \{\pm\delta\}) \cup \{\infty\} = \{u \in \mathbb{F}_p \mid u^2 \neq \Delta\} \cup \{\infty\}$$

and restrict map ψ to $\mathcal{F}_{p,\Delta}$:

$$\psi : \mathcal{F}_{p,\Delta} \to \mathbb{F}_p^\times,\ u \mapsto \begin{cases} \frac{u+\delta}{u-\delta} & \text{if } u \neq \infty\,, \\ 1 & \text{otherwise}\,. \end{cases}$$

For completeness, we show that $\mathcal{F}_{p,\Delta}$ equipped with the group law $\circledast$ (defined hereafter) and $\mathbb{F}_p^\times$ are isomorphic. Clearly, the map $\psi : \mathcal{F}_{p,\Delta} \to \mathbb{F}_p^\times$ is injective and thus defines a bijection. Indeed, suppose $\psi(u_1) = \psi(u_2)$ for some $u_1, u_2 \in \mathcal{F}_{p,\Delta}$. If $\psi(u_1) \neq 1$, this implies $(u_1 + \delta)(u_2 - \delta) = (u_2 + \delta)(u_1 - \delta)$ and in turn $u_1 = u_2$; it $\psi(u_1) = 1$ then again this implies $u_1 = u_2\ (= \infty)$ since $(u + \delta)/(u - \delta) \neq 1$ for every $u \in \mathcal{F}_{p,\Delta}$. The inverse map is given by

$$\psi^{-1} : \mathbb{F}_p^\times \to \mathcal{F}_{p,\Delta},\ v \mapsto \begin{cases} \frac{\delta(v+1)}{v-1} & \text{if } v \neq 1\,, \\ \infty & \text{otherwise.} \end{cases}$$

Furthermore, map ψ yields a homomorphism from $\mathcal{F}_{p,\Delta}$ to $\mathbb{F}_p^\times$. Let $u_1, u_2 \in \mathcal{F}_{p,\Delta} \setminus \{\infty\}$ with $u_1 \neq -u_2$. Then we have

$$\psi(u_1)\cdot\psi(u_2) = \frac{(u_1+\delta)(u_2+\delta)}{(u_1-\delta)(u_2-\delta)} = \frac{u_1u_2+\Delta+\delta(u_1+u_2)}{u_1u_2+\Delta-\delta(u_1+u_2)} = \frac{\frac{u_1u_2+\Delta}{u_1+u_2}+\delta}{\frac{u_1u_2+\Delta}{u_1+u_2}-\delta}$$
$$= \psi(u_3) \quad \text{where } u_3 = \tfrac{u_1u_2+\Delta}{u_1+u_2}.$$

Note also that $\psi(\infty) = 1$ and that $\frac{1}{\psi(u_1)} = \frac{u_1-\delta}{u_1+\delta} = \frac{-u_1+\delta}{-u_1-\delta} = \psi(-u_1)$.

We write $\mathcal{F}_{p,\Delta}$ multiplicatively and use $\circledast$ to denote its group law. In more detail, we have:

- the neutral element is ∞: $u \circledast \infty = \infty \circledast u = u$ for all $u \in \mathcal{F}_{p,\Delta}$;
- the inverse of $u \in \mathcal{F}_{p,\Delta} \setminus \{\infty\}$ is $-u$: $u \circledast (-u) = (-u) \circledast u = \infty$;
- given $u_1, u_2 \in \mathcal{F}_{p,\Delta} \setminus \{\infty\}$, their product is given by:

$$u_1 \circledast u_2 = \begin{cases} \frac{u_1u_2+\Delta}{u_1+u_2} & \text{if } u_1 \neq -u_2\,, \\ \infty & \text{otherwise.} \end{cases}$$

The Group $\mathcal{Z}_{N,\Delta}$. The previous setting naturally extends through Chinese remaindering. Let $N = pq$ be an RSA modulus. Then

$$\mathcal{Z}_{N,\Delta} := \mathcal{F}_{p,\Delta} \times \mathcal{F}_{q,\Delta} \cong (\mathbb{Z}/N\mathbb{Z})^\times \tag{1}$$

is a group w.r.t. $\circledast$ and has order $\phi(N)$.

For each element $u \in \mathcal{Z}_{N,\Delta}$, there exists a unique pair of elements $u_p \in \mathcal{F}_{p,\Delta}$ and $u_q \in \mathcal{F}_{q,\Delta}$ such that $u \bmod p = u_p$ and $u \bmod q = u_q$. We denote this equivalence by $u = [u_p, u_q]$ and let $\infty = [\infty_p, \infty_q]$ represent the neutral element. We also define the subset $\widetilde{\mathcal{Z}}_{N,\Delta} := \big((\mathcal{F}_{p,\Delta} \setminus \{\infty_p\}) \times (\mathcal{F}_{q,\Delta} \setminus \{\infty_q\})\big) \cup \{\infty\}$.

Efficient methods for working in $\mathcal{Z}_{N,\Delta}$ are discussed in Appendix B.

Remark 5. Please note that 0 is of order 2 as an element of $\mathcal{Z}_{N,\Delta}$, namely $0 \circledast 0 = \infty$. Note also that for any $u \in \mathcal{Z}_{N,\Delta}$, $u \neq 0, \infty$, we have $u \circledast 0 = \Delta/u$.

4.3 The Subset $\mathcal{S}_{N,\Delta}$ of Squares in $\widetilde{\mathcal{Z}}_{N,\Delta}$

We now have all ingredients to introduce the useful quasi-group $\mathcal{S}_{N,\delta}$. Let $N = pq$ be an RSA modulus and let $\Delta = \delta^2 \in \mathbb{QR}_N$. Consider the set of squares in $\widetilde{\mathcal{Z}}_{N,\Delta}$, namely $(\widetilde{\mathcal{Z}}_{N,\Delta})^2 = \{u \circledast u \mid u \in \widetilde{\mathcal{Z}}_{N,\Delta}\} \subset (\mathcal{Z}_{N,\Delta})^2$, or more exactly, the subset

$$\mathcal{S}_{N,\Delta} \stackrel{\text{def}}{=} \left\{ \tfrac{u^2+\Delta}{2u} \mid u \in (\mathbb{Z}/N\mathbb{Z})^\times \text{ and } \gcd(u^2-\Delta, N) = 1 \right\} \tag{2}$$
$$= (\widetilde{\mathcal{Z}}_{N,\Delta})^2 \setminus \{\infty\}.$$

The set $\mathcal{S}_{N,\Delta}$ almost defines a group: it contains all elements s of the group $(\mathcal{Z}_{N,\Delta})^2 = \{s = u \circledast u \mid u \in \mathcal{Z}_{N,\Delta}\}$ minus elements of the form $[s_p, \infty_q]$ or

$[\infty_p, s_q]$ (and $\infty = [\infty_p, \infty_q]$). Since it is a subset of $(\mathcal{Z}_{N,\Delta})^2$, $\mathcal{S}_{N,\Delta}$ is endowed with the $\circledast$-law. In practice, for cryptographic applications, working in $(\widetilde{\mathcal{Z}}_{N,\Delta})^2 = \mathcal{S}_{N,\Delta} \cup \{\infty\}$ rather than in $(\mathcal{Z}_{N,\Delta})^2$ does not really matter since the probability that operation $\circledast$ is not defined on $(\widetilde{\mathcal{Z}}_{N,\Delta})^2$ is negligible.

What makes the quasi-group $\mathcal{S}_{N,\Delta}$ special is that, up to a scaling factor of two, it represents the set of all valid [components of] Cocks ciphertexts. Before making this statement clear, we need to explain what is meant by 'valid component'. This follows from the next lemma, adapted from [12, Sect. 5]. Basically, it shows that, given a (generalized) Cocks ciphertext $C = (c, \bar{c})$, among its two components c and $\bar{c}$, one carries no information whatsoever about the corresponding plaintext. The component that yields the plaintext is called *valid component*.

Lemma 1. *Using the notations of Sect. 3.1, let $C = (c, \bar{c})$ be a (generalized) Cocks ciphertext. If $\mathcal{H}(\mathsf{id}) \notin \mathbb{QR}_N$ then the component c corresponds with the same probability to the encryption of message $m = 1$ or $m = -1$. Conversely, if $u\,\mathcal{H}(\mathsf{id}) \notin \mathbb{QR}_N$ then the component $\bar{c}$ corresponds with the same probability to the encryption of message $m = 1$ or $m = -1$.*

Proof. Suppose that $\mathcal{H}(\mathsf{id}) \notin \mathbb{QR}_N$ (i.e., $\mathcal{H}(\mathsf{id}) \in \mathbb{J}_N \setminus \mathbb{QR}_N$). Let $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. We have $c = t + \frac{R_{\mathsf{id}}}{t} \bmod N$ for some random $t \in (\mathbb{Z}/N\mathbb{Z})^\times$ such $\left(\frac{t}{N}\right) = m$. Consider also $t_1, t_2, t_3 \in (\mathbb{Z}/N\mathbb{Z})^\times$ such that

- $t_1 \equiv t \pmod p$, $t_1 \equiv R_{\mathsf{id}}/t \pmod q$;
- $t_2 \equiv R_{\mathsf{id}}/t \pmod p$, $t_2 \equiv t \pmod q$;
- $t_3 \equiv R_{\mathsf{id}}/t \pmod p$, $t_3 \equiv R_{\mathsf{id}}/t \pmod q$.

Note that the condition $\left(\frac{t}{N}\right) = m$ implies $\left(\frac{t_1}{N}\right) = \left(\frac{t_2}{N}\right) = \left(\frac{t_3}{N}\right) = m$. The four possible values t, t_1, t_2, and t_3 are equally likely since $c \equiv t + R_{\mathsf{id}}/t \equiv t_1 + R_{\mathsf{id}}/t_1 \equiv t_2 + R_{\mathsf{id}}/t_2 \equiv t_3 + R_{\mathsf{id}}/t_3 \pmod N$. At the same time, since $R_{\mathsf{id}} \in \mathbb{J}_N \setminus \mathbb{QR}_N$ we also have $\left(\frac{t}{N}\right) = \left(\frac{t_3}{N}\right) \neq \left(\frac{t_1}{N}\right) = \left(\frac{t_2}{N}\right)$. Hence, component c leaks no information about $\left(\frac{t}{N}\right)$; it has the same probability to be 1 or -1.

The case $u\,\mathcal{H}(\mathsf{id}) \notin \mathbb{QR}_N$ is proved similarly. □

With the notations of Sect. 3.1, from a ciphertext $C = (c, \bar{c})$, letting $\Delta = \mathcal{H}(\mathsf{id}) \in \mathbb{QR}_N$ (resp. $\Delta = u\,\mathcal{H}(\mathsf{id}) \in \mathbb{QR}_N$) and $\gamma = c$ (resp. $\bar{c}$), we have

$$\frac{\gamma}{2} \in \mathcal{S}_{N,\Delta} \implies \exists \tau \in (\mathbb{Z}/N\mathbb{Z})^\times \text{ with } \gcd(\tau^2 - \Delta, N) = 1 \text{ such that } \gamma = \tfrac{\tau^2+\Delta}{\tau}.$$

In other words, up to a factor of two, the valid component of C is an element of $\mathcal{S}_{N,\Delta}$; i.e., $\frac{\gamma}{2} \in \mathcal{S}_{N,\Delta}$. By Lemma 1, the other component does not matter. Note also that the condition $\gcd(\tau^2 - \Delta, N) = 1$ is implicit in the (generalized) Cocks encryption since $\left(\frac{\tau}{N}\right) = m \in \{\pm 1\}$ where $\tau = t$ (resp. $\tau = \bar{t}$) and decryption is obtained as

$$\left(\frac{\gamma \pm 2\delta}{N}\right)$$

which is 0 when $\gcd(\tau^2 - \Delta, N) \neq 1 \iff \tau \equiv \pm\delta (\mathrm{mod}\, \{p, q\})$. The condition $\tau \in (\mathbb{Z}/N\mathbb{Z})^\times$ (instead of $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$) is trivially satisfied since $\left(\frac{\tau}{N}\right) \in \{\pm 1\}$.

5 Computing over Cocks Ciphertexts

Homomorphic encryption is a form of encryption which allows combining two ciphertexts through a non-private operation that results in a third ciphertext which, when decrypted, yields a plaintext that is the combination of the corresponding two plaintexts through a specific operation. Mathematically, for two ciphertexts $C_1 = \texttt{ENCRYPT}(m_1)$ and $C_2 = \texttt{ENCRYPT}(m_2)$, there exists a non-private operation ∂ such that $C_1 \,\partial\, C_2 = \texttt{ENCRYPT}(m_1 \star m_2)$ for some specific operation $\star$. Examples of known operations $\star$ include (modular) addition and (modular) multiplication.

5.1 HOM Procedure

Consider the following procedure **HOM**. It takes as input two elements $x_1, x_2 \in \mathbb{Z}/N\mathbb{Z}$ and an element $\Gamma \in \mathbb{J}_N$, and outputs an element $z \in \mathbb{Z}/N\mathbb{Z}$. We write $z = \mathbf{HOM}(x_1, x_2, \Gamma)$.

1: **procedure HOM**(x_1, x_2, Γ)
2: Define $D = x_1 x_2 + 4\Gamma \bmod N$ and $U = x_1 + x_2 \bmod N$;
3: Select $t \in \mathbb{Z}/N\mathbb{Z}$ such that $\left(\frac{\theta}{N}\right) = 1$ where

$$\theta = t\,D + (t^2 + \Gamma)U \bmod N\,;$$

4: Evaluate

$$z = \frac{(t^2 + \Gamma)D + 4\Gamma t U}{\theta} \bmod N\,;$$

5: Return z.
6: **end procedure**

Remark 6. Note that when $\left(\frac{U}{N}\right) = \left(\frac{x_1+x_2}{N}\right) = 1$, we can take $t = 0$ (in Procedure **HOM**, Line 3). This yields $\theta = \Gamma U \bmod N\,(\in \mathbb{J}_N)$ and in turn $z = D/U \bmod N$.

5.2 Application

The **HOM** procedure allows for computing over two Cocks ciphertexts.

Specifically, suppose we are given two ciphertexts $C_1 = \{c_1, \bar{c}_1\}$ and $C_2 = \{c_2, \bar{c}_2\}$ that are the respective encryption of two messages m_1 and m_2 for a same identity, say id. The system parameters are $\mathsf{mpk} = \{N, u, \mathcal{H}\}$ where $N = pq$ is an RSA modulus, u is a quadratic non-residue in $\mathbb{J}_N$, and $\mathcal{H}$ is a hash function mapping bit-strings to $\mathbb{J}_N$. As in the description given in Sect. 3.1, we assume that the message space is $\mathcal{M} = \{\pm 1\}$.

Let $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. Two applications of the **HOM** procedure yields a third ciphertext $C_3 = (c_3, \bar{c_3})$ obtained as

$$c_3 = \mathbf{HOM}(c_1, c_2, R_{\mathsf{id}}) \quad \text{and} \quad \bar{c}_3 = \mathbf{HOM}(\bar{c}_1, \bar{c}_2, uR_{\mathsf{id}}).$$

A simple calculation shows that C_3 is the encryption of $m_3 = m_1 \cdot m_2$.

Proof. Suppose first that $R_{\mathsf{id}} \in \mathbb{QR}_N$. Then, letting $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$, we have:

$$\begin{aligned}\left(\frac{c_3+2r_{\mathsf{id}}}{N}\right) &= \left(\frac{\theta}{N}\right)\left(\frac{c_3+2r_{\mathsf{id}}}{N}\right) = \left(\frac{(t^2+R_{\mathsf{id}})D+4R_{\mathsf{id}}tU+2r_{\mathsf{id}}(tD+(t^2+R_{\mathsf{id}})U)}{N}\right)\\ &= \left(\frac{(t^2+R_{\mathsf{id}}+2r_{\mathsf{id}}t)D+(2r_{\mathsf{id}}t+t^2+R_{\mathsf{id}})(2r_{\mathsf{id}}U)}{N}\right) = \left(\frac{(t+r_{\mathsf{id}})^2(D+2r_{\mathsf{id}}U)}{N}\right)\\ &= \left(\frac{D+2r_{\mathsf{id}}U}{N}\right) = \left(\frac{(c_1c_2+4R_{\mathsf{id}})+2r_{\mathsf{id}}(c_1+c_2)}{N}\right)\\ &= \left(\frac{c_1+2r_{\mathsf{id}}}{N}\right)\left(\frac{c_2+2r_{\mathsf{id}}}{N}\right)\end{aligned}$$

as desired.

The case $R_{\mathsf{id}} \in \mathbb{J}_N \setminus \mathbb{QR}_N$ is similar. Letting $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2} \bmod N$, we then have $\left(\frac{\bar{c}_3+2r_{\mathsf{id}}}{N}\right) = \left(\frac{\bar{c}_1+2r_{\mathsf{id}}}{N}\right)\left(\frac{\bar{c}_2+2r_{\mathsf{id}}}{N}\right)$. □

Why Does It Work? At first sight, the **HOM** procedure may appear cumbersome. Actually it is not. Without loss of generality, assume that input $\Gamma \leftarrow R_{\mathsf{id}} = {r_{\mathsf{id}}}^2 \in \mathbb{QR}_N$. A straightforward application of the homomorphism induced by the underlying $\circledast$ law will not get the correct result. Clearly, if we call $c_3 = \mathbf{HOM}(c_1, c_2, R_{\mathsf{id}})$ and let

$$\frac{c_3'}{2} = \frac{c_1}{2} \circledast \frac{c_2}{2} \iff c_3' = \frac{c_1c_2 + 4R_{\mathsf{id}}}{c_1 + c_2}$$

then $\left(\frac{c_3+2r_{\mathsf{id}}}{N}\right) = \left(\frac{c_3'+2r_{\mathsf{id}}}{N}\right)$ if and only if $\left(\frac{c_1+c_2}{N}\right) = 1$. Indeed, the above definition of c_3' immediately yields

$$\left(\frac{c_3 + 2r_{\mathsf{id}}}{N}\right) := \left(\frac{c_1 + 2r_{\mathsf{id}}}{N}\right)\left(\frac{c_2 + 2r_{\mathsf{id}}}{N}\right) = \left(\frac{c_1 + c_2}{N}\right)\left(\frac{c_3' + 2r_{\mathsf{id}}}{N}\right).$$

In other words, c_3' is the encryption of $m_1 \cdot m_2$ if and only if $\left(\frac{c_1+c_2}{N}\right) = 1$. This problem is resolved by randomizing one of the input ciphertexts using the homomorphism. One way to achieve this is to replace ciphertext c_1 with an equivalent randomized ciphertext,

$$\frac{c_1}{2} \leftarrow \frac{c_1}{2} \circledast \frac{\mathbb{1}}{2}$$

until $\left(\frac{c_1+c_2}{N}\right) = 1$, where $\mathbb{1} = \mathtt{ENCRYPT}_{\mathsf{msk}}(\mathsf{id}, 1)$ is a random Cocks encryption (w.r.t. R_{id}) of message $m = 1$. Note that no secret is involved in the randomization of c_1. This is the design strategy behind the **HOM** procedure.

We have seen that the **HOM** procedure can be used to randomize ciphertexts. Likewise, it can be used to flip the value of a plaintext message $m_1 \in \{\pm 1\}$ corresponding to a given Cocks ciphertext $C_1 = (c_1, \bar{c}_1)$ by taking the encryption of $m_2 = -1$ for ciphertext $C_2 = (c_2, \bar{c}_2)$.

A variant of Cocks' scheme that makes easier the computation over ciphertexts can be found in Appendix C.

It is also worth noting that when the message space is $\{0,1\}$ rather than $\{\pm 1\}$ then the scheme is homomorphic with respect the XOR operator. Indeed, letting $b_1 = \mu^{-1}(m_1)$ and $b_2 = \mu^{-1}(m_2)$, we have $\mu^{-1}(m_1 \cdot m_2) = b_1 \oplus b_2$, where $\mu^{-1}(m_i) = (1 - m_i)/2$ —see Sect. 3. We so get $\mu(b_1 \oplus b_2) = (-1)^{b_1 \oplus b_2} = (-1)^{b_1 + b_2} = m_1 \cdot m_2$.

6 An Anonymous IBE Scheme

In numerous scenarios, the recipient's identity in a transmission needs to be kept *anonymous*. This allows users to maintain some privacy. Protecting communication content may be not enough, as already observed in, e.g., [3,4,21]. For example, by analyzing the traffic between an antenna and a mobile device, one can recover some information about [at least] user's position and some details about the use of her mobile device. This information leaks easily during all day: it is a common habit, indeed, to use a mobile phone every day and to keep it (almost) always switched on.

As pointed out by Galbraith (see [6, Sect. 4], Cocks' scheme is *not* anonymous. A detailed discussion on the so-called Galbraith's test can be found in [2, Sect. 2.3]). In the same paper, Ateniese and Gasti also show that Galbraith's is the "best test" possible against the anonymity of Cocks' scheme.

In this section, we rephrase Galbraith's test using our representation. We then build on an original technique developed in [11] to get anonymized Cocks ciphertexts. However, unlike [11], there is no ciphertext expansion. Anonymized Cocks ciphertexts have the same size as non-anonymized ciphertexts. The decryption algorithm is modified accordingly by first de-anonymizing the ciphertext and then applying the regular decryption process. The resulting scheme is shown to meet the ANO-IND-ID-CPA security notion under the QR assumption in the random oracle model or, equivalently, under the IQR assumption in the standard model (cf. Remark 2).

6.1 Making Cocks Ciphertexts Anonymous

As Eq. (6) indicates, the subset $\widetilde{\mathcal{Z}}_{N,\Delta} \subset \mathcal{Z}_{N,\Delta}$ can be defined as

$$\widetilde{\mathcal{Z}}_{N,\Delta} = \{u \in \mathbb{Z}/N\mathbb{Z} \mid \gcd(u^2 - \Delta, N) = 1\} \cup \{\infty\}$$

where $N = pq$ and $\Delta \in \mathbb{QR}_N$. The following subsets of $\widetilde{\mathcal{Z}}_{N,\Delta}$ will be useful:

- $\widehat{\mathcal{Z}}_{N,\Delta} := \{u \in \mathbb{Z}/N\mathbb{Z} \mid \gcd(u^2 - \Delta, N) = 1\} \subset \widetilde{\mathcal{Z}}_{N,\Delta}$;
- $\widehat{\mathcal{Z}}^{[-1]}_{N,\Delta} := \{u \in \mathbb{Z}/N\mathbb{Z} \mid \left(\frac{u^2-\Delta}{N}\right) = -1\} \subset \widehat{\mathcal{Z}}_{N,\Delta}$;
- $\widehat{\mathcal{Z}}^{[+1]}_{N,\Delta} := \{u \in \mathbb{Z}/N\mathbb{Z} \mid \left(\frac{u^2-\Delta}{N}\right) = 1\} \subset \widehat{\mathcal{Z}}_{N,\Delta}$;
- $(\widehat{\mathcal{Z}}_{N,\Delta})^2 := \{u \in \mathbb{Z}/N\mathbb{Z} \mid \left(\frac{u^2-\Delta}{p}\right) = \left(\frac{u^2-\Delta}{q}\right) = 1\} \subset \widehat{\mathcal{Z}}^{[+1]}_{N,\Delta}$.

We have

$$\widehat{\mathcal{Z}}_{N,\Delta} = \widetilde{\mathcal{Z}}_{N,\Delta} \setminus \{\infty\} = (\mathcal{F}_{p,\Delta} \setminus \{\infty_p\}) \times (\mathcal{F}_{q,\Delta} \setminus \{\infty_q\})$$

from Eq. (5). By definition (see Sect. 4.2), for $u \in \mathcal{F}_{p,\Delta} \setminus \{\infty_p\}$, we have $\psi(u) = \frac{u+\delta}{u-\delta}$, and $\psi(\infty_p) = 1$. Multiplying both sides by $(u-\delta)^2$, the latter identity yields

$$\psi(u) \cdot (u-\delta)^2 = u^2 - \Delta. \tag{3}$$

The group $(\mathcal{F}_{p,\Delta})^2 = \{u \circledast u \mid u \in \mathcal{F}_{p,\Delta}\}$ is the subgroup of squares in $\mathcal{F}_{p,\Delta}$. Therefore, if $u \in (\mathcal{F}_{p,\Delta})^2$ then $\psi(u)$ is a quadratic residue modulo p; i.e., $\left(\frac{\psi(u)}{p}\right) = 1$. Together with Eq. (3) it follows that $u^2 - \Delta$ is a quadratic residue modulo p when $u \in (\mathcal{F}_{p,\Delta})^2$, $u \neq \infty_p$. This gives an alternative definition for the group $(\mathcal{F}_{p,\Delta})^2$, namely $(\mathcal{F}_{p,\Delta})^2 = \left\{u \in \mathcal{F}_{p,\Delta} \mid \left(\frac{u^2-\Delta}{p}\right) = 1\right\} \cup \{\infty_p\} = \left\{u \in \mathbb{F}_p \mid \left(\frac{u^2-\Delta}{p}\right) = 1\right\} \cup \{\infty_p\}$. Hence, using Chinese remaindering, we get

$$(\widehat{\mathcal{Z}}_{N,\Delta})^2 = \big((\mathcal{F}_{p,\Delta})^2 \setminus \{\infty_p\}\big) \times \big((\mathcal{F}_{q,\Delta})^2 \setminus \{\infty_q\}\big).$$

This means that $(\widehat{\mathcal{Z}}_{N,\Delta})^2$ is an alternative representation for the subset $\mathcal{S}_{N,\Delta}$ of squares in $\widetilde{\mathcal{Z}}_{N,\Delta}$. Likewise, $\widehat{\mathcal{Z}}^{[-1]}_{N,\Delta}$ denotes the subset of elements that are squares modulo p and non-squares modulo q, or vice-versa; and $\widehat{\mathcal{Z}}^{[+1]}_{N,\Delta}$ denotes the subset of elements that are either both squares modulo p and modulo q, or both non-squares modulo p and modulo q.

We have shown that:

Proposition 2. *Let $N = pq$ be an RSA modulus and let $w \in \widehat{\mathcal{Z}}_{N,\Delta}$. If*

$$\left(\frac{w^2 - \Delta}{N}\right) = -1$$

then $w \notin \mathcal{S}_{N,\Delta}$. □

This is nothing but Galbraith's test. Back to the anonymity problem, letting $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$, it implies that elements of the form $\frac{c}{2} = \frac{t^2+R_{\mathsf{id}}}{2t} \bmod N$ (respectively, $\frac{\bar{c}}{2} = \frac{\bar{t}^2+uR_{\mathsf{id}}}{2\bar{t}} \bmod N$) —where $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$ is derived from some user's identity id— cannot be used as part of a ciphertext for user with identity id' because if $\left(\frac{(c/2)^2-R_{\mathsf{id}'}}{N}\right) = -1$ (respectively, if $\left(\frac{(\bar{c}/2)^2-uR_{\mathsf{id}'}}{N}\right) = -1$) —where $R_{\mathsf{id}'} = \mathcal{H}(\mathsf{id}')$ for some other identity id'— then one can conclude that the identity of the recipient of the ciphertext is <u>not</u> id'. This clearly violates the anonymity requirement.

This issue is easily solved by $\circledast$-multiplying with probability 1/2 the value of $\frac{c}{2}$ (resp. $\frac{\bar{c}}{2}$) by an element $\frac{d}{2}$ satisfying $\left(\frac{(d/2)^2-\Delta}{N}\right) = -1$ (resp. $\left(\frac{(d/2)^2-u\Delta}{N}\right) = -1$). The decryption algorithm, assuming it is the legitimate recipient of the ciphertext, can then $\circledast$-divide by $\frac{d}{2}$ the ciphertext in the case it were $\circledast$-multiplied by $\frac{d}{2}$; letting $\frac{e}{2}$ (resp. $\frac{\bar{e}}{2}$) the received part of the ciphertext, it is easy for the decryption algorithm to know if $\frac{e}{2} = \frac{c}{2}$ or $\frac{e}{2} = \frac{c}{2} \circledast \frac{d}{2}$ (resp. $\frac{\bar{e}}{2} = \frac{\bar{c}}{2}$ or $\frac{\bar{e}}{2} = \frac{\bar{c}}{2} \circledast \frac{d}{2}$) by checking if $\left(\frac{(e/2)^2-\Delta}{N}\right) = 1$ or -1 (resp. $\left(\frac{(\bar{e}/2)^2-u\Delta}{N}\right) = 1$ or -1), respectively.

6.2 Anonymous IBE Without Ciphertext Expansion

There is a variety of possible instantiations of the above methodology. An efficient implementation can be achieved by specializing hash function $\mathcal{H}$. Instead of considering a function mapping bit-strings to any element of $\mathbb{J}_N$, we require that, in addition, on input id, the output must satisfy the extra condition $\left(\frac{d^2-4R_{\mathsf{id}}}{N}\right) = \left(\frac{d^2-4uR_{\mathsf{id}}}{N}\right) = -1$ for some given d:

$$\mathcal{H}_d : \{0,1\}^* \to \mathbb{J}_N,\ \mathsf{id} \mapsto \mathcal{H}_d(\mathsf{id})\ \text{ s.t. } \left(\frac{d^2-4\mathcal{H}_d(\mathsf{id})}{N}\right) = \left(\frac{d^2-4u\mathcal{H}_d(\mathsf{id})}{N}\right) = -1. \quad (4)$$

Here is the resulting scheme.

$\texttt{SETUP}(1^\kappa)$**.** Given a security parameter κ, `SETUP` generates an RSA modulus $N = pq$ where p and q are prime. It also selects an element $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$ and a global integer d. The public system parameters are $\mathsf{mpk} = \{N, u, d, \mathcal{H}_d\}$ where $\mathcal{H}_d$ is a cryptographic hash function as per Eq. (4). The master secret key is $\mathsf{msk} = \{p, q\}$.

$\texttt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$**.** Given identity id, algorithm `EXTRACT` sets $R_{\mathsf{id}} = \mathcal{H}_d(\mathsf{id})$. Then if $R_{\mathsf{id}} \in \mathbb{QR}_N$ it computes $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$; otherwise it computes $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2} \bmod N$. `EXTRACT` returns user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

$\texttt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}, m)$**.** To encrypt a message $m \in \{\pm 1\}$ for a user with identity id, `ENCRYPT` defines $R_{\mathsf{id}} = \mathcal{H}_d(\mathsf{id})$. It chooses at random $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$ such that $\left(\frac{t}{N}\right) = \left(\frac{\bar{t}}{N}\right) = m$ and lets

$$c^{(0)} = t + \frac{R_{\mathsf{id}}}{t} \bmod N\,, \quad c^{(1)} = \frac{c^{(0)}d + 4R_{\mathsf{id}}}{c^{(0)} + d} \bmod N\,,$$
$$\bar{c}^{(0)} = \bar{t} + \frac{uR_{\mathsf{id}}}{\bar{t}} \bmod N\,, \quad \bar{c}^{(1)} = \frac{\bar{c}^{(0)}d + 4uR_{\mathsf{id}}}{\bar{c}^{(0)} + d} \bmod N.$$

It chooses random bits $\beta_1, \beta_2 \in \{0,1\}$ and sets $c = c^{(\beta_1)}$ and $\bar{c} = \bar{c}^{(\beta_2)}$. The returned ciphertext is $C = (c, \bar{c})$.

$\texttt{DECRYPT}_{\mathsf{usk}}(C)$ Let $R_{\mathsf{id}} = \mathcal{H}_d(\mathsf{id})$**.** From $\mathsf{usk} = \{r_{\mathsf{id}}\}$ and $C = (c, \bar{c})$, if ${r_{\mathsf{id}}}^2 \equiv R_{\mathsf{id}} (\bmod N)$, `DECRYPT` sets $\gamma = c$ and $\Delta = R_{\mathsf{id}}$; otherwise it sets $\gamma = \bar{c}$ and $\Delta = uR_{\mathsf{id}}$. Next, it computes $\sigma = \left(\frac{\gamma^2 - 4\Delta}{N}\right)$. Finally, it returns plaintext m as

$$m = \begin{cases} \left(\frac{\gamma + 2r_{\mathsf{id}}}{N}\right) & \text{if } \sigma = 1\,, \\ \left(\frac{(\gamma + 2r_{\mathsf{id}})(d - 2r_{\mathsf{id}})(d - \gamma)}{N}\right) & \text{if } \sigma = -1. \end{cases}$$

Remark 7. The choice $p \equiv -q \,(\bmod 4)$ (or equivalently $N \equiv 3 \,(\bmod 4)$) simplifies the setting. In this case, we know that $\left(\frac{-1}{p}\right) = -\left(\frac{-1}{q}\right)$ and therefore $\left(\frac{-1}{N}\right) = -1$. A nice observation is that $d = 0$ is a valid parameter when $N \equiv 3 \,(\bmod 4)$ since then $\left(\frac{d^2-4\mathcal{H}(\mathsf{id})}{N}\right) = \left(\frac{d^2-4u\mathcal{H}(\mathsf{id})}{N}\right) = \left(\frac{-1}{N}\right) = -1$ as desired. Any cryptographic hash function $\mathcal{H}$ mapping bit-strings to $\mathbb{J}_N$ can be used.

6.3 Security Analysis

The two next propositions assess the security of the scheme under the quadratic residuosity assumption.

Proposition 3. *The scheme of Sect. 6.2 is* IND-ID-CPA *under the quadratic residuosity assumption in the random oracle model.*

Proof. Assume there exists an IND-ID-CPA adversary $\mathcal{A}$ against the previous scheme (Sect. 6.2). We can then use $\mathcal{A}$ to break the semantic security of the generalized Cocks' scheme (Sect. 3.1), which in turn will contradict the quadratic residuosity assumption. This is readily verified by observing that if $C = (c, \bar{c})$ is a valid ciphertext for the scheme of Sect. 3.1 for some user i with identity id then $C' = (c', \bar{c}')$ is a valid ciphertext for the scheme of Sect. 6.2, where $c' = c$ with probability $1/2$ and $c' = \frac{cd+4R_{\mathsf{id}}}{c+d} \bmod N$ with probability $1/2$ and, likewise, $\bar{c}' = \bar{c}$ with probability $1/2$ and $\bar{c}' = \frac{\bar{c}d+4R_{\mathsf{id}}}{\bar{c}+d} \bmod N$ with probability $1/2$. □

Before proving that the scheme is anonymous, we need the following lemma.

Lemma 2. *Let* RSAgen *be a probabilistic algorithm which, given a security parameter κ, outputs primes p and q and their product $N = pq$. Let also δ be a random element in $(\mathbb{Z}/N\mathbb{Z})^\times$ and $\Delta = \delta^2 \bmod N$. Then, under the quadratic residuosity assumption,*

$$\left| Pr\big[\mathcal{D}(x, \Delta, N) = 1 \mid x \xleftarrow{R} (\widehat{\mathcal{Z}}_{N,\Delta})^2\big] - Pr\big[\mathcal{D}(x, \Delta, N) = 1 \mid x \xleftarrow{R} \widehat{\mathcal{Z}}^{[+1]}_{N,\Delta} \setminus (\widehat{\mathcal{Z}}_{N,\Delta})^2\big] \right|$$

is negligible for any probabilistic polynomial-time distinguisher $\mathcal{D}$; the probabilities are taken over the experiment of running $(N, p, q) \leftarrow \mathsf{RSAgen}(1^\kappa)$, sampling $\delta \xleftarrow{R} (\mathbb{Z}/N\mathbb{Z})^\times$, and choosing at random $x \in (\widehat{\mathcal{Z}}_{N,\Delta})^2$ and $x \in \widehat{\mathcal{Z}}^{[+1]}_{N,\Delta} \setminus (\widehat{\mathcal{Z}}_{N,\Delta})^2$.

Proof. As previously shown in Sect. 6.1, we have that $(\widehat{\mathcal{Z}}_{N,\Delta})^2 = \mathcal{S}_{N,\Delta}$ (i.e., the set of all valid components of Cocks ciphertexts). The lemma now follows as an immediate application of [2, Lemma 2]. □

Proposition 4. *The scheme Sect. 6.2 is* ANO-ID-CPA *under the quadratic residuosity assumption in the random oracle model.*

Proof. As mentioned in Sect. 2.2, since the scheme is already known to be IND-ID-CPA, it suffices to prove that the statistical distance between the two distributions

$$D_0 = \big\{(\mathsf{id}_0^*, \mathsf{id}_1^*, \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}_0^*, m)) \mid m \xleftarrow{R} \{\pm 1\}\big\}$$

and

$$D_1 = \big\{(\mathsf{id}_0^*, \mathsf{id}_1^*, \mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}_1^*, m)) \mid m \xleftarrow{R} \{\pm 1\}\big\}$$

is negligible. In our case, a ciphertext encrypted for identity id_b^* (with $b \in \{0, 1\}$) is of the form $C_b = (c_b, \bar{c}_b)$. From Lemma 1, only the valid component can help in

distinguishing D_0 from D_1. Without loss of generality, we assume that $\mathcal{H}(\mathsf{id}_0^*) = \mathcal{H}(\mathsf{id}_1^*) \in \mathbb{QR}_N$. Letting $\Delta_b = \mathcal{H}(\mathsf{id}_b^*)$, the valid component is then

$$c_b := c_b^{(0)} = t + \frac{\Delta_b}{t} \bmod N \quad \text{or} \quad c_b := c_b^{(1)} = \frac{c_b^{(0)} d + 4\Delta_b}{c_b^{(0)} + d} \bmod N$$

where $\left(\frac{t}{N}\right) = m$.

Omitting $\mathsf{id}_0^*, \mathsf{id}_1^*$ to ease the reading, the above criterion requires that the distributions $D_0 = \{c_0^{(\beta)} \mid \beta \xleftarrow{R} \{0,1\}\}$ and $D_1 = \{c_1^{(\beta)} \mid \beta \xleftarrow{R} \{0,1\}\}$ —or equivalently rescaling by a factor of two, that the distributions

$$D_0^* = \left\{ \tfrac{c_0^{(\beta)}}{2} \mid \beta \xleftarrow{R} \{0,1\} \right\} \quad \text{and} \quad D_1^* = \left\{ \tfrac{c_1^{(\beta)}}{2} \mid \beta \xleftarrow{R} \{0,1\} \right\}$$

must be indistinguishable with overwhelming probability. Using the $\circledast$ operator, the elements of D_b^* (for $b \in \{0,1\}$) are

$$\tfrac{c_b^{(\beta)}}{2} = \begin{cases} \frac{1}{2}\left(t + \frac{\Delta_b}{t}\right) = \frac{t^2+\Delta_b}{2t} = t \circledast t & \text{when } \beta = 0 \\ \frac{1}{2}\left(\frac{c_b^{(0)} d + 4\Delta_b}{c_b^{(0)} + d}\right) = \frac{\frac{c_b^{(0)}}{2}\frac{d}{2} + \Delta_b}{\frac{c_b^{(0)}}{2} + \frac{d}{2}} = \frac{c_b^{(0)}}{2} \circledast \frac{d}{2} = (t \circledast t) \circledast \frac{d}{2} & \text{when } \beta = 1 \end{cases}$$

where $t \xleftarrow{R} \{t \in (\mathbb{Z}/N\mathbb{Z})^\times \mid \gcd(t^2 - \Delta_b, N) = 1\}$. Hence, we can see that

$$D_b^* = \begin{cases} \left\{u \mid u \xleftarrow{R} (\widehat{\mathcal{Z}}_{N,\Delta_b})^2\right\} \overset{c}{\equiv} \left\{u \mid u \xleftarrow{R} \widehat{\mathcal{Z}}_{N,\Delta_b}^{[+1]}\right\} & \text{when } \beta = 0 \\ \left\{u \mid u \xleftarrow{R} \widehat{\mathcal{Z}}_{N,\Delta_b}^{[-1]}\right\} & \text{when } \beta = 1 \end{cases}.$$

The first assertion (when $\beta = 0$) follows from Lemma 2 (the notation $\overset{c}{\equiv}$ means computationally equivalent —under the QR assumption in this case). The second assertion (when $\beta = 1$) follows by noting that the Jacobi symbol $\left(\frac{\frac{d^2}{4} - \Delta_b}{N}\right) = -1$ and thus $\frac{d}{2} \in \widehat{\mathcal{Z}}_{N,\Delta_b}^{[-1]}$.

As a consequence, under the QR assumption, the distribution D_b^* appears indistinguishable from the uniform distribution over $\widehat{\mathcal{Z}}_{N,\Delta_b}^{[+1]} \cup \widehat{\mathcal{Z}}_{N,\Delta_b}^{[-1]} = \widehat{\mathcal{Z}}_{N,\Delta_b}$. This concludes the proof by noting that D_0^* and D_1^* are essentially the same sets: any random element is D_0^* is also an element in D_1^*, and vice-versa. □

Altogether, this proves that the scheme achieves ANO-IND-ID-CPA under the quadratic residuosity assumption in the random oracle model.

7 Conclusion

Somewhat surprisingly, we identified and detailed the algebraic group structure underlying Cocks encryption. The knowledge of this structure gives a better understanding of Cocks' scheme and allows one to see it differently. In particular, the hidden homomorphism opens the way to applications that were before not readily available or possible with Cocks' scheme, including homomorphic computations or anonymous encryption.

Acknowledgments. I am grateful to Michael Clear for sending a copy of [11] and to Dan Boneh for useful discussions.

A Proof of Proposition 1

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be adversary that can break the IND-ID-CPA security of the generalized scheme described in Sect. 3.1 with probability ϵ. We will use $\mathcal{A}$ to decide whether a random element w in $\mathbb{J}_N$ is quadratic residue modulo N or not.

A.1 First Case: u Is Universally Fixed

The first case assumes that system parameter u is a universally fixed parameter. It covers the original Cocks' scheme wherein $p, q \equiv 3 \pmod 4$ and $u = -1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$.

Let $\mathcal{H} : \{0,1\}^* \to \mathbb{J}_N$ be a hash function viewed as a random oracle. Consider the following distinguisher[2] $\mathcal{D}(w, u, N)$ for solving the QR problem. The goal of $\mathcal{D}$ is to distinguish a random element $w \in \mathbb{QR}_N$ from a random element $w \in \mathbb{J}_N \setminus \mathbb{QR}_N$.

1. Set $\mathsf{mpk} = \{N, u, \mathcal{H}\}$, and give mpk to $\mathcal{A}_1^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot),\mathcal{H}(\cdot)}$ —$\mathcal{A}_1$ has oracle access to $\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot)$ and $\mathcal{H}(\cdot)$, it may issue a number of extraction and hash queries, after what it selects a target identity id^*;
2. Depending on id^*:
   (a) If $\mathcal{H}(\mathsf{id}^*) = w$ then
      i. Choose a random bit $b \in \{0,1\}$, let $R_{\mathsf{id}^*} = \mathcal{H}(\mathsf{id}^*)$, and compute the encryption of $(-1)^b$ as $C_b = (c_b, \bar{c}_b)$ where
      $$c_b = t + \frac{R_{\mathsf{id}^*}}{t} \bmod N\,, \quad \bar{c}_b = \bar{t} + \frac{uR_{\mathsf{id}^*}}{\bar{t}} \bmod N\,,$$
      for some random elements $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$ such that $\left(\frac{t}{N}\right) = (-1)^b$ and $\left(\frac{\bar{t}}{N}\right) = (-1)^{1-b}$;
      ii. Give s and $C_b = (c_b, \bar{c}_b)$ to $\mathcal{A}_2^{\mathtt{EXTRACT}_{\mathsf{msk}}(\cdot),\mathcal{H}(\cdot)}$ —$\mathcal{A}_2$ may issue more extraction and hash queries, after what it returns its guess b';
      iii. If $b' = b$ return 1; otherwise return 0.
   (b) If $\mathcal{H}(\mathsf{id}^*) \neq w$ then
      i. Choose a random bit $b' \in \{0,1\}$;
      ii. Return b'.

It remains to detail how $\mathcal{D}$ simulates answers to oracle queries. $\mathcal{D}$ maintains a history list $\mathsf{Hist}[\mathcal{H}]$ composed of triplets. The list is initialized to $\emptyset$. It also maintains a counter k initialized to 0. Let q_{H_1} denote the number of hash queries that are not followed by extract queries and let q_{E_1} denote the number of extract

[2] Note that $\mathcal{D}$ is given $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$ as it is universally fixed.

queries, made by $\mathcal{A}_1$. Without loss of generality, we assume that $\mathcal{A}_1$ issues a hash query on id^*. Finally, we let k_1 denote a random integer in $\{1, \ldots, q_{H_1} + q_{E_1}\}$ chosen by $\mathcal{D}$.

Hash Queries. When $\mathcal{A}$ queries oracle $\mathcal{H}$ on some id, $\mathcal{D}$ checks whether there is an entry of the form (id, h, r) in $\mathsf{Hist}[\mathcal{H}]$; i.e., a triplet with id as the first component. If so, it returns h. Otherwise, it does the following:

1. Increment k;
2. Depending on k:
   (a) If $k = k_1$, define $h = w$ and append $(\mathsf{id}, h, \perp)$ to $\mathsf{Hist}[\mathcal{H}]$;
   (b) Else (if $k \neq k_1$), define $h = u^{-j} r^2 \bmod N$ with $r \xleftarrow{R} (\mathbb{Z}/N\mathbb{Z})^\times$ and $j \xleftarrow{R} \{0, 1\}$ and append (id, h, r) to $\mathsf{Hist}[\mathcal{H}]$;
3. Return h.

Extraction Queries. When $\mathcal{A}$ queries oracle `EXTRACT` on some id, $\mathcal{D}$ checks whether there is an entry of the form (id, h, r) in $\mathsf{Hist}[\mathcal{H}]$. If not, it calls $\mathcal{H}(\mathsf{id})$ so that there is an entry. Let (id, h, r) denote the entry in $\mathsf{Hist}[\mathcal{H}]$ corresponding to id. Depending on it, $\mathcal{D}$ does the following:

1. If $r \neq \perp$ then return r;
2. If $r = \perp$ then abort.

We now analyze the success probability of $\mathcal{D}$ in solving the QR challenge. Since u is an element in $\mathbb{J}_N \setminus \mathbb{QR}_N$, the resulting mpk appear as valid system parameters. Three subcases can be distinguished.

Subcase i. The first subcase supposes $w = \mathcal{H}(\mathsf{id}^*) \in \mathbb{QR}_N$. The condition $w = \mathcal{H}(\mathsf{id}^*)$ requires that id^* is the k_1-th query to $\mathcal{H}$. Further, since $\mathcal{H}(\mathsf{id}^*) \in \mathbb{QR}_N$, Lemma 1 teaches that $C_b = (c_b, \bar{c}_b)$ is a valid ciphertext for b. Namely, component c_b correctly decrypts to $(-1)^b$ and component $\bar{c}_b$ is of no use. Hence, $\mathcal{D}$ returns 1 exactly when $\mathcal{A}$ wins in the $\mathsf{IND\text{-}ID\text{-}CPA}$ game, provided that there is no abort. But since $\mathcal{A}$ is not allowed to submit id^* to `EXTRACT` (and so there is no abort when id^* is the k_1-th query), we get $\Pr[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{QR}_N \wedge w = \mathcal{H}(\mathsf{id}^*)] = \epsilon$.

Subcase ii. The second subcase supposes $w = \mathcal{H}(\mathsf{id}^*) \in \mathbb{J}_N \setminus \mathbb{QR}_N$. Since $\mathcal{H}(\mathsf{id}^*) \in \mathbb{J}_N \setminus \mathbb{QR}_N$, Lemma 1 teaches that $C_b = (c_b, \bar{c}_b)$ is a valid ciphertext for $(-1)^{(1-b)}$ —it is worth noticing that $\left(\frac{\bar{t}}{N}\right) = (-1)^{1-b}$. Hence, $\mathcal{D}$ returns 1 exactly when $\mathcal{A}$ looses in the $\mathsf{IND\text{-}ID\text{-}CPA}$ game. We therefore get $\Pr[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{J}_N \setminus \mathbb{QR}_N \wedge w = \mathcal{H}(\mathsf{id}^*)] = 1 - \epsilon$.

Subcase iii. The last subcase supposes $w \neq \mathcal{H}(\mathsf{id}^*)$. In this case $\mathcal{D}$ returns a random bit, regardless of w. Therefore, we have $\Pr[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{QR}_N \wedge w \neq \mathcal{H}(\mathsf{id}^*)] = \Pr[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{J}_N \setminus \mathbb{QR}_N \wedge w \neq \mathcal{H}(\mathsf{id}^*)] = 1/2$.

We so obtain:

$$\begin{aligned}
&\Pr\left[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{QR}_N\right] \\
&\quad = \Pr\left[w = \mathcal{H}(\mathsf{id}^*)\right] \cdot \Pr\left[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{QR}_N \wedge w = \mathcal{H}(\mathsf{id}^*)\right] \\
&\qquad + \Pr\left[w \neq \mathcal{H}(\mathsf{id}^*)\right] \cdot \Pr\left[\mathcal{D}(w, u, N) = 1 \mid w \in \mathbb{QR}_N \wedge w \neq \mathcal{H}(\mathsf{id}^*)\right] \\
&\quad = \frac{1}{q_{H_1}} \cdot \epsilon + \left(1 - \frac{1}{q_{H_1}}\right) \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon - \frac{1}{2}}{q_{H_1}}
\end{aligned}$$

and similarly,

$$\begin{aligned}\Pr\big[\mathcal{D}(w,u,N)=1 \mid w\in \mathbb{J}_N\setminus \mathbb{QR}_N\big] &= \frac{1}{q_{H_1}}\cdot(1-\epsilon)+\left(1-\frac{1}{q_{H_1}}\right)\cdot\frac{1}{2}\\ &= \frac{1}{2}+\frac{\frac{1}{2}-\epsilon}{q_{H_1}}.\end{aligned}$$

Putting all together, we get:

$$\left|\Pr\big[\mathcal{D}(w,u,N)=1 \mid w\in \mathbb{QR}_N\big]-\Pr\big[\mathcal{D}(w,u,N)=1 \mid w\in \mathbb{J}_N\setminus \mathbb{QR}_N\big]\right| = \frac{2}{q_{H_1}}\left|\epsilon-\frac{1}{2}\right|$$

which must be negligible by the QR assumption. As a consequence, $|\epsilon-\frac{1}{2}|$ must be negligible, which means that the scheme is IND-ID-CPA secure under the QR assumption.

A.2 Second Case: u Is Random

In this case, the proof can be obtained along the lines of the proof offered in [8, Appendix B.2] for the Boneh-Gentry-Hamburg scheme. The proof features a tight reduction. It however crucially requires that parameter u is defined as a random element in $\mathbb{J}_N\setminus\mathbb{QR}_N$.

B Arithmetic in $\mathcal{Z}_{N,\Delta}$

As mentioned in Sect. 4.2, each element u of the group $\mathcal{Z}_{N,\Delta}=\mathcal{F}_{p,\Delta}\times\mathcal{F}_{q,\Delta}$ can be uniquely represented by a pair $[u_p,u_q]$ with $u_p\in\mathcal{F}_{p,\Delta}$ and $u_q\in\mathcal{F}_{q,\Delta}$, and $\infty=[\infty_p,\infty_q]$. There is a slight complication when doing arithmetic in $\mathcal{Z}_{N,\Delta}$ as we need to deal with the elements of the form $[u_p,\infty_q]$ or $[\infty_p,u_q]$. This can be circumvented by adopting a projective representation. An element $u\in\mathcal{Z}_{N,\Delta}$ can be written as a pair $(U:Z)$. We say that two elements $u=(U:Z)$ and $u'=(U':Z')$ are equivalent if there exists some $\lambda\in(\mathbb{Z}/N\mathbb{Z})^\times$ such that $U'=\lambda U$ and $Z'=\lambda Z$. Hence, from the definition of ψ^{-1}, we can represent $\mathcal{Z}_{N,\Delta}$ as

$$\mathcal{Z}_{N,\Delta}=\big\{(\delta(v+1):v-1)\mid v\in(\mathbb{Z}/N\mathbb{Z})^\times\big\}.$$

The neutral element is $\infty=(1:0)$. The inverse of an element $(U:Z)$ is $(-U:Z)$. The product of two elements $(U_1:Z_1),(U_2,Z_2)\in\mathcal{Z}_{N,\Delta}$ is given by

$$(U_1:Z_1)\circledast(U_2,Z_2)=(U_1U_2+\Delta Z_1Z_2:U_1Z_2+U_2Z_1).$$

Observe that the group law is complete with the projective representation: it works for all inputs.

Another way to deal with the elements of the form $[u_p,\infty_q]$ or $[\infty_p,u_q]$ is simply to ignore them and to work in the subset

$$\widetilde{\mathcal{Z}}_{N,\Delta}=\big((\mathcal{F}_{p,\Delta}\setminus\{\infty_p\})\times(\mathcal{F}_{q,\Delta}\setminus\{\infty_q\})\big)\cup\{\infty\} \tag{5}$$

$$=\{u\in\mathbb{Z}/N\mathbb{Z}\mid \gcd(u^2-\Delta,N)=1\}\cup\{\infty\}. \tag{6}$$

wherever it is defined, the $\circledast$-law in $\widetilde{\mathcal{Z}}_{N,\Delta}$ coincides with the group law on $\mathcal{Z}_{N,\Delta}$:

$$u_1 \circledast u_2 = \begin{cases} \frac{u_1 u_2 + \Delta}{u_1 + u_2} \pmod N & \text{if } u_1 \neq -u_2 \\ \infty & \text{otherwise} \end{cases}.$$

(If $u_1 = \infty$ then $u_1 \circledast u_2 = u_2$; if $u_2 = \infty$ then $u_1 \circledast u_2 = u_1$.)

C Some Variants of Cocks' Scheme

The **HOM** is dependent of the cryptosystem. We propose below some variants of Cocks' scheme that leads to better efficiency. In particular, obtaining the encryption of the complementary value is almost free.

C.1 Basic Scheme

$\texttt{SETUP}(1^\kappa)$**.** Given a security parameter κ, `SETUP` generates an RSA modulus $N = pq$ where p and q are prime. It also selects an element $u \in \mathbb{J}_N \backslash \mathbb{QR}_N$. The public system parameters are $\mathsf{mpk} = \{N, u, \mathcal{H}\}$ where $\mathcal{H}$ is a cryptographic hash function mapping bit-strings to $\mathbb{J}_N$; i.e., $\mathcal{H} : \{0,1\}^* \to \mathbb{J}_N$. The master secret key is $\mathsf{msk} = \{p, q\}$.

$\texttt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$**.** Given identity id, key derivation algorithm `EXTRACT` sets $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. If $R_{\mathsf{id}} \in \mathbb{QR}_N$ it computes $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$; otherwise it computes $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2} \bmod N$. `EXTRACT` returns user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

$\texttt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}, m)$**.** To encrypt a message $m \in \{\pm 1\}$ for a user with identity id, `ENCRYPT` defines $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. It chooses at random $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$ and computes

$$\varepsilon = m \cdot \left(\frac{t}{N}\right), \quad c = t + \frac{R_{\mathsf{id}}}{t} \bmod N, \quad \bar{\varepsilon} = m \cdot \left(\frac{\bar{t}}{N}\right), \quad \bar{c} = \bar{t} + \frac{uR_{\mathsf{id}}}{\bar{t}} \bmod N.$$

The returned ciphertext is $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$.

$\texttt{DECRYPT}_{\mathsf{usk}}(C)$ From $\mathsf{usk} = \{r_{\mathsf{id}}\}$ and $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$, if ${r_{\mathsf{id}}}^2 \equiv \mathcal{H}(\mathsf{id}) \pmod N$, `DECRYPT` sets $\nu = \varepsilon$ and $\gamma = c$; otherwise it sets $\nu = \bar{\varepsilon}$ and $\gamma = \bar{c}$. Next it computes $\tau = \left(\frac{\gamma + 2r_{\mathsf{id}}}{N}\right)$ using secret key r_{id} and returns plaintext $m = \nu \cdot \tau$.

Homomorphic Computation. Let $C_1 = (\varepsilon_1, c_1, \bar{\varepsilon}_1, \bar{c}_1)$ and $C_2 = (\varepsilon_2, c_2, \bar{\varepsilon}_2, \bar{c}_2)$ be the respective encryption of messages m_1 and m_2 for a user with identity id. Then, letting $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$ and $\bar{R}_{\mathsf{id}} = u \cdot \mathcal{H}(\mathsf{id}) \bmod N$, we get that $C_3 = (\varepsilon_3, c_3, \bar{\varepsilon}_3, \bar{c}_3)$ with

$$\varepsilon_3 = \varepsilon_1 \cdot \varepsilon_2 \cdot \left(\frac{c_1 + c_2}{N}\right), \quad c_3 = \frac{c_1 c_2 + 4R_{\mathsf{id}}}{c_1 + c_2} \bmod N,$$
$$\bar{\varepsilon}_3 = \bar{\varepsilon}_1 \cdot \bar{\varepsilon}_2 \cdot \left(\frac{\bar{c}_1 + \bar{c}_2}{N}\right), \quad \text{and} \quad \bar{c}_3 = \frac{\bar{c}_1 \bar{c}_2 + 4\bar{R}_{\mathsf{id}}}{\bar{c}_1 + \bar{c}_2} \bmod N$$

is the encryption of message $m_3 = m_1 \cdot m_2$ (for the user with identity id).

Complementary Encryption. Given the encryption of a message $m \in \{\pm 1\}$, it is easy to get the encryption of the complementary value. If $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$ is the encryption of $m \in \{\pm 1\}$ then $C' = (-\varepsilon, c, -\bar{\varepsilon}, \bar{c})$ is the encryption of $-m$.

C.2 Compact Variant

As an illustration, suppose that $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id}) \in \mathbb{QR}_N$ in the previous scheme. If $C = (\varepsilon, c)$ with $\varepsilon = m \cdot \left(\frac{t}{N}\right)$ and $c = t + R_{\mathsf{id}}/t \bmod N$ is a valid encryption for message $m \in \{\pm 1\}$ then so is $C' := (\varepsilon', c')$ where $\varepsilon' = \varepsilon \cdot \left(\frac{-1}{N}\right)$ and $c' = N - c$. Indeed, letting $t' = -t \bmod N$, we have

$$c' \equiv -c \equiv -\left(t + \frac{R_i}{t}\right) \equiv t' + \frac{R_i}{t'} \,(\mathrm{mod}\, N) \quad \text{and}$$

$$\varepsilon' = m \cdot \left(\frac{t'}{N}\right) = m \cdot \left(\frac{-t}{N}\right) = \varepsilon \cdot \left(\frac{-1}{N}\right).$$

One of the two equivalent ciphertexts $C = (\varepsilon, c)$ and $C' = (\varepsilon', c')$ is (at least) one bit shorter than the other one.

As a result, if ℓ represents the bit-length of RSA modulus N this allows reducing the size a whole ciphertext to at most 2ℓ bits, as in Cocks' scheme.

$\mathtt{SETUP}(1^\kappa)$**.** Given a security parameter κ, `SETUP` generates an RSA modulus $N = pq$ where p and q are prime. It also selects an element $u \in \mathbb{J}_N \backslash \mathbb{QR}_N$. The public system parameters are $\mathsf{mpk} = \{N, u, \mathcal{H}\}$ where $\mathcal{H}$ is a cryptographic hash function mapping bit-strings to $\mathbb{J}_N$; i.e., $\mathcal{H} : \{0,1\}^* \to \mathbb{J}_N$. The master secret key is $\mathsf{msk} = \{p, q\}$.

$\mathtt{EXTRACT}_{\mathsf{msk}}(\mathsf{id})$**.** Given identity id, key derivation algorithm `EXTRACT` sets $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. If $R_{\mathsf{id}} \in \mathbb{QR}_N$ it computes $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$; otherwise it computes $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2} \bmod N$. `EXTRACT` returns user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

$\mathtt{ENCRYPT}_{\mathsf{mpk}}(\mathsf{id}, m)$**.** To encrypt a message $m \in \{\pm 1\}$ for a user with identity id, `ENCRYPT` defines $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. It chooses at random $t, \bar{t} \in \mathbb{Z}/N\mathbb{Z}$ and computes

$$\varepsilon' = m \cdot \left(\frac{t}{N}\right), \;\; c' = t + \frac{R_{\mathsf{id}}}{t} \bmod N, \;\; \bar{\varepsilon}' = m \cdot \left(\frac{\bar{t}}{N}\right), \;\; \bar{c}' = \bar{t} + \frac{uR_{\mathsf{id}}}{\bar{t}} \bmod N.$$

Define $c = \min(c', N - c')$ and $\bar{c} = \min(\bar{c}', N - \bar{c}')$. If $c = c'$ then define $\varepsilon = \varepsilon'$; otherwise define $\varepsilon = \varepsilon' \cdot \left(\frac{-1}{N}\right)$. Similarly, if $\bar{c} = \bar{c}'$ then define $\bar{\varepsilon} = \bar{\varepsilon}'$; otherwise define $\bar{\varepsilon} = \bar{\varepsilon}' \cdot \left(\frac{-1}{N}\right)$. The returned ciphertext is $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$.

$\mathtt{DECRYPT}_{\mathsf{usk}}(C)$ From $\mathsf{usk} = \{r_{\mathsf{id}}\}$ and $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$, if ${r_{\mathsf{id}}}^2 \equiv \mathcal{H}(\mathsf{id}) \,(\mathrm{mod}\, N)$, `DECRYPT` sets $\nu = \varepsilon$ and $\gamma = c$; otherwise it sets $\nu = \bar{\varepsilon}$ and $\gamma = \bar{c}$. Next it computes $\tau = \left(\frac{\gamma + 2r_{\mathsf{id}}}{N}\right)$ using secret key r_{id} and returns plaintext $m = \nu \cdot \tau$.

Remark 8. Assuming that primes p and q satisfy the extra condition $p \equiv q \pmod 4$ (for example if $N = pq$ is a Blum integer) —in which case $\left(\frac{-1}{N}\right) = 1$— the encryption algorithm can then form the ciphertext $C = (\varepsilon, c, \bar{\varepsilon}, \bar{c})$ more simply by defining $\varepsilon = m \cdot \left(\frac{t}{N}\right)$ and $\bar{\varepsilon} = m \cdot \left(\frac{\bar{t}}{N}\right)$.

D Public-Key Encryption with Keyword Search

A prominent application of anonymous IBE scheme resides in public-key encryption with keyword search (or PEKS) [6]. Basically, PEKS is a form of encryption that allows searching on data that is encrypted using a public-key system. A typical application is for an email gateway to test whether or not the keyword "urgent" is present in an email. The gateway then routes the email if it is the case. Of course the gateway should only learn whether the word "urgent" is present but nothing else about the email. In the email use-case, another practical application is to test the sender's name of the email and to route the emails accordingly. Further applications for PEKS can be found in [1,6]. Of particular interest is the concept of temporarily searchable encryption [1, Sect. 6].

D.1 Definition

A *public-key encryption with keyword search scheme* [7] is defined as a tuple of four algorithms $(\texttt{KEYGEN}, \texttt{PEKS}, \texttt{TRAPDOOR}, \texttt{TEST})$:

Key Generation. The key generation algorithm KEYGEN is a randomized algorithm that takes as input some security parameter 1^κ and outputs a matching pair $(\mathsf{upk}, \mathsf{usk})$ of public key and private key: $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \texttt{KEYGEN}(1^\kappa)$.

Public-Key Encryption With Keyword Search (PEKS). Let $\mathcal{W}$ denote the keyword space. The PEKS algorithm PEKS takes as input a public key upk and a keyword $w \in \mathcal{W}$, and returns a searchable ciphertext S. We write $S \leftarrow \texttt{PEKS}_{\mathsf{upk}}(w)$.

Trapdoor. The trapdoor algorithm TRAPDOOR takes as input the private key usk (corresponding to upk) and a keyword w, and returns a trapdoor T_w for keyword w. We write $T_w \leftarrow \texttt{TRAPDOOR}_{\mathsf{usk}}(w)$.

Test. The test algorithm TEST takes as input a searchable ciphertext S and a trapdoor T_w, and returns a bit b. A bit b with 1 means "accept" or "yes", and a bit b with 0 means "reject" or "no". We write $b \leftarrow \texttt{TEST}(S, T_w)$.

It is required that $\texttt{TEST}(\texttt{PEKS}_{\mathsf{upk}}(w), \texttt{TRAPDOOR}_{\mathsf{usk}}(w)) = 1$ for all keywords $w \in \mathcal{W}$.

D.2 Public-Key Encryption with Keyword Search from Quadratic Residuosity

In a PEKS scheme, a sender can send messages in encrypted form to a receiver so that the receiver can allow a designated proxy to search keywords in the encrypted messages without incurring any (additional) loss of privacy. In [6], Boneh *et al.* suggest the following methodology:

– The sender encrypts the message being sent with a (regular) public-key cryptosystem;
– She appends to the resulting ciphertext a PEKS for each keyword.

In more detail, to encrypt a message m with searchable keywords $w_1, \ldots, w_n$ for the receiver with public key upk, the sender computes and sends

$$c = \mathtt{ENCRYPT}_{\mathsf{upk}}(m)\,, \quad S_1 = \mathtt{PEKS}_{\mathsf{upk}}(w_1)\,, \quad \ldots\,, \quad S_n = \mathtt{PEKS}_{\mathsf{upk}}(w_n).$$

The whole ciphertext is $C = \{c, S_1, \ldots, S_n\}$. Now if the receiver has given a proxy a trapdoor T_{w_j} for keyword w_j then this proxy can test whether the corresponding plaintext m contains the keyword w_j, but nothing more.

A conversion to turn an anonymous identity-based scheme (under certain conditions) into a PEKS scheme is developed in [6]. Some subsequent refinements are described in [1]. Applied to the scheme of Sect. 6.2 as a building block, we so obtain a PEKS scheme based on the quadratic residuosity. For slightly better efficiency, instead of verifying whether $x_i = \mu^{-1}(\nu_i \cdot \tau_i)$ $(\in \{0,1\})$, for $0 \le i \le k-1$, the TEST algorithm equivalently verifies whether $\tau_i = \nu_i \cdot (1 - 2x_i)$. In detail, the scheme is as follows.

$\mathtt{KEYGEN}(1^\kappa)$**.** Given a security parameter κ, KEYGEN generates an RSA modulus $N = pq$ where p and q are prime. It defines a security parameter k depending on κ. It also selects an element $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$ and a global integer d. The user's public key is $\mathsf{upk} = \{N, k, u, d, \mathcal{H}_d\}$ where $\mathcal{H}_d$ is a cryptographic hash function mapping bit-strings to $\mathbb{J}_N$ as per Eq. (4). The user's private key is $\mathsf{usk} = \{p, q\}$.

$\mathtt{PEKS}_{\mathsf{upk}}(w)$**.** To encrypt a keyword $w \in \{0,1\}^*$, PEKS selects a k-bit integer $x = \sum_{i=0}^{k-1} x_i 2^i$ (with $x_i \in \{0,1\}$). It defines $R = \mathcal{H}_d(w)$.
For $i = 0, \ldots, k-1$, it does the following:
1. choose at random $t_i, \bar{t}_i \in \mathbb{Z}/N\mathbb{Z}$;
2. let

$$\varepsilon_i = (-1)^{x_i} \left(\frac{t_i}{N}\right), \quad c_i{}^{(0)} = t_i + \frac{R}{t_i} \bmod N\,, \quad c_i{}^{(1)} = \frac{c_i{}^{(0)} d + 4R}{c_i{}^{(0)} + d} \bmod N\,,$$

$$\bar{\varepsilon}_i = (-1)^{x_i} \left(\frac{\bar{t}_i}{N}\right), \quad \bar{c}_i^{(0)} = \bar{t}_i + \frac{uR}{\bar{t}_i} \bmod N\,, \quad \bar{c}_i^{(1)} = \frac{\bar{c}_i^{(0)} d + 4uR}{\bar{c}_i^{(0)} + d} \bmod N\,;$$

3. choose random bits $\beta_{1,i}, \beta_{2,i} \in \{0,1\}$ and set $c_i = c_i{}^{(\beta_{1,i})}$ and $\bar{c}_i = \bar{c}_i^{(\beta_{2,i})}$.
PEKS returns the searchable ciphertext $S = \{x, \varepsilon_0, c_0, \bar{\varepsilon}_0, \bar{c}_0, \ldots, \varepsilon_{k-1}, c_{k-1}, \bar{\varepsilon}_{k-1}, \bar{c}_{k-1}\}$.

$\mathtt{TRAPDOOR}_{\mathsf{usk}}(w)$ Given keyword w, trapdoor algorithm TRAPDOOR sets $R = \mathcal{H}(w)$. If $R \in \mathbb{QR}_N$ it computes $T_w = R^{1/2} \bmod \mathrm{N}$; otherwise it computes $T_w = (uR)^{1/2} \bmod N$. TRAPDOOR returns T_w.

$\mathtt{TEST}(S, T_w)$ For keyword w, TEST uses the trapdoor T_w. Let $R = \mathcal{H}_d(w)$. Given a searchable ciphertext $S = \{x, \varepsilon_0, c_0, \bar{\varepsilon}_0, \bar{c}_0, \ldots, \varepsilon_{k-1}, c_{k-1}, \bar{\varepsilon}_{k-1}, \bar{c}_{k-1}\}$, if ${T_w}^2 \equiv R \,(\mathrm{mod}\, N)$, TEST sets $\nu_i = \varepsilon_i$, $\gamma_i = c_i$ for $0 \le i \le k-1$, and $\Delta = R$; otherwise it sets $\nu_i = \bar{\varepsilon}_i$, $\gamma_i = \bar{c}_i$ for $0 \le i \le k-1$, and $\Delta = uR$.
Next, for $i = 0, \ldots, k-1$, it does the following:

1. set $\sigma_i = \left(\frac{{\gamma_i}^2 - 4\Delta}{N}\right)$;
2. set

$$\tau_i = \begin{cases} \left(\frac{\gamma_i + 2T_w}{N}\right) & \text{if } \sigma_i = 1 \\ \left(\frac{(\gamma_i + 2T_w)(d - 2T_w)(d - \gamma_i)}{N}\right) & \text{if } \sigma_i = -1 \end{cases};$$

3. set $b_i = 1$ if $\tau_i = \nu_i \cdot (1 - 2x_i)$; set $b_i = 0$ otherwise;

`TEST` returns 1 if and only if $b_i = 1$ for all $0 \leq i \leq k - 1$; and 0 otherwise.

E A Remark on Boneh-Gentry-Hamburg Abstract IBE System

Cocks' scheme was subsequently revisited by Boneh, Gentry, and Hamburg [8]. The advantage of their scheme resides in the length of the ciphertexts. While the encryption of an ℓ-bit message requires $2\ell \cdot \log_2 N$ bits with Cocks' scheme, ciphertext size in Boneh-Gentry-Hamburg scheme is about $\ell + \log_2 N$ bits.

This section simplifies the abstract IBE system with short ciphertexts as presented in [8, Sect. 3].

E.1 Description

`SETUP` and `EXTRACT` are similar to Cocks' scheme. `ENCRYPT` and `DECRYPT` require a deterministic algorithm $\mathcal{Q}$ taking as input an RSA modulus N and three elements $u, R, S \in \mathbb{Z}/N\mathbb{Z}$ and returning four *IBE-compatible polynomials* $f, g, \bar{f}, \tau \in \mathbb{Z}/N\mathbb{Z}[X]$. Polynomials $f, \bar{f}, g, \tau$ are said IBE-compatible if and only if the following conditions are met:

c1. If $R, S \in \mathbb{QR}_N$ then $f(r)g(s) \in \mathbb{QR}_N$ for all square roots r of R and s of S;
c2. If $R \in \mathbb{QR}_N$ then $f(r)f(-r)S \in \mathbb{QR}_N$ for all square roots r of R;
c3. If $uR, S \in \mathbb{QR}_N$ then $\bar{f}(\bar{r})g(s)\tau(s) \in \mathbb{QR}_N$ for all square roots $\bar{r}$ of uR and s of S;
c4. If $uR \in \mathbb{QR}_N$ then $\bar{f}(\bar{r})\bar{f}(-\bar{r})S \in \mathbb{QR}_N$ for all square roots $\bar{r}$ of uR;
c5. If $S \in \mathbb{QR}_N$ then $\tau(s)\tau(-s)u \in \mathbb{QR}_N$ for all square roots s of S;
c6. Polynomial τ is independent of R.

In more detail, the Boneh-Gentry-Hamburg scheme goes as follows.

`SETUP`(1^κ)**.** Given a security parameter κ, `SETUP` generates an RSA modulus $N = pq$ where p and q are prime. It also generates a random element $u \in \mathbb{J}_N \setminus \mathbb{QR}_N$. The public system parameters are $\{N, u, \mathcal{H}, \mathcal{Q}\}$ where $\mathcal{H}$ is a cryptographic hash function mapping bitstrings to $\mathbb{J}_N$. The master secret key is $\mathsf{msk} = \{p, q\}$.

`EXTRACT`$_{\mathsf{msk}}(\mathsf{id})$**.** Using hash function $\mathcal{H}$, `EXTRACT` sets $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$. If $R_{\mathsf{id}} \in \mathbb{QR}_N$ it computes $r_{\mathsf{id}} = {R_{\mathsf{id}}}^{1/2} \bmod N$; otherwise it computes $r_{\mathsf{id}} = (uR_{\mathsf{id}})^{1/2} \bmod N$. `EXTRACT` returns user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

ENCRYPT(id, m)**.** To encrypt a message $m \in \{\pm 1\}$ for user with identity id, ENCRYPT

- chooses at random $s \in \mathbb{Z}/N\mathbb{Z}$ and computes $S = s^2 \bmod N$;
- runs $\mathcal{Q}(N, u, R_{\mathsf{id}}, S)$ where $R_{\mathsf{id}} = \mathcal{H}(\mathsf{id})$ to obtain g and τ;
- computes $k = \left(\frac{\tau(s)}{N}\right)$;
- forms $w = m \cdot \left(\frac{g(s)}{N}\right)$;
- returns ciphertext $C = (S, k, w)$.

DECRYPT(usk, C)**.** To decrypt $C = (S, k, w)$, intended for user with identity id, DECRYPT runs $\mathcal{Q}(N, u, R_{\mathsf{id}}, S)$ to obtain f and $\bar{f}$. Plaintext m is then recovered as

$$m = \begin{cases} w \cdot \left(\frac{f(r_{\mathsf{id}})}{N}\right) & \text{if } {r_{\mathsf{id}}}^2 \equiv R_{\mathsf{id}} \pmod N \\ w \cdot k \cdot \left(\frac{\bar{f}(r_{\mathsf{id}})}{N}\right) & \text{otherwise} \end{cases},$$

using the user's private key $\mathsf{usk} = \{r_{\mathsf{id}}\}$.

The correctness of the decryption follows from conditions C1–C6 imposed to polynomials $f, \bar{f}, g, \tau$.

The encryption of an ℓ-bit message $m = (m_1, m_2, \ldots, m_\ell) \in \{\pm 1\}^\ell$ proceeds broadly in the same way except that the user's private key is now $\mathsf{usk} = \{r_{\mathsf{id},1}, r_{\mathsf{id},2}, \ldots, r_{\mathsf{id},\ell}\}$ where ${r_{\mathsf{id},j}}^2 \equiv R_{\mathsf{id},j} \pmod N$ if $R_{\mathsf{id},j} \in \mathbb{QR}_N$ and ${r_{\mathsf{id},j}}^2 \equiv u R_{\mathsf{id},j} \pmod N$ if $R_{\mathsf{id},j} \in \mathbb{J}_N \setminus \mathbb{QR}_N$, and where $R_{\mathsf{id},j} = \mathcal{H}(\mathsf{id}, j)$ for $1 \le j \le \ell$. In other words, identity id is hashed ℓ times so as to produce ℓ values $R_{\mathsf{id},j}$ for $1 \le j \le \ell$. Now, each pair $(S, R_{\mathsf{id},j})$ (with the *same* S) is used to encrypt one message bit m_j; namely, $w_j = m_j \cdot \left(\frac{g_j(s)}{N}\right)$ where g_j is obtained by running $\mathcal{Q}(N, u, R_{\mathsf{id},j}, S)$. By the last condition, polynomial τ is always the same for all values of $R_{\mathsf{id},j}$. The ciphertext corresponding to message $m = (m_1, m_2, \ldots, m_\ell)$ is therefore given by $C = \{S, k, (w_1, w_2, \ldots, w_\ell)\}$. Plaintext message m is recovered from C bit-by-bit using private key $\mathsf{usk} = \{r_{\mathsf{id},1}, r_{\mathsf{id},2}, \ldots, r_{\mathsf{id},\ell}\}$ as $m_j = w_j \cdot \left(\frac{f(r_{\mathsf{id},j})}{N}\right)$ if ${r_{\mathsf{id},j}}^2 \equiv R_{\mathsf{id},j} \pmod N$ and as $m_j = w_j \cdot k \cdot \left(\frac{\bar{f}(r_{\mathsf{id},j})}{N}\right)$ otherwise, for $1 \le j \le \ell$.

E.2 A Simplified Abstract IBE

As described in the previous section, the abstract Boneh-Gentry-Hamburg system makes use of polynomials $f, \bar{f}, g, \tau \in \mathbb{Z}/N\mathbb{Z}[X]$. To simplify the notation, we consider one-bit messages but the discussion readily extends to ℓ-bit messages, $\ell > 1$.

We observe that polynomials f and $\bar{f}$ are evaluated at r_{id} and that polynomials g and τ are evaluated at s. Furthermore, we note that the values of R_{id}

and of S are publicly known. So, letting δ denote the degree of polynomial f and $f(X) = \sum_{k=0}^{\delta} f_k X^k$ with $f_k \in \mathbb{Z}/N\mathbb{Z}$, we can write

$$f(r_{\mathsf{id}}) = \sum_{k=0}^{\delta} f_k {r_{\mathsf{id}}}^k = A\, r_{\mathsf{id}} + B \pmod N \tag{7}$$

where

$$A = \sum_{\substack{0\le k\le \delta \\ k \text{ odd}}} f_k {r_{\mathsf{id}}}^{(k-1)} = \begin{cases} \displaystyle\sum_{\substack{0\le k\le \delta \\ k \text{ odd}}} f_k {R_{\mathsf{id}}}^{(k-1)/2} \pmod N & \text{if } {r_{\mathsf{id}}}^2 \equiv R_{\mathsf{id}} \pmod N \\ \displaystyle\sum_{\substack{0\le k\le \delta \\ k \text{ odd}}} f_k (uR_{\mathsf{id}})^{(k-1)/2} \pmod N & \text{otherwise} \end{cases}$$

and

$$B = \sum_{\substack{0\le k\le \delta \\ k \text{ even}}} f_k {r_{\mathsf{id}}}^k = \begin{cases} \displaystyle\sum_{\substack{0\le k\le \delta \\ k \text{ even}}} f_k {R_{\mathsf{id}}}^{k/2} \pmod N & \text{if } {r_{\mathsf{id}}}^2 \equiv R_{\mathsf{id}} \pmod N \\ \displaystyle\sum_{\substack{0\le k\le \delta \\ k \text{ even}}} f_k (uR_{\mathsf{id}})^{k/2} \pmod N & \text{otherwise} \end{cases}.$$

There is therefore no loss of generality to consider *degree-1* polynomials for f. The same conclusion holds for polynomials $\bar{f}$ (evaluated at r_{id}), and for polynomials g and τ (evaluated at s).

As a result, we define $f(X) = f_1 X + f_0$, $\bar{f}(X) = \bar{f}_1 X + \bar{f}_0$, $g(X) = g_1 X + g_0$, and $\tau(X) = \tau_1 X + \tau_0$. These four polynomials returned by public algorithm $\mathcal{Q}$ must be IBE-compatible.

For example, given R_{id} and S, one can select parameters $f_0, f_1, g_0, g_1 \in \mathbb{Z}/N\mathbb{Z}$ such that

$$2f_0g_0 \in \mathbb{QR}(N) \quad \text{and} \quad R_{\mathsf{id}} \left(\tfrac{f_1}{f_0}\right)^2 + S \left(\tfrac{g_1}{g_0}\right)^2 = 1 \pmod N. \tag{8}$$

This ensures that compatibility conditions C1 and C2 are satisfied.

Proof. Multiplying the second equation through ${f_0}^2$ yields $S{f_0}^2(\frac{g_1}{g_0})^2 = {f_0}^2 - R_{\mathsf{id}}{f_1}^2 = f(r_{\mathsf{id}})f(-r_{\mathsf{id}})$ for any square root r_{id} of R_{id}. Consequently, we have $f(r_{\mathsf{id}})f(-r_{\mathsf{id}})S = \left(Sf_0 \frac{g_1}{g_0}\right)^2 \in \mathbb{QR}(N)$. We also have $\left(r_{\mathsf{id}} \frac{f_1}{f_0} + s \frac{g_1}{g_0} + 1\right)^2 = R_{\mathsf{id}} \left(\frac{f_1}{f_0}\right)^2 + S \left(\frac{g_1}{g_0}\right)^2 + 1 + 2r_{\mathsf{id}} s \frac{f_1}{f_0}\frac{g_1}{g_0} + 2r_{\mathsf{id}} \frac{f_1}{f_0} + 2s \frac{g_1}{g_0} = \frac{2}{f_0g_0}(f_0g_0 + r_{\mathsf{id}} s f_1 g_1 + r_{\mathsf{id}} f_1 g_0 + s g_1 f_0) = \frac{2}{f_0g_0} f(r_{\mathsf{id}})g(s)$ for any square root r_{id} of R_{id} and any square root s of S. Since $2f_0g_0 \in \mathbb{QR}(N)$, it thus follows that $f(r_{\mathsf{id}})g(s) = \frac{2f_0g_0}{4}\left(r_{\mathsf{id}} \frac{f_1}{f_0} + s \frac{g_1}{g_0} + 1\right)^2 \in \mathbb{QR}(N)$, as required. □

We also define polynomial $\bar{g} \in \mathbb{Z}/N\mathbb{Z}[X]$ given by $\bar{g}(X) = \bar{g}_1 X + \bar{g}_0$ where $\bar{g}_1 = g_1\tau_0 + g_0\tau_1 \bmod N$ and $\bar{g}_0 = g_1\tau_1 S + g_0\tau_0 \bmod N$. It is worth noticing that evaluated at s we have $\bar{g}(s) \equiv g(s)\tau(s) \pmod N$. Analogously, we require

$$2\bar{f}_0\bar{g}_0 \in \mathbb{QR}(N) \quad \text{and} \quad uR_{\mathsf{id}} \left(\tfrac{\bar{f}_1}{\bar{f}_0}\right)^2 + S \left(\tfrac{\bar{g}_1}{\bar{g}_0}\right)^2 = 1 \pmod N \tag{9}$$

so as to fulfill compatibility conditions C3 and C4. Compatibility conditions C5 and C6 are automatically satisfied from the product formula in [8, Lemma 5.1]. If (f_0, f_1, g_0, g_1) is a solution to Eq. (8) and if (α, β) is a solution to $u\alpha^2 + S\beta^2 = 1$ then $(\bar{f}_0, \bar{f}_1, \bar{g}_0, \bar{g}_1)$ is a solution to Eq. (9) provided that

$$\frac{\bar{f}_1}{\bar{f}_0} = \frac{\frac{f_1}{f_0}\alpha}{S\frac{g_1}{g_0}\beta + 1} \pmod N \quad \text{and} \quad \frac{\bar{g}_1}{\bar{g}_0} \stackrel{\text{def}}{\equiv} \frac{g_1\tau_0 + g_0\tau_1}{g_1\tau_1 S + g_0\tau_0} \equiv \frac{\frac{g_1}{g_0} + \beta}{S\frac{g_1}{g_0}\beta + 1} \pmod N \tag{10}$$

with $2\bar{f}_0\bar{g}_0 \in \mathbb{QR}(N)$.

The instantiation presented in [8, Sect. 4] corresponds to the choice $f_0 = 1$, $f_1 = x$, $g_0 = 2$, $g_1 = 2y$, $\bar{f}_0 = Sy\beta + 1$, $\bar{f}_1 = x\alpha$, $\tau_0 = 1$ and $\tau_1 = \beta$, for some (x, y) satisfying $R_{\mathsf{id}}x^2 + Sy^2 = 1 \pmod N$ and (α, β) satisfying $u\alpha^2 + S\beta^2 = 1 \pmod N$.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptology **21**(3), 350–391 (2008)
2. Ateniese, G., Gasti, P.: Universally anonymous IBE based on the quadratic residuosity assumption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 32–47. Springer, Heidelberg (2009)
3. Barth, A., Boneh, D., Waters, B.: Privacy in encrypted content distribution using private broadcast encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 52–64. Springer, Heidelberg (2006)
4. 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)
5. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press (1993)
6. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
8. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 647–657. IEEE Computer Society (2007)
9. Boneh, D., LaVigne, R., Sabin, M.: Identity-based encryption with e^{th} residuosity and its incompressibility. In: Autumn TRUST Conference, Washington DC, 9–10 October 2013, poster presentation
10. Clear, M., Hughes, A., Tewari, H.: Homomorphic encryption with access policies: characterization and new constructions. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 61–87. Springer, Heidelberg (2013)

11. Clear, M., Tewari, H., McGoldrick, C.: Anonymous IBE from quadratic residuosity with improved performance. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT. LNCS, vol. 8469, pp. 377–397. Springer, Heidelberg (2014)
12. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
13. Desmedt, Y.G., Quisquater, J.-J.: Public-key systems based on the difficulty of tampering. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 111–117. Springer, Heidelberg (1987)
14. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on Jacobi symbols. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)
15. van Dijk, M., Woodruff, D.P.: Asymptotically optimal communication for torus-based cryptography. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 157–178. Springer, Heidelberg (2004)
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors from hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) Proceedings of the 40th Annua ACM Symposium on Theory of Computing (STOC 2008). pp. 197–206. ACM Press (2008)
17. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
18. Halevi, S.: A sufficient condition for key-privacy. IACR Cryptology ePrint Archive, Report 2005/005 (2005)
19. Jhanwar, M.P., Barua, R.: A variant of Boneh-Gentry-Hamburg's pairing-free identity based encryption scheme. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 314–331. Springer, Heidelberg (2009)
20. Joye, M., Neven, G. (eds.): Identity-Based Cryptography, Cryptology and Information Security Series, vol. 2. IOS Press, Amsterdam (2009)
21. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)
22. Rubin, K., Silverberg, A.: Torus-based cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 349–365. Springer, Heidelberg (2003)
23. Rubin, K., Silverberg, A.: Compression in finite fields and torus-based cryptography. SIAM J. Comput. **37**(5), 1401–1428 (2008)
24. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. IACR Cryptology ePrint Archive, Report 2003/054 (2003)
25. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

Identity-Based Hierarchical Key-Insulated Encryption Without Random Oracles

Yohei Watanabe[1,3(✉)] and Junji Shikata[1,2]

[1] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Japan
watanabe-yohei-xs@ynu.jp
[2] Institute of Advanced Sciences, Yokohama National University, Yokohama, Japan
shikata@ynu.ac.jp
[3] Information Technology Research Institute, AIST, Tokyo, Japan

Abstract. Key-insulated encryption is one of the effective solutions to a key exposure problem. Recently, identity-based encryption (IBE) has been used as one of fundamental cryptographic primitives in a wide range of various applications, and it is considered that the identity-based key-insulated security has a huge influence on the resulting applications. At Asiacrypt'05, Hanaoka et al. proposed an identity-based hierarchical key-insulated encryption (hierarchical IKE) scheme. Although their scheme is secure in the random oracle model, it has a "hierarchical key-updating structure," which is attractive functionality that enhances key exposure resistance.

In this paper, we first propose the hierarchical IKE scheme without random oracles. Our hierarchical IKE scheme is secure under the symmetric external Diffie–Hellman (SXDH) assumption, which is known as the simple and static one. Furthermore, when the hierarchy depth is one (i.e. not hierarchical case), our scheme is the first IKE scheme that achieves constant-size parameters including public parameters, secret keys, and ciphertexts.

Keywords: Key-insulated encryption · Identity-based hierarchical key-insulated encryption · Hierarchical identity-based encryption · Asymmetric pairing

1 Introduction

1.1 Background

A key exposure problem is unavoidable since human errors cannot seem to be eliminated in the future, and many researchers have tackled this problem in modern cryptography area so far. *Key-insulation*, which is introduced by Dodis et al. [12], is one solution to this problem. Specifically, they proposed public key encryption with the key-insulated property, which is called *public-key-based*

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 255–279, 2016.
DOI: 10.1007/978-3-662-49384-7_10

key-insulated encryption (PK-KIE). In PK-KIE, a user has two kinds of secret keys, so-called *a decryption key* and *a helper key*. The decryption key is used for decrypting ciphertexts and assumed to be stored in a powerful but insecure device such as laptops and smartphones. Meanwhile, the helper key is used for updating the decryption key and assumed to be stored in a physically-secure but computationally-limited device such as USB pen drives. Traditionally, in key-insulated cryptography, the following two kinds of security notions are considered:

1. If a number of decryption keys are exposed, the fact does not affect decryption keys at other time-periods.
2. Even if a helper key is exposed, the security is not compromised unless at least one decryption key is exposed.

We say a key-insulated system is secure if it satisfies 1; and it is *strongly* secure if it satisfies both 1 and 2. Specifically, the lifetime of the system is divided into discrete time-periods, and the user can decrypt the ciphertext encrypted at some time-period t by using a decryption key updated at the same time-period t. Therefore, even if the decryption key at t is exposed, the fact does not affect decryption keys at other time-periods, and hence the impact of the exposure can be significantly reduced.

Following a seminal work by Dodis et al. [12], symmetric-key-based key-insulated encryption [14], key-insulated signatures [13], and parallel key-insulated encryption [17,18,23] have been proposed so far. In addition to key-insulated cryptography, researchers have tackled the key exposure problem in various flavors. In forward-secure cryptography [1,8], users update their own secret keys at the beginning of each time-period. Even if the secret key is exposed, an adversary cannot get any information of ciphertexts encrypted at previous time-periods. Intrusion-resilient cryptography [10,11,20] realizes both key-insulated security and forward security simultaneously at the sacrifice of efficiency and practicality.

In this paper, we focus on the key-insulation paradigm in the identity-based setting. Since identity-based encryption (IBE) has been used as one of fundamental cryptographic primitives in a wide range of various applications, we believe that the identity-based key-insulated security has a huge influence on the resulting applications. Also, developing key-insulated cryptography in the identity-based area is the first step to consider the key-insulated security in the attribute-based [3,26] and functional encryption [7] settings. Thus, we consider that it is important to consider the identity-based key-insulated security. However, in the IBE context, there are only few researches on key-insulation. Hanaoka et al. [19] proposed the first identity-based (hierarchical) key-insulated encryption (IKE) scheme in the random oracle model. In their hierarchical IKE scheme, the key-updating mechanism has the hierarchical structure (and the scheme does not have a delegating property). Namely, not only a decryption key but also a helper key can be updated by a higher-level helper key. Since this "hierarchy" is not the same as that of hierarchical IBE (HIBE) [16], only applying techniques used in the HIBE context is insufficient for constructing secure

(in particular, *strongly* secure) IKE schemes. The hierarchical property is attractive since it enhances resistance to key exposure and there seem to be various applications due to progress in information technology (e.g., the popularization of smartphones). Let us consider an example: Suppose that each employee has a smartphone for business use, a laptop, and a PC at his office. A decryption key is stored in the smartphone, and it is updated by a 1-st level helper key stored in his laptop every day. However, the 1-st level helper key might be leaked since he carries around the laptop, and connects to the Internet via the laptop. Thus, the 1-st level helper key is also updated by a 2-nd level helper key stored in his PC every two–three months. Since the PC is not completely isolated from the Internet, every half a year, his boss updates the 2-nd level helper key is updated by 3-rd level helper key stored in an isolated private device. Thus, we believe hierarchical IKE has many potential applications.

After the proposal of hierarchical IBE by Hanaoka et al., two (not hierarchical) IKE schemes with additional properties in the standard model were proposed. One is the so-called *parallel* IKE scheme, which was proposed by Weng et al. [31]. The other is the so-called *threshold* IKE scheme, which was proposed by Weng et al. [32]. These two schemes enhance the resistance to helper key exposure by splitting a helper key into multiple ones. However, once the (divided) helper key is leaked, the security cannot be recovered. We now emphasize that the hierarchical key-updating structure is useful since even if some helper key is exposed, the helper key can be updated. However, there have been no hierarchical IKE schemes without random oracles so far.

1.2 Our Contribution

In this paper, our aim is to construct a hierarchical IKE scheme such that: (1) we can prove the security in the standard model from simple computational assumptions; and (2) when the hierarchy depth is one (i.e., not hierarchical case), the scheme achieves all constant-size parameters including public parameters, secret keys, and ciphertexts.

As a result, we propose the first hierarchical IKE scheme in the standard model. Specifically, we construct the hierarchical IKE scheme from the symmetric external Diffie–Hellman (SXDH) assumption, which is a static and simple one. Further, the proposed scheme achieves the constant-size parameters when the hierarchy is one, whereas public parameters of the (not hierarchical) existing scheme [32] depend on sizes of identity spaces (also see Sect. 4.1 for comparison). This is due to differences of base IBE schemes of each scheme. Our (hierarchical) IKE scheme is based on the Jutla–Roy IBE [22] and its variant [25], whereas the existing scheme (but not hierarchical one) [32] is based on the Waters IBE [29]. In the following, we explain why a naive solution is insufficient and why achieving (1) and (2) is challenging.

Why a (Trivial) Hierarchical IKE Scheme from HIBE is Insufficient.[1] One may think that a hierarchical IKE scheme can be easily obtained from an arbitrary HIBE scheme. However, the resulting IKE scheme is insecure in our security model, which was first formalized in [19]. The reason for this is that our security model includes the *strong* security model, and hence the fact makes a hierarchical IKE scheme from HIBE insecure. More specifically, a trivial construction is as follows. Let $sk_{\mathtt{I}}$ be a secret key for some identity $\mathtt{I}$ in HIBE, and $hk_{\mathtt{I}}^{(\ell)}$ be an ℓ-th level helper key for $\mathtt{I}$ in IKE. We set $sk_{\mathtt{I}}$ as $hk_{\mathtt{I}}^{(\ell)}$, and lower-level helper and decryption keys can be obtained from $sk_{\mathtt{I}}$ by regarding time-periods as descendants' identity. However, it is easy to see that if ℓ-th level helper key is exposed, then an adversary can obtain all lower-level keys, and thus, the resulting scheme does not meet the strong security. In fact, Bellare and Palacio [2] showed that *not strongly secure* PK-KIE is equivalent to IBE for a similar reason.

Difficulties in Constructing a Constant-Size IKE Scheme from Simple Computational Assumptions. The main difficulty in constructing an IKE scheme is that an adversary can get various keys regarding a target identity $\mathtt{I}^*$, whereas in (H)IBE, the adversary cannot get any information on a secret key for $\mathtt{I}^*$. This point makes a construction methodology non-trivial. Actually, it seems difficult to apply the Waters dual-system IBE [30] (and its variant [24]) as the underlying basis of IKE schemes. Technically, in their scheme each of secret keys and ciphertexts contains some random exponent, so-called tag_K and tag_C, respectively. In their proof, these tags for some $\mathtt{I}$ are needed to be generated by inputting $\mathtt{I}$ into some pairwise independent function, which is embedded into public parameters in advance. This generating procedure is necessary for cancellation of values and hence the security proof. Although it holds $tag_K = tag_C$ for the same identity $\mathtt{I}$, the proof works well since it is enough to generate only tag_K for all identities $\mathtt{I} \neq \mathtt{I}^*$ and only tag_C for the target identity $\mathtt{I}^*$. However, in the IKE setting, not only tag_C but also tag_K for $\mathtt{I}^*$ have to be generated since an adversary can get leaked decryption and helper keys for $\mathtt{I}^*$, and hence, the proof does not go well. To overcome this challenging point, we set (the variant of) the Jutla–Roy IBE [22,25], which is another type of constant-size IBE schemes, as the basis of our IKE scheme, and thus we can realize the first constant-size IKE scheme under the SXDH assumption. Further, we can also obtain the hierarchical IKE scheme by extending the technique into the hierarchical setting.

Organization of This Paper. In Sect. 2, we describe the notation used in this paper, asymmetric pairings, complexity assumptions, and functions which map time to discrete time-periods. In Sect. 3, we give a model and security definition of hierarchical IKE. In Sect. 4, we propose a direct construction of our hierarchical IKE scheme, and give the efficiency comparison among our scheme and existing schemes. In Sect. 5, we show the security proof of our scheme. In Sect. 6, we show a CCA-secure hierarchical IKE scheme. In Sect. 7, we conclude this paper.

[1] This fact was also mentioned in [19].

2 Preliminaries

Notation. In this paper, "probabilistic polynomial-time" is abbreviated as "PPT". Let $\mathbb{Z}_p := \{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^\times := \mathbb{Z}_p \backslash \{0\}$. If we write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}(x_1, x_2, \ldots, x_n)$ for an algorithm $\mathcal{A}$ having n inputs and m outputs, it means to input $x_1, x_2, \ldots, x_n$ into $\mathcal{A}$ and to get the resulting output $y_1, y_2, \ldots, y_m$. We write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}^{\mathcal{O}}(x_1, x_2, \ldots, x_n)$ to indicate that an algorithm $\mathcal{A}$ that is allowed to access an oracle $\mathcal{O}$ takes $x_1, x_2, \ldots, x_n$ as input and outputs $(y_1, y_2, \ldots, y_m)$. If $\mathcal{X}$ is a set, we write $x \stackrel{\$}{\leftarrow} \mathcal{X}$ to mean the operation of picking an element x of $\mathcal{X}$ uniformly at random. We use λ as a security parameter. $\mathcal{M}$ and $\mathcal{I}$ denote sets of plaintexts and IDs, respectively, which are determined by a security parameter λ.

Bilinear Group. A bilinear group generator $\mathcal{G}$ is an algorithm that takes a security parameter λ as input and outputs a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where p is a prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are multiplicative cyclic groups of order p, g_1 and g_2 are (random) generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and e is an efficiently computable and non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following bilinear property: For any $u, u' \in \mathbb{G}_1$ and $v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$, and for any $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ and any $a \in \mathbb{Z}_p$, $e(u^a, v) = e(u, v^a) = e(u, v)^a$.

A bilinear map e is called symmetric or a "Type-1" pairing if $\mathbb{G}_1 = \mathbb{G}_2$. Otherwise, it is called asymmetric. In the asymmetric setting, e is called a "Type-2" pairing if there is an efficiently computable isomorphism either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. If no efficiently computable isomorphisms are known, then it is called a "Type-3" pairing. In this paper, we focus on the Type-3 pairing, which is the most efficient setting (For details, see [9,15]).

Symmetric External Diffie–Hellman (SXDH) Assumption. We give the definition of the decisional Diffie–Hellman (DDH) assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, which are called the DDH1 and DDH2 assumptions, respectively.

Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH1 problem as follows.

$$Adv_{\mathcal{G},\mathcal{A}}^{DDH1}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}, \\ c_1, c_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ b \stackrel{\$}{\leftarrow} \{0,1\}, \\ \text{if } b = 0 \text{ then } T := g_1^{c_1 c_2}, \\ \text{else } T \stackrel{\$}{\leftarrow} \mathbb{G}_1, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1, g_2, g_1^{c_1}, g_1^{c_2}, T) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 1 (DDH1 Assumption). *The DDH1 assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv_{\mathcal{G},\mathcal{A}}^{DDH1}(\lambda)$ is negligible in λ.*

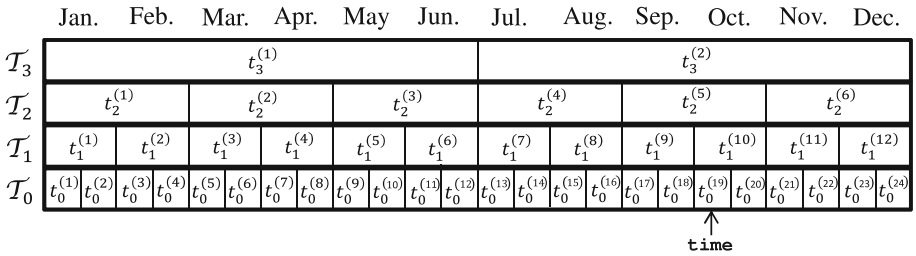


Fig. 1. Intuition of time-period map functions.

Similarly, we define the DDH2 problem. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH2 problem as follows.

$$Adv^{DDH2}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr \left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}, \\ c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } T := g_2^{c_1 c_2}, \\ \text{else } T \xleftarrow{\$} \mathbb{G}_2, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1, g_2, g_2^{c_1}, g_2^{c_2}, T) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 2 (DDH2 Assumption). *The DDH2 assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv^{DDH2}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.*

Definition 3 (SXDH Assumption). *We say that the SXDH assumption relative to a generator $\mathcal{G}$ holds if both the DDH1 and DDH2 assumptions relative to $\mathcal{G}$ hold.*

Time-Period Map Functions. In this paper, we deal with *several kinds of time-periods* since we consider that update intervals of each level key are different. For example, in some practical applications, it might be suitable that a decryption key (i.e. 0-th level key) and a 1-st level helper key should be updated every day and every three months, respectively. To describe such different update intervals of each level key, we use functions, which is so-called *time-period map functions*. This functions were also used in [19]. Now, let $\mathcal{T}$ be a (possibly infinite) set of *time*, and $\mathcal{T}_j$ $(0 \leq j \leq \ell - 1)$ be a finite set of *time-periods*. We assume $|\mathcal{T}_0| \geq |\mathcal{T}_1| \geq \cdots \geq |\mathcal{T}_{\ell-1}|$. This means that a lower-level key is updated more frequently than the higher-level keys. Then, we assume there exists a function T_j $(0 \leq j \leq \ell - 1)$ which map time $\mathtt{time} \in \mathcal{T}$ to a time-period $t_j \in \mathcal{T}_j$. For the understanding of readers, by letting $\mathtt{time}$ = `9:59/7th/Oct./2015` and $\ell := 4$, we give an example in Fig. 1 and below. For example, we have $T_0(\mathtt{time}) = t_0^{(19)} =$ `1st-15th/Oct./2015`, $T_1(\mathtt{time}) = t_1^{(10)} =$ `Oct./2015`, $T_2(\mathtt{time}) = t_2^{(5)} =$ `Oct.-Dec./2015`, and $T_3(\mathtt{time}) = t_3^{(2)} =$ `Jul.-Dec./2015`. Namely, in this example, it is assumed that the decryption key, and 1-st, 2-nd, and 3-rd helper keys are updated every half a month, every month, every three months, and every half a year. Further, we can also define a function T_ℓ such that $T_\ell(\mathtt{time}) = 0$ for all $\mathtt{time} \in \mathcal{T}$.

3 Identity-Based Hierarchical Key-Insulated Encryption

3.1 The Model

In ℓ-level hierarchical IKE, a key generation center (KGC) generates an initial decryption key $dk_{\mathtt{I},0}$ and ℓ initial helper keys $hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}$ as a secret key for a user $\mathtt{I}$. Suppose that all time-period map functions $T_0, \ldots, T_{\ell-1}$ are available to all users. The key-updating procedure when the user wants to get a decryption key at current time $\mathtt{time} \in \mathcal{T}$ from the initial helper keys is as follows. The ℓ-th level helper key $hk_{\mathtt{I},0}^{(\ell)}$ is a long-term one and is never updated. First, the user generates *key update* $\delta_{t_{\ell-1}}^{(\ell-1)}$ for the $(\ell-1)$-th level helper key from $hk_{\mathtt{I},0}^{(\ell)}$ and a time-period $t_{\ell-1} := T_{\ell-1}(\mathtt{time}) \in \mathcal{T}_{\ell-1}$. Then, the $(\ell-1)$-th level helper key $hk_{\mathtt{I},0}^{(\ell-1)}$ can be updated by the key update $\delta_{t_{\ell-1}}^{(\ell-1)}$, and the user get the helper key $hk_{\mathtt{I},t_{\ell-1}}^{(\ell-1)}$ at the time-period $t_{\ell-1}$. Similarly, the i-th level helper key $hk_{\mathtt{I},t_i}^{(i)}$ at the time-period $t_i := T_i(\mathtt{time}) \in \mathcal{T}_i$ can be obtained from $hk_{\mathtt{I},0}^{(i)}$ and $\delta_{t_i}^{(i)}$, where $\delta_{t_i}^{(i)}$ is generated from the $(i+1)$-th level helper key $hk_{\mathtt{I},t_{i+1}}^{(i+1)}$. The user can finally get the decryption key $dk_{\mathtt{I},t_0}$ at a time-period $t_0 := T_0(\mathtt{time}) \in \mathcal{T}_0$ from the 1-st level helper key $hk_{\mathtt{I},T_1(\mathtt{time})}^{(1)}$. Anyone can encrypt a plaintext M with the identity $\mathtt{I}$ and current time $\mathtt{time}^*$, and the user can decrypt the ciphertext C with his decryption key $dk_{\mathtt{I},t_0}$ only if $t_0 = T_0(\mathtt{time}^*)$. At $\mathtt{time}' \in \mathcal{T}$, the user can update the time-period of the decryption key from any time-period t_0 to $t_0' := T_0(\mathtt{time}') \in \mathcal{T}_0$ by using key update $\delta_{T_0(\mathtt{time}')}^{(0)}$. The key update $\delta_{T_0(\mathtt{time}')}^{(0)}$ can be obtained from $hk_{\mathtt{I},t_1'}^{(1)}$ only if $t_1' = T_1(\mathtt{time}')$. If not, it is necessary to get $\delta_{T_1(\mathtt{time}')}^{(1)}$ and update $hk_{\mathtt{I},t_1'}^{(1)}$. In this manner, the decryption and helper keys are updated.

An ℓ-level hierarchical IKE scheme Π_{IKE} consists of six-tuple algorithms (PGen, Gen, Δ-Gen, Upd, Enc, Dec) defined as follows. For simplicity, we omit a public parameter in the input of all algorithms except for the PGen algorithm.

- $(pp, mk) \leftarrow \mathsf{PGen}(\lambda, \ell)$: A probabilistic algorithm for parameter generation. It takes a security parameter λ and the maximum hierarchy depth ℓ as input, and outputs a public parameter pp and a master key mk.
- $(dk_{\mathtt{I},0}, hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}) \leftarrow \mathsf{Gen}(mk, \mathtt{I})$: An algorithm for user key generation. It takes mk and an identity $\mathtt{I} \in \mathcal{I}$ as input, and outputs an initial secret key $dk_{\mathtt{I},0}$ associated with $\mathtt{I}$ and initial helper keys $hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}$, where $hk_{\mathtt{I},0}^{(i)}$ $(1 \le i \le \ell)$ is assumed to be stored user's i-th level private device.
- $\delta_{T_{i-1}(\mathtt{time})}^{(i-1)}$ or $\perp \leftarrow \Delta\text{-}\mathsf{Gen}(hk_{\mathtt{I},t_i}^{(i)}, \mathtt{time})$: An algorithm for key update generation. It takes an i-th helper key $hk_{\mathtt{I},t_i}^{(i)}$ at a time period $t_i \in \mathcal{T}_i$ and current time $\mathtt{time}$ as input, and outputs key update $\delta_{T_{i-1}(\mathtt{time})}^{(i-1)}$ if $t_i = T_i(\mathtt{time})$; otherwise, it outputs $\perp$.
- $hk_{\mathtt{I},\tau_i}^{(i)} \leftarrow \mathsf{Upd}(hk_{\mathtt{I},t_i}^{(i)}, \delta_{\tau_i}^{(i)})$: A probabilistic algorithm for decryption key generation. It takes an i-th helper key $hk_{\mathtt{I},t_i}^{(i)}$ at a time-period $t_i \in \mathcal{T}_i$ and key update

$\delta^{(i)}_{\tau_i}$ at a time-period $\tau_i \in \mathcal{T}_i$ as input, and outputs a renewal i-th helper key $hk^{(i)}_{\mathtt{I},\tau_i}$ at τ_i. Note that for any $t_0 \in \mathcal{T}_0$, $hk^{(0)}_{\mathtt{I},t_0}$ means $dk_{\mathtt{I},t_0}$.

- $\langle C, \mathtt{time}\rangle \leftarrow \mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: A probabilistic algorithm for encryption. It takes an identity $\mathtt{I}$, current time $\mathtt{time}$, and a plaintext $M \in \mathcal{M}$ as input, and outputs a pair of a ciphertext and current time $\langle C, \mathtt{time}\rangle$.
- M or $\perp \leftarrow \mathsf{Dec}(dk_{\mathtt{I},t_0}, \langle C, \mathtt{time}\rangle)$: A deterministic algorithm for decryption. It takes $dk_{\mathtt{I},t_0}$ and $\langle C, \mathtt{time}\rangle$ as input, and outputs M or $\perp$, where $\perp$ indicates decryption failure.

In the above model, we assume that Π_{IKE} meets the following correctness property: For all security parameter λ, all $\ell := poly(\lambda)$, all $(mk, pp) \leftarrow \mathsf{PGen}(\lambda, \ell)$, all $M \in \mathcal{M}$, all $(dk_{\mathtt{I},0}, hk^{(1)}_{\mathtt{I},0}, \ldots, hk^{(\ell)}_{\mathtt{I},0}) \leftarrow \mathsf{Gen}(mk, \mathtt{I})$, and all $\mathtt{time} \in \mathcal{T}$, it holds that $M \leftarrow \mathsf{Dec}(dk_{\mathtt{I},T_0(\mathtt{time})}, \mathsf{Enc}(\mathtt{I}, \mathtt{time}, M))$, where $dk_{\mathtt{I},T_0(\mathtt{time})}$ is generated as follows: For $i = \ell, \ldots, 1$, $hk^{(i-1)}_{\mathtt{I},T_{i-1}(\mathtt{time})} \leftarrow \mathsf{Upd}(hk^{(i-1)}_{\mathtt{I},t_{i-1}}, \Delta\text{-}\mathsf{Gen}(hk^{(i)}_{\mathtt{I},T_i(\mathtt{time})}, \mathtt{time}))$, where some $t_i \in \mathcal{T}_i$ and $hk^{(0)}_{\mathtt{I},T_0(\mathtt{time})} := dk_{\mathtt{I},T_0(\mathtt{time})}$.

3.2 Security Definition

We consider a security notion for indistinguishability against key exposure and chosen plaintext attack for IKE (IND-KE-CPA). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-KE-CPA security is defined by

$$Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (pp, mk) \leftarrow \mathsf{PGen}(\lambda), \\ (M_0^*, M_1^*, \mathtt{I}^*, \mathtt{time}^*, state) \leftarrow \mathcal{A}^{KG(\cdot), KI(\cdot,\cdot,\cdot)}(\mathsf{find}, pp), \\ b \xleftarrow{\$} \{0,1\}, C^* \leftarrow \mathsf{Enc}(\mathtt{I}^*, \mathtt{time}^*, M_b^*), \\ b' \leftarrow \mathcal{A}^{KG(\cdot), KI(\cdot,\cdot,\cdot)}(\mathsf{guess}, C^*, state) \end{array} \right] - \frac{1}{2} \right|.$$

where $KG(\cdot)$ and $KI(\cdot,\cdot,\cdot)$ are defined as follows.

- $\boldsymbol{KG}(\cdot)$**:** For a query $\mathtt{I} \in \mathcal{I}$, it stores and returns $(dk_{\mathtt{I},0}, hk^{(1)}_{\mathtt{I},0}, \ldots, hk^{(\ell)}_{\mathtt{I},0})$ by running $\mathsf{Gen}(mk, \mathtt{I})$.
- $\boldsymbol{KI}(\cdot,\cdot,\cdot)$**:** For a query $(i, \mathtt{I}, \mathtt{time}) \in \{0, 1, \ldots, \ell\} \times \mathcal{I} \times \mathcal{T}$, it returns $hk^{(i)}_{\mathtt{I},T_i(\mathtt{time})}$ by running $\delta^{(j-1)}_{T_{j-1}(\mathtt{time})} \leftarrow \Delta\text{-}\mathsf{Gen}(hk^{(j)}_{\mathtt{I},T_j(\mathtt{time})}, \mathtt{time})$ and $hk^{(j-1)}_{\mathtt{I},T_{j-1}(\mathtt{time})} \leftarrow \mathsf{Upd}(hk^{(j-1)}_{\mathtt{I},t}, \delta^{(j-1)}_{T_{j-1}(\mathtt{time})})$ for $j = \ell, \ldots, i+1$ (if $(dk_{\mathtt{I},0}, hk^{(1)}_{\mathtt{I},0}, \ldots, hk^{(\ell)}_{\mathtt{I},0})$ is not stored, it first generates and stores them by running Gen).

$\mathtt{I}^*$ is never issued to the KG oracle. $\mathcal{A}$ can issue any queries $(i, \mathtt{I}, \mathtt{time})$ to the KI oracle if there exists at least one *special level* $j \in \{0, 1, \ldots, \ell\}$ such that

1. For any $\mathtt{time} \in \mathcal{T}$, $(j, \mathtt{I}^*, \mathtt{time})$ is never issued to KI.
2. For any $(i, \mathtt{time}) \in \{0, 1, \ldots, j-1\} \times \mathcal{T}$ such that $T_i(\mathtt{time}) = T_i(\mathtt{time}^*)$, $(i, \mathtt{I}^*, \mathtt{time})$ is never issued to KI.

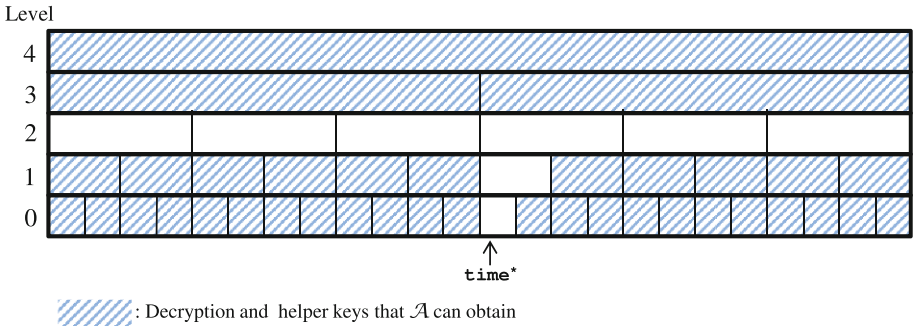


Fig. 2. Pictorial representation of secret keys for $\mathtt{I}^*$ that $\mathcal{A}$ can obtain by issuing to *KI*.

In Fig. 2, we give intuition of keys that $\mathcal{A}$ can obtain by issuing to the *KI* oracle. In this example, let $\ell = 4$ and a special level $j = 2$.

Definition 4 (IND-KE-CPA [19]). *An IKE scheme Π_{IKE} is said to be IND-KE-CPA secure if for all PPT adversaries $\mathcal{A}$, $Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ is negligible in λ.*

Remark 1. As also noted in [19], there is no need to consider *key update exposure* explicitly (i.e. consider an oracle which returns any key update as much as possible) since in the above definition, $\mathcal{A}$ can get such key update from helper keys obtained from the *KI* oracle.

Remark 2. As explained in Sect. 1, in key-insulated cryptography including the public key setting [2,12,17] and the identity-based setting [19,31,32], two kinds of security notions have been traditionally considered: standard security and *strong* security. In most of previous works [2,12,17–19,23,31,32], authors have considered how their scheme could achieve the strong security. We note that IND-KE-CPA security actually includes the strong security, and the fact is easily checked by setting $\ell = 1$.

By modifying the above IND-KE-CPA game so that $\mathcal{A}$ can access to the decryption oracle $Dec(\cdot,\cdot)$, which receives $(\mathtt{I}, \langle C, \mathtt{time}\rangle)$ and returns M or $\perp$, we can also define indistinguishability against key exposure and chosen ciphertext attack for IKE (IND-KE-CCA). $\mathcal{A}$ is not allowed to issue $(\mathtt{I}^*, \langle C^*, \mathtt{time}\rangle)$ such that $T_0(\mathtt{time}) = T_0(\mathtt{time}^*)$ to *Dec*. Let $Adv^{IND\text{-}KE\text{-}CCA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ be $\mathcal{A}$'s advantage against IND-KE-CCA security.

Definition 5 (IND-KE-CCA [19]). *An IKE scheme Π_{IKE} is said to be IND-KE-CCA secure if for all PPT adversaries $\mathcal{A}$, $Adv^{IND\text{-}KE\text{-}CCA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ is negligible in λ.*

4 Our Construction

Our basic idea is a combination of (the variant of) the Jutla–Roy HIBE [22,25] and threshold secret sharing schemes [4,27]. A secret B is divided into ℓ shares $\beta_0, \ldots, \beta_{\ell-1}$, and both the secret and shares are used in exponent of a generator $g_2 \in \mathbb{G}_2$. B is embedded into the exponent of a secret key for $\mathtt{I}^*$ of the Jutla–Roy HIBE, and the resulting key is an ℓ-th level initial helper key $hk_{\mathtt{I},0}^{(\ell)}$. Roughly speaking, B works as "noise". Other initial helper keys $hk_{\mathtt{I},0}^{(i)}$ and an initial decryption key contain $g_2^{-\beta_i}$ and $g_2^{-\beta_0}$, respectively. As a lower-level key is generated, shares are eliminated from the secret B, and finally B is entirely removed when generating (or updating) a decryption key. Intuitively, since no secret keys at some special level $j \in \{0, \ldots, \ell\}$ are exposed, an adversary cannot get all β_i. Hence, he cannot generate valid decryption keys that can decrypt the challenge ciphertext for $\mathtt{I}^*$ at $\mathtt{time}^*$.

An IKE scheme Π_{IKE} =(PGen, Gen, Δ-Gen, Upd, Enc, Dec) is constructed as follows.

– $\mathsf{PGen}(\lambda, \ell)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, \{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}$, $x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z = e(g_1, g_2)^{-x_0\alpha + y_0},\ u_{1,j} := g_1^{-x_{1,j}\alpha + y_{1,j}}\ (0 \le j \le \ell),$$
$$w_1 := g_1^{-x_2\alpha + y_2},\ h_1 := g_1^{-x_3\alpha + y_3}.$$

It outputs

$$pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z),$$
$$mk := (x_0, y_0).$$

– $\mathsf{Gen}(mk, ID)$: It chooses $\beta_0, \ldots, \beta_{\ell-1}, r \xleftarrow{\$} \mathbb{Z}_p$, and let $B := \sum_{i=0}^{\ell-1} \beta_i$. It computes

$$R_j := g_2^{-\beta_j}\ (0 \le j < \ell),$$
$$D_1 := (g_2^{y_2})^r,\ D_1' := g_2^{y_0}\left((g_2^{y_{1,\ell}})^{\mathtt{I}} g_2^{y_3}\right)^r,$$
$$D_2 := (g_2^{x_2})^{-r},\ D_2' := g_2^{-x_0}\left((g_2^{x_{1,\ell}})^{\mathtt{I}} g_2^{x_3}\right)^{-r},$$
$$D_3 := g_2^{r+B},$$
$$K_j := (g_2^{y_{1,j}})^r\ (0 \le j \le \ell - 1),\ K_j' := (g_2^{x_{1,j}})^{-r}\ (0 \le j \le \ell - 1).$$

It outputs

$$dk_{\mathtt{I},0} := R_0,\ hk_{\mathtt{I},0}^{(i)} := R_i\ (1 \le i \le \ell - 1),$$
$$hk_{\mathtt{I},0}^{(\ell)} := (D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{\ell-1}).$$

– Δ-$\mathsf{Gen}(hk^{(i)}_{\mathtt{I},t_i}, \mathtt{time})$: If $t_i \neq T_i(\mathtt{time})$, it outputs $\perp$. Otherwise, parse $hk^{(i)}_{\mathtt{I},t_i}$ as $(R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$.[2] It chooses $\hat{r} \leftarrow \mathbb{Z}_p$, and let $t_j := T_j(\mathtt{time})$ $(i-1 \leq j \leq \ell-1)$. It computes

$$\hat{d}_1 := D_1 (g_2^{y_2})^{\hat{r}},\ \hat{d}'_1 := D'_1 (K_{i-1})^{t_{i-1}} \Big((g_2^{y_{1,\ell}})^{\mathtt{I}} \prod_{j=i-1}^{\ell-1} \big((g_2^{y_{1,j}})^{t_j} \big) g_2^{y_3} \Big)^{\hat{r}},$$

$$\hat{d}_2 := D_2 (g_2^{x_2})^{-\hat{r}},\ \hat{d}'_2 := D'_2 (K'_{i-1})^{t_{i-1}} \Big((g_2^{x_{1,\ell}})^{\mathtt{I}} \prod_{j=i-1}^{\ell-1} \big((g_2^{x_{1,j}})^{t_j} \big) g_2^{x_3} \Big)^{-\hat{r}},$$

$$\hat{d}_3 := D_3 g_2^{\hat{r}},$$

$$\hat{k}_j := K_j (g_2^{y_{1,j}})^{\hat{r}}\ (0 \leq j \leq i-2),\ \hat{k}'_j := K'_j (g_2^{x_{1,j}})^{-\hat{r}}\ (0 \leq j \leq i-2).$$

It outputs $\delta^{(i-1)}_{t_{i-1}} := (\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-2})$.[3]

– $\mathsf{Upd}(hk^{(i)}_{\mathtt{I},t_i}, \delta^{(i)}_{\tau_i})$: Parse $hk^{(i)}_{\mathtt{I},t_i}$ and $\delta^{(i)}_{\tau_i}$ as $(R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$ and $(\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-1})$, respectively. It computes $D_3 := \hat{d}_3 R_i$, and sets $(D_j, D'_j) := (\hat{d}_j, \hat{d}'_j)$ $(j = 1, 2)$ and $(K_j, K'_j) := (\hat{k}_j, \hat{k}'_j)$ $(0 \leq j \leq i-1)$. Finally, it outputs $hk^{(i)}_{\mathtt{I},\tau_i} := (R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$.

– $\mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: It chooses $s, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := M z^s,\ C_1 := g_1^s,\ C_2 := (g_1^{\alpha})^s,\ C_3 := \Big(\prod_{j=0}^{\ell-1} (u_{1,j}^{t_j}) u_{1,\ell}^{\mathtt{I}} w_1^{\mathtt{tag}} h_1 \Big)^s,$$

where $t_j := T_j(\mathtt{time})$ $(0 \leq j \leq \ell-1)$. It outputs $C := (C_0, C_1, C_2, C_3, \mathtt{tag})$.

– $\mathsf{Dec}(dk_{\mathtt{I},t_0}, \langle C, \mathtt{time} \rangle)$: If $t_0 \neq T_0(\mathtt{time})$, then it outputs $\perp$. Otherwise, parse $dk_{\mathtt{I},t_0}$ and C as $(R_0, D_1, D'_1, D_2, D'_2, D_3)$ and $(C_0, C_1, C_2, C_3, \mathtt{tag})$, respectively. It computes

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D'_1) e(C_2, D_2^{\mathtt{tag}} D'_2)}.$$

We show the correctness of our Π_{IKE}. Suppose that r denotes internal randomness of $hk^{(\ell)}_{\mathtt{I},0}$, which are generated when running $\mathsf{Gen}(mk, \mathtt{I})$, and $r^{(j)}$ denotes internal randomness of $\delta^{(j-1)}_{\mathtt{I},t_{j-1}}$ $(1 \leq j \leq \ell)$, which is generated when running Δ-$\mathsf{Gen}(hk^{(j)}_{\mathtt{I},t_j}, \mathtt{time})$. Then we can write $dk_{\mathtt{I},\tau_0} := (R_0, D_1, D'_1, D_2, D'_2, D_3)$ as

$$D_1 := g_2^{y_2 \tilde{r}},\ D'_1 := g_2^{y_0 + \tilde{r}(\mathtt{I} y_{1,\ell} + \sum_{j=0}^{\ell-1} (t_j y_{1,j}) + y_3)},$$

[2] In the case $i = \ell$, R_ℓ means an empty string, namely we have $hk^{(\ell)}_{\mathtt{I},0} := (D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{\ell-1})$.

[3] In the case $i = 1$, $\{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{\ell-1}$ means an empty string, namely we have $\delta^{(0)}_{\mathtt{I},t_0} := (\hat{d}_1, \ldots, \hat{d}_5)$.

$$D_2 := g_2^{x_2\tilde{r}},\ D_2' := g_2^{-x_0-\tilde{r}(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}\left(t_j x_{1,j}\right)+x_3)},\ D_3 := g_2^{\tilde{r}},$$

where $\tilde{r} := r + \sum_{i=1}^{\ell} r^{(j)}$.

Suppose that $dk_{\mathtt{I},t_0} = (R_0, D_1, D_1', D_2, D_2', D_3)$ and $C = (C_0, C_1, C_2, C_3, \mathtt{tag})$ are correctly generated. Then, we have

$$\begin{aligned}
&\frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1')e(C_2, D_2^{\mathtt{tag}} D_2')} \\
&= Me(g_1, g_2)^{(-x_0\alpha+y_0)s} \\
&\cdot \frac{e(g_1^{s(\sum_{j=0}^{\ell-1} t_j(-x_{1,j}\alpha+y_{1,j})+\mathtt{I}(-x_{1,\ell}\alpha+y_{1,\ell})+\mathtt{tag}(-x_2\alpha+y_2)-x_3\alpha+y_3)}, g_2^{\tilde{r}})}{e(g_1^s, g_2^{y_2\tilde{r}\mathtt{tag}+y_0+\tilde{r}(\mathtt{I}y_{1,\ell}+\sum_{j=0}^{\ell-1}\left(t_j y_{1,j}\right)+y_3)})e(g_1^{\alpha s}, g_2^{-x_2\tilde{r}\mathtt{tag}-x_0-\tilde{r}(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}\left(t_j x_{1,j}\right)+x_3)})} \\
&= Me(g_1, g_2)^{(-x_0\alpha+y_0)s} \frac{1}{e(g_1^s, g_2^{y_0})e(g_1^{\alpha s}, g_2^{-x_0})} = M.
\end{aligned}$$

We obtain the following theorem. The proof is postponed to Sect. 5.

Theorem 1. *If the SXDH assumption holds, then the resulting ℓ-level hierarchical IKE scheme Π_{IKE} is IND-KE-CPA secure.*

4.1 Parameters Evaluation and Comparison

First, we show the parameter sizes and computational costs of our hierarchical IKE scheme in Table 1.

Table 1. Parameters evaluation of our ℓ-level hierarchical IKE scheme. $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order p, and $|\mathbb{G}|$ denotes the bit-length of a group element in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$, for simplicity. $|\mathcal{M}|$ and $|\mathbb{Z}_p|$ also denote the bit-length of plaintext and an element in $\mathbb{Z}_p$, respectively. $\#pp$, $\#dk$, $\#hk_i$, and $\#C$ denote sizes of public parameters, decryption keys, i-th helper keys, and ciphertexts, respectively. In computational cost analysis, $[\cdot,\cdot,\cdot,\cdot]$ means the number of [pairing, multi-exponentiation, regular exponentiation, fixed-based exponentiation]. For comparison we mention that relative tunings for the various operations are as follows: [pairing ≈ 5, multi-exp ≈ 1.5, regular-exp $:= 1$, fixed-based-exp $\ll 0.2$].

Scheme	$\#pp$	$\#dk$	$\#hk_i$	$\#C$	Enc. cost	Dec. cost	Assumption
Ours	$(3\ell+13)\lvert\mathbb{G}\rvert$	$6\lvert\mathbb{G}\rvert$	$(2i+6)\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	$[0, 0, \ell+4, 1]$	$[3, 0, 2, 0]$	SXDH

Table 2. Efficiency comparison between our construction and existing schemes. The notation used here is the same as that in Table 1 except for $\#hk$, which denotes the helper key size. What n appears in public-parameter sizes means that the public-parameter size depends on the size of its identity space.

Scheme	$\#pp$	$\#dk$	$\#hk$	$\#C$	Enc. cost	Dec. cost	Assumption
HHSI05 [19] ($\ell = 1$)	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert$	$\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{G}\rvert + \lvert\mathcal{M}\rvert$	$[1, 0, 2, 1]$	$[4, 0, 2, 1]$	CBDH (in ROM)
WLC+08 [32]	$(2n+5)\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert$	$2\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert$	$[0, 1, 3, 1]$	$[3, 0, 0, 0]$	DBDH
Ours ($\ell = 1$)	$16\lvert\mathbb{G}\rvert$	$6\lvert\mathbb{G}\rvert$	$7\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_p\rvert$	$[0, 0, 5, 1]$	$[3, 0, 2, 0]$	SXDH

Also, an efficiency comparison between our IKE scheme and the existing IKE schemes [19,32] is given in Table 2. In fact, the WLC+08 scheme [32] has the threshold property and does not have a hierarchical structure, and therefore, we set the threshold value is one in the WLC+08 scheme and the hierarchy depth is one in the HHSI05 scheme [19] and our scheme for the fair comparison. The HHSI05 scheme meets the IND-KE-CCA security, however the scheme is secure only in the random oracle model (ROM). Both the WLC+08 scheme and ours meet the IND-KE-CPA security in the standard model (i.e. without random oracles). Although assumptions behind these schemes (i.e. the computational bilinear Diffie–Hellman (CBDH), decisional bilinear Diffie–Hellman (DBDH),[4] and SXDH assumptions) are different, they all are static and simple. We emphasize that the threshold structure does not strengthen the underlying DBDH assumption of the WLC+08 scheme since the structure was realized via only threshold secret sharing techniques [4,27]. Note that we do not take into account the parallel IKE scheme [31] since the model of the scheme is slightly different from those of the above schemes. However, the public parameter size of the parallel IKE scheme also depends on the size of its identity space, and we mention that this is due to the underlying Waters IBE [29], not due to the parallel property.

As can be seen, we first achieve the IKE scheme with constant-size parameters in the standard model. Again, we also get the first IKE scheme in the hierarchical setting without random oracles.

5 Proof of Security

We describe how semi-functional ciphertexts and secret keys are generated as follows.

Semi-functional Ciphertext: Parse a normal ciphertext C as $(C_0, C_1, C_2, C_3, \mathtt{tag})$. A semi-functional ciphertext $\widetilde{C} := (\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \widetilde{\mathtt{tag}})$ is computed as follows:

$$
\begin{aligned}
\tilde{C}_0 &:= C_0 e(g_1, g_2)^{-x_0\mu} = M e(g_1, g_2)^{-x_0(\alpha s+\mu)+y_0 s},\\
\tilde{C}_1 &:= C_1,\\
\tilde{C}_2 &:= C_2 g_1^{\mu} = g_1^{\alpha s+\mu},\\
\tilde{C}_3 &:= C_3 \Bigg((g_1^{x_{1,\ell}})^{\mathtt{I}} \prod_{j=0}^{\ell-1} \big((g_1^{x_{1,j}})^{t_j} \big) (g_1^{x_2})^{\mathtt{tag}} g_1^{x_3} \Bigg)^{-\mu}\\
&= C_3 g_1^{-\mu(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j x_{1,j})+x_2\mathtt{tag}+x_3)}\\
&= g_1^{-(\alpha s+\mu)(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j x_{1,j})+x_2\mathtt{tag}+x_3)} g_1^{s(\mathtt{I}y_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j y_{1,j})+y_2\mathtt{tag}+y_3)},
\end{aligned}
$$

[4] The formal definitions of the CBDH and DBDH assumptions are given in Appendix A.

and $\widetilde{\mathtt{tag}} := \mathtt{tag}$, where $\mu \xleftarrow{\$} \mathbb{Z}_p$.

Semi-functional Decryption and Helper Key: Parse a normal helper key $hk^{(i)}_{\mathtt{I},t_i}$ as $(R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$. A semi-functional helper key $\widetilde{hk}^{(i)}_{\mathtt{I},t_i} := (\tilde{R}_i, \tilde{D}_1, \tilde{D}'_1, \tilde{D}_2, \tilde{D}'_2, \tilde{D}_3, \{(\tilde{K}_j, \tilde{K}'_j)\}_{j=0}^{i-1})$ is computed as follows: $R_i := \tilde{R}_i$,

$$
\begin{aligned}
\tilde{D}_1 &:= D_1 g_2^{\gamma} = g_2^{y_2 r + \gamma},\\
\tilde{D}'_1 &:= D_1 g_2^{\gamma\phi} = g_2^{y_0 + r(\mathtt{I} y_{1,\ell} + \sum_{j=i}^{\ell-1}(t_j y_{1,j}) + y_3) + \gamma\phi},\\
\tilde{D}_2 &:= D_2 g_2^{-\frac{\gamma}{\alpha}} = g_2^{-r x_2 - \frac{\gamma}{\alpha}},\\
\tilde{D}'_2 &:= D_2 g_2^{-\frac{\gamma\phi}{\alpha}} = g_2^{-x_0 - r(\mathtt{I} x_{1,\ell} + \sum_{j=i}^{\ell-1}(t_j x_{1,j}) + x_3) - \frac{\gamma\phi}{\alpha}},\\
\tilde{D}_3 &:= D_3,\\
\tilde{K}_j &:= K_j g_2^{\gamma\phi_j} = g_2^{r y_{1,j} + \gamma\phi_j} \ (0 \le j \le i-1),\\
\tilde{K}'_j &:= K'_j g_2^{-\frac{\gamma\phi_j}{\alpha}} = g_2^{-r x_{1,j} - \frac{\gamma\phi_j}{\alpha}} \ (0 \le j \le i-1),
\end{aligned}
$$

where $\gamma, \phi, \{\phi_j\}_{j=0}^{i-1} \xleftarrow{\$} \mathbb{Z}_p$. Note that $hk^{(0)}_{\mathtt{I},t_0}$ means $dk_{\mathtt{I},t_0}$ for any $t_0 \in \mathcal{T}_0$. In particular, $\widetilde{hk}^{(0)}_{\mathtt{I},t_0}$ $(= \widetilde{dk}_{\mathtt{I},t_0})$ is called a semi-functional decryption key. We also note that in order to generate the semi-functional decryption or helper key, $g_2^{\frac{1}{\alpha}}$ is needed in addition to the public parameter.

A semi-functional ciphertext can be decrypted with a normal key. This fact can be easily checked by

$$
\frac{e(g_1^{\mu(\mathtt{I} y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j y_{1,j}) + y_2 \mathtt{tag} + y_3)}, D_3) e(g_1, g_2)^{-x_0 \mu}}{e(g_1^{\mu}, D_1^{\mathtt{tag}} D'_1)} = 1.
$$

Also, a normal ciphertext can be decrypted with a semi-functional decryption key since it holds $e(C_1, g_2^{\gamma \mathtt{tag}} g_2^{\gamma\phi}) e(C_2, g_2^{-\frac{\gamma}{\alpha}\mathtt{tag}} g_2^{-\frac{\gamma\phi}{\alpha}}) = 1$.

A helper or decryption key obtained by running the Δ-Gen and Upd algorithms with a semi-functional helper key is also semi-functional.

Proof (*of Theorem 1*). Based on [22,25], we prove the theorem through a sequence of games. We first define the following games:

$\mathsf{Game}_{\mathsf{Real}}$: This is the same as the IND-KE-CPA game described in Sect. 3.

Game_0: This is the same as $\mathsf{Game}_{\mathsf{Real}}$ except that the challenge ciphertext is semi-functional.

Game_k $(1 \le k \le q)$**:** This is the same as Game_0 except for the following modification: Let q be the maximum number of identities issued to the KG or KI oracles, and $\mathtt{I}_i$ $(1 \le i \le q)$ be an i-th identity issued to the oracles. If queries regarding the first k identities $\mathtt{I}_1, \ldots, \mathtt{I}_k$ are issued, then semi-functional decryption and/or helper keys are returned. The rest of keys (i.e., keys regarding $\mathtt{I}_{k+1}, \ldots, \mathtt{I}_q$) are normal.

$\mathsf{Game}_{\mathsf{Final}}$: This is the same as Game_q except that the challenge ciphertext is a semi-functional one of a random element of $\mathbb{G}_T$.

Let S_{Real}, S_k $(0 \le k \le q)$, and S_{Final} be the probabilities that the event $b' = b$ occurs in $\mathsf{Game}_{\mathsf{Real}}$, Game_k, and $\mathsf{Game}_{\mathsf{Final}}$, respectively. Then, we have

$$Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda) \le |S_{\mathsf{Real}} - S_0| + \sum_{i=1}^{q} |S_{i-1} - S_i| + |S_q - S_{\mathsf{Final}}| + |S_{\mathsf{Final}} - \frac{1}{2}|.$$

The rest of the proof follows from the following lemmas.

Lemma 1. *If the DDH1 assumption holds, then it holds that* $|S_{\mathsf{Real}} - S_0| \le Adv^{DDH1}_{\mathcal{G},\mathcal{B}}(\lambda)$.

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_1^{c_1}, g_1^{c_2}, g_2, T)$ of the DDH1 problem. Then, $\mathcal{B}$ randomly chooses $x_0, y_0, \{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$, and creates

$$z := e(g_1^{c_1}, g_2)^{-x_0} e(g_1, g_2)^{y_0},\ u_{1,j} := (g_1^{c_1})^{-x_{1,j}} g_1^{y_{1,j}}\ (0 \le j \le \ell),$$
$$w_1 := (g_1^{c_1})^{-x_2} g_1^{y_2},\ h_1 := (g_1^{c_1})^{-x_3} g_1^{y_3}.$$

$\mathcal{B}$ sends $pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z)$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $mk := (x_0, y_0)$ and we implicitly set $\alpha := c_1$.

$\mathcal{B}$ can simulate the KG and KI oracles since $\mathcal{B}$ knows the master key.

In the challenge phase, $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*, \mathtt{time}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ chooses $\mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$, and let $t_j^* := T_j(\mathtt{time}^*)$ $(0 \le j \le \ell - 1)$. $\mathcal{B}$ computes

$$C_0^* := M_d e(T, g_2)^{-x_0} e(g_1^{c_2}, g_2)^{y_0},\ C_1^* := g_1^{c_2},\ C_2^* := T,$$
$$C_3^* := T^{-\mathtt{I}^* x_{1,\ell} - \sum_{j=0}^{\ell-1}(t_j^* x_{1,j}) - x_2 \mathtt{tag}^* - x_3} (g_1^{c_2})^{\mathtt{I}^* y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* y_{1,j}) + y_2 \mathtt{tag}^* + y_3}.$$

$\mathcal{B}$ sends $C^* := (C_0^*, C_1^*, C_2^*, C_3^*, \mathtt{tag}^*)$ to $\mathcal{A}$.

If $b = 0$, then the above ciphertext is normal by setting $s := c_2$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$C_0^* = M_d e(g_1, g_2)^{-x_0(c_1 c_2 + \mu) + y_0 c_2} = M_d e(g_1, g_2)^{-x_0(\alpha s + \mu) + y_0 s},$$
$$C_2^* = g_1^{c_1 c_2 + \mu} = g_1^{\alpha s + \mu},$$
$$C_3^* = g^{-(c_1 c_2 + \mu)(\mathtt{I}^* x_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* x_{1,j}) + x_2 \mathtt{tag}^* + x_3)} g_1^{c_2(\mathtt{I}^* y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* y_{1,j}) + y_2 \mathtt{tag}^* + y_3)}$$
$$= g^{-(\alpha s + \mu)(\mathtt{I}^* x_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* x_{1,j}) + x_2 \mathtt{tag}^* + x_3)} g_1^{s(\mathtt{I}^* y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* y_{1,j}) + y_2 \mathtt{tag}^* + y_3)}.$$

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the DDH1 problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Lemma 2. *For every $k \in \{1, \ldots, q\}$, if the DDH2 assumption holds, then it holds that $|S_{k-1} - S_k| \leq Adv^{DDH2}_{\mathcal{G},\mathcal{B}}(\lambda)$.*

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_2, g_2^{c_1}, g_2^{c_2}, T)$ of the DDH2 problem. Then, $\mathcal{B}$ randomly chooses $x'_0, y_0, \{(x'_{1,j}, y'_{1,j}, y''_{1,j})\}_{j=0}^{\ell}, x'_2, x'_3, y'_3, y''_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and (implicitly) sets

$$
\begin{aligned}
&x_0 := \frac{x'_0 + y_0}{\alpha},\ x_{1,j} := \frac{x'_{1,j} + y_{1,j}}{\alpha},\ \text{where } y_{1,j} := y'_{1,j} + c_2 y''_{1,j}\ (0 \leq j \leq \ell),\\
&x_2 := \frac{x'_2 + c_2}{\alpha},\ y_2 := c_2,\\
&x_3 := \frac{x'_3 + y_3}{\alpha},\ \text{where } y_3 := y'_3 + c_2 y''_3.
\end{aligned}
$$

$\mathcal{B}$ creates

$$
\begin{aligned}
&z := e(g_1, g_2)^{-x'_0},\ u_{1,j} := g_1^{-x'_{1,j}}\ (0 \leq j \leq \ell),\ w_1 := g_1^{-x'_2},\ h_1 := g_1^{-x'_3},\\
&g_2^{x_{1,j}} := g_2^{\frac{x'_{1,j}+y'_{1,j}}{\alpha}} (g_2^{c_2})^{\frac{y''_{1,j}}{\alpha}}\ (0 \leq j \leq \ell),\ g_2^{y_{1,j}} := g_2^{y'_{1,j}} (g_2^{c_2})^{y''_{1,j}}\ (0 \leq j \leq \ell),\\
&g_2^{x_2} := g_2^{\frac{x'_2}{\alpha}} (g_2^{c_2})^{\frac{1}{\alpha}},\ g_2^{y_2} := g_2^{c_2},\ g_2^{x_3} := g_2^{\frac{x'_3+y'_3}{\alpha}} (g_2^{c_2})^{\frac{y''_3}{\alpha}},\ g_2^{y_3} := g_2^{y'_3} (g_2^{c_2})^{y''_3}.
\end{aligned}
$$

$\mathcal{B}$ sends $pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{y_2}, g_2^{x_3}, g_2^{y_3}, z)$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $mk := (x_0, y_0)$.

We show how $\mathcal{B}$ simulates the *KG* and *KI* oracles. Let $\mathtt{I}_i$ $(1 \leq i \leq q)$ be an i-th identity issued to the oracles. Without loss of generality, we consider $\mathcal{A}$ issues all identities $\mathtt{I}_i \neq \mathtt{I}^*$ to the *KG* oracle, and issues only queries regarding $\mathtt{I}^*$ to the *KI* oracle.

KG Oracle. $\mathcal{B}$ creates $k-1$ semi-functional decryption and helper keys, and embeds T into the k-th keys. The rest of keys are normal.

Case $i < k$**:** After receiving $\mathtt{I}_i$, $\mathcal{B}$ creates and returns semi-functional keys. Since $\mathcal{B}$ knows the master key and α, $\mathcal{B}$ can create both normal and semi-functional keys.

Case $i = k$**:** After receiving $\mathtt{I}_k$, $\mathcal{B}$ creates semi functional keys by embedding T as follows: $\mathcal{B}$ chooses $\beta_0, \ldots, \beta_{\ell-1} \xleftarrow{\$} \mathbb{Z}_p$ and sets $B := \sum_{j=0}^{\ell-1} \beta_j$. $\mathcal{B}$ computes

$$
\begin{aligned}
&R_j := g_2^{-\beta_j}\ (0 \leq j < \ell),\\
&D_1 := T,\\
&D'_1 := g_2^{y_0} (g_2^{c_1})^{\mathtt{I}_k y'_{1,\ell} + y'_3} T^{\mathtt{I}_k y''_{1,\ell} + y''_3},\\
&D_2 := \left((g_2^{c_1})^{x'_2} T \right)^{-\frac{1}{\alpha}},\\
&D'_2 := g_2^{-\frac{x'_0}{\alpha}} (g_2^{c_1})^{-\frac{\mathtt{I}_k (x'_{1,\ell} + y'_{1,\ell}) + x'_3 + y'_3}{\alpha}} g_2^{-\frac{y_0}{\alpha}} T^{-\frac{\mathtt{I}_k y''_{1,\ell} + y''_3}{\alpha}},
\end{aligned}
$$

$$D_3 := g_2^{c_1} g_2^{B},$$
$$K_j := (g_2^{c_1})^{y'_{1,j}} (T)^{y''_{1,j}} \ (0 \le j \le \ell - 1),$$
$$K'_j := (g_2^{c_1})^{-\frac{x'_{1,j}+y'_{1,j}}{\alpha}} T^{-\frac{y''_{1,j}}{\alpha}} \ (0 \le j \le \ell - 1).$$

$\mathcal{B}$ sets $dk_{\mathtt{I},0} := R_0$, $hk_{\mathtt{I},0}^{(i)} := R_i$ $(1 \le i \le \ell - 1)$, $hk_{\mathtt{I},0}^{(\ell)} := (D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{\ell-1})$. If $b = 0$, then it is easy to see that the above keys are normal by setting $r := c_1$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$\begin{aligned}
D_1 &:= T = g_2^{c_1 c_2 + \gamma} = g_2^{y_2 r + \gamma}, \\
D'_1 &:= g_2^{y_0} (g_2^{c_1})^{\mathtt{I}_k y'_{1,\ell} + y'_3} T^{\mathtt{I}_k y''_{1,\ell} + y''_3} \\
&= g_2^{y_0 + c_1(\mathtt{I}_k(y'_{1,\ell} + c_2 y''_{1,\ell}) + y'_3 + c_2 y''_3)} g_2^{\gamma(\mathtt{I}_k y''_{1,\ell} + y''_3)} = g_2^{y_0 + r(\mathtt{I}_k y_{1,\ell} + y_3)} g_2^{\gamma\phi}, \\
D_2 &:= \left((g_2^{c_1})^{x'_2} T\right)^{-\frac{1}{\alpha}} = g_2^{-\frac{c_1(x'_2+c_2)}{\alpha}} g_2^{-\frac{\gamma}{\alpha}} = g_2^{-r x_2} g_2^{-\frac{\gamma}{\alpha}}, \\
D'_2 &:= g_2^{-\frac{x'_0}{\alpha}} (g_2^{c_1})^{-\frac{\mathtt{I}_k(x'_{1,\ell} + y'_{1,\ell}) + x'_3 + y'_3}{\alpha}} g_2^{-\frac{y_0}{\alpha}} T^{-\frac{\mathtt{I}_k y''_{1,\ell} + y''_3}{\alpha}} \\
&= g_2^{-\frac{(x'_0 + y_0) + c_1(\mathtt{I}_k(x'_{1,\ell} + y'_{1,\ell} + c_2 y''_{1,\ell}) + (x'_3 + y'_3 + c_2 y''_3))}{\alpha}} g_2^{-\frac{\gamma(\mathtt{I}_k y''_{1,\ell} + y''_3)}{\alpha}} \\
&= g_2^{-x_0 - r(\mathtt{I}_k x_{1,\ell} + x_3)} g_2^{-\frac{\gamma\phi}{\alpha}}, \\
K_j &:= (g_2^{c_1})^{y'_{1,j}} (T)^{y''_{1,j}} = g_2^{c_1(y'_{1,j} + c_2 y''_{1,j})} g_2^{\gamma y''_{1,j}} = g_2^{r y_{1,j}} g_2^{\gamma \phi_j} \ (0 \le j \le \ell - 1), \\
K'_j &:= (g_2^{c_1})^{-\frac{x'_{1,j}+y'_{1,j}}{\alpha}} T^{-\frac{y''_{1,j}}{\alpha}} \\
&= g_2^{-\frac{c_1(x'_{1,j} + y'_{1,j} + c_2 y''_{1,j})}{\alpha}} g_2^{-\frac{\gamma y''_{1,j}}{\alpha}} = g_2^{-r x_{1,j}} g_2^{-\frac{\gamma \phi_j}{\alpha}} \ (0 \le j \le \ell - 1),
\end{aligned}$$

where $T := g_2^{c_1 c_2 + \gamma}$, $r := c_1$, $\phi := \mathtt{I}_k y''_{1,\ell} + y''_3$, and $\phi_j := y''_{1,j}$ $(0 \le j \le \ell - 1)$. Since $y''_{1,j}$ and y''_3 are chosen uniformly at random, ϕ and ϕ_j are also uniformly distributed.

Case $i > k$**:** After receiving $\mathtt{I}_i$, $\mathcal{B}$ creates and returns normal keys by using the master key.

KI Oracle. Suppose that $\mathcal{A}$ issues $k - 1$ identities $\mathtt{I}_1, \ldots, \mathtt{I}_{k-1}$ to the *KG* oracle, and then issues a query $(i, \mathtt{I}^*, \mathtt{time})$ (i.e., $\mathtt{I}^* (= \mathtt{I}_k)$) to the *KI* oracle. Note that for some special level $j \in \{0, \ldots, \ell\}$, $\mathcal{A}$ cannot issue $\mathtt{time}$ such that $T_i(\mathtt{time}) = T_i(\mathtt{time}^*)$ if $i < j$ ($\mathcal{B}$ does not need to know where level is special one in advance). $\mathcal{B}$ creates and stores semi-functional decryption and helper keys $(\widetilde{d}_{\mathtt{I}^*,0}, \widetilde{hk}_{\mathtt{I}^*,0}^{(1)}, \ldots, \widetilde{hk}_{\mathtt{I}^*,0}^{(\ell)})$ as in the case $i = k$ of the *KG* oracle. We also note that from the second query, $\mathcal{B}$ answers queries by using the stored keys. Then, $\mathcal{B}$ repeatedly runs $\delta_{t_{j-1}}^{(j-1)} \leftarrow \Delta\text{-}\mathsf{Gen}(hk_{\mathtt{I}^*,t_j}^{(j)}, \mathtt{time}^*)$ and $hk_{\mathtt{I}^*,t_{j-1}^*}^{(j-1)} \mathsf{Upd}(hk_{\mathtt{I}^*,0}^{(j-1)}, \delta_{t_{j-1}}^{(j-1)})$ for $j = \ell, \ldots, i+1$, where $t_\ell := 0$ and $t_j := T_j(\mathtt{time})$ $(0 \le j \le \ell - 1)$. Again, the key generated by semi-functional helper keys is also semi-functional. $\mathcal{B}$ returns $hk_{\mathtt{I}^*,t_i}^{(i)}$ to $\mathcal{A}$.

In the challenge phase, $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*, \mathtt{time}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$, and sets $t_j^* := T_j(\mathtt{time}^*)$ $(0 \le j \le \ell - 1)$. However, $\mathcal{B}$ cannot create the semi-functional ciphertext for $\mathtt{I}^*$ without knowledge of c_2 (and hence $y_{1,j}$ $(0 \le j \le \ell)$ and y_3). To generate the semi-functional ciphertext without the knowledge, $\mathcal{B}$ sets

$$\widetilde{\mathtt{tag}}^* := -\sum_{j=0}^{\ell-1}(t_j^* y''_{1,j}) - \mathtt{I}^* y''_{1,\ell} - y''_3.$$

Since $y''_{1,0}, \ldots, y''_{1,\ell}$ and y''_3 are chosen uniformly at random, probability distribution of $\widetilde{\mathtt{tag}}^*$ is also uniformly at random from $\mathcal{A}$'s view.[5] Then, $\mathcal{B}$ chooses $s, \mu \xleftarrow{\$} \mathbb{Z}_p$, and computes

$$\begin{aligned}
\tilde{C}_0^* &:= M_d^* z^s e(g_1, g_2)^{-x_0\mu} = M_d^* e(g_1, g_2)^{-x_0(\alpha s+\mu)+y_0 s},\\
\tilde{C}_1^* &:= g_1^s,\\
\tilde{C}_2^* &:= g_1^{\alpha s+\mu}\\
\tilde{C}_3^* &:= \Big(\prod_{j=0}^{\ell-1}(u_{1,j}^{t_j^*}) u_{1,\ell}^{\mathtt{I}} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s g_1^{\mu(y'_3+\sum_{j=0}^{\ell-1}(t_j^* y'_{1,j})+\mathtt{I}^* y'_{1,\ell})}\\
&= \Big(\prod_{j=0}^{\ell-1}(u_{1,j}^{t_j^*}) u_{1,\ell}^{\mathtt{I}} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s\\
&\quad \cdot g_1^{\mu(\sum_{j=0}^{\ell-1}(t_j^*(y'_{1,j}+c_2 y''_{1,j}))+\mathtt{I}^*(y'_{1,\ell}+c_2 y''_{1,\ell})+c_2\widetilde{\mathtt{tag}}^*+y'_3+c_2 y''_3)}\\
&\quad \cdot g_1^{-c_2\mu(\sum_{j=0}^{\ell-1}(t_j^* y''_{1,j})+\mathtt{I}^* y''_{1,\ell}+\widetilde{\mathtt{tag}}^*+y''_3)}\\
&= \Big(\prod_{j=0}^{\ell-1}(u_{1,j}^{t_j^*}) u_{1,\ell}^{\mathtt{I}} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s g_1^{\mu(\sum_{j=0}^{\ell-1}(t_j^* y_{1,j})+\mathtt{I}^* y_{1,\ell}+y_2\widetilde{\mathtt{tag}}^*+y_3)}.
\end{aligned}$$

$\mathcal{B}$ sends $\widetilde{C}^* := (\tilde{C}_0^*, \tilde{C}_1^*, \tilde{C}_2^*, \tilde{C}_3^*, \widetilde{\mathtt{tag}}^*)$ to $\mathcal{A}$.

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the DDH2 problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Lemma 3. $|S_q - S_{\mathsf{Final}}| \le \frac{q}{p}$.

Proof. We modify the setup procedure and the semi-functional keys generation procedure in Game_q, and the modification turns out $\mathsf{Game}_{\mathsf{Final}}$. We show that before and after the modification are statistically indistinguishable without probability $\frac{q}{p}$.

[5] The fact that the formula in such a form is uniformly distributed was traditionally studied in the context of unconditionally secure authentication protocols (e.g., [5,21,28]).

In the setup phase, we randomly choose $x'_0, y_0, \{(x'_{1,j}, y_{1,j})\}_{j=0}^{\ell}, x'_2, y_2, x'_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and set

$$x_0 := \frac{x'_0 + y_0}{\alpha},\ x_{1,j} := \frac{x'_{1,j} + y_{1,j}}{\alpha}\ (0 \le j \le \ell),\ x_2 := \frac{x'_2 + y_2}{\alpha},\ x_3 := \frac{x'_3 + y_3}{\alpha}.$$

$\mathcal{B}$ creates

$$z := e(g_1, g_2)^{-x'_0},\ u_{1,j} := g_1^{-x'_{1,j}}\ (0 \le j \le \ell),\ w_1 := g_1^{-x'_2},\ h_1 := g_1^{-x'_3},$$
$$g_2^{x_{1,j}} := g_2^{\frac{x'_{1,j}+y_{1,j}}{\alpha}}\ (0 \le j \le \ell),\ g_2^{x_2} := g_2^{\frac{x'_2+y_2}{\alpha}},\ g_2^{x_3} := g_2^{\frac{x'_3+y_3}{\alpha}}.$$

We set $pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{y_2}, g_2^{x_3}, g_2^{y_3}, z)$ and $mk := (x_0, y_0)$.

When generating (initial) semi-functional keys, we choose $\beta_0, \ldots, \beta_{\ell-1}, r, \phi', \phi'_0, \ldots, \phi'_{\ell-1}, \gamma \xleftarrow{\$} \mathbb{Z}_p$, and (implicitly) set $B := \sum_{j=0}^{\ell-1} \beta_i$, $\phi' := y_0 + r(\mathtt{I} y_{1,\ell} + y_3) + \gamma\phi$, and $\phi'_j := r y_{1,j} + \gamma\phi_j$ $(0 \le j \le \ell - 1)$. We compute

$$\begin{aligned}
\tilde{R}_j &:= g_2^{-\beta_j}\ (0 \le j \le \ell - 1),\\
\tilde{D}_1 &:= g_2^{y_2 r + \gamma},\\
\tilde{D}'_1 &:= g_2^{\phi'} = g_2^{y_0 + r(\mathtt{I} y_{1,\ell} + y_3) + \gamma\phi},\\
\tilde{D}_2 &:= g_2^{-r\frac{x'_2+y_2}{\alpha} - \frac{\gamma}{\alpha}} = g_2^{-r x_2 - \frac{\gamma}{\alpha}},\\
\tilde{D}'_2 &:= g_2^{-\frac{1}{\alpha}(\phi' + x'_0 + r(x'_3 + \mathtt{I} x'_{1,\ell}))}\\
&= g_2^{-\frac{1}{\alpha}(x'_0 + y_0 + r\mathtt{I}(x'_{1,\ell} + y_{1,\ell}) + \gamma\phi + r(x'_3 + y_3))} = g_2^{-x_0 - r(\mathtt{I} x_{1,\ell} + x_3) - \frac{\gamma\phi}{\alpha}},\\
\tilde{D}_3 &:= g_2^{r+B},\\
\tilde{K}_j &:= g_2^{\phi'_j} = g_2^{r y_{1,j} + \gamma\phi_j}\ (0 \le j \le \ell - 1),\\
\tilde{K}'_j &:= g_2^{-\frac{r x'_{1,j} + \phi'_j}{\alpha}} = g_2^{-\frac{r(x'_{1,j} + y_{1,j}) + \gamma\phi_j}{\alpha}} = g_2^{-r x_{1,j} - \frac{\gamma\phi_j}{\alpha}}\ (0 \le j \le \ell - 1).
\end{aligned}$$

We set $dk_{\mathtt{I},0} := \tilde{R}_0$, $hk_{\mathtt{I},0}^{(j)} := \tilde{R}_j$ $(1 \le j \le \ell - 1)$, and $hk_{\mathtt{I},0}^{(\ell)} := (\tilde{D}_1, \tilde{D}'_1, \tilde{D}_2, \tilde{D}'_2, \tilde{D}_3, \{(\tilde{K}_j, \tilde{K}'_j)\}_{j=0}^{\ell-1})$. We emphasize that although the above secret keys are well-formed, y_0, $\{y_{1,j}\}_{j=0}^{\ell}$, and y_3 are not used in the above procedure.

On the other hand, the first component of the challenge ciphertext is generated as $\tilde{C}_0^* := M_b z^s e(g_1, g_2)^{-x_0\mu} = M_b e(g_1, g_2)^{-x_0(\alpha s + \mu) + y_0}$. This means that y_0, which is independent of secret keys and public parameters, masks $\tilde{C}_0^*$, and hence $\tilde{C}_0^*$ becomes the ciphertext of a random element of $\mathbb{G}_T$.

Since γ is chosen uniformly at random, ϕ and ϕ_j are distributed uniformly at random if $\gamma \neq 0$. An event that $\gamma = 0$ occurs with probability $1/p$. Every query regarding $\mathtt{I}_i$ $(1 \le i \le q)$ may cause this event, and hence, we have $|S_q - S_{\mathsf{Final}}| \le \frac{q}{p}$. □

Proof of Theorem 1. From Lemmas 1, 2, and 3, we have $Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda) \leq |S_{\mathsf{Real}} - S_0| + \sum_{i=1}^{q} |S_{i-1} - S_i| + |S_q - S_{\mathsf{Final}}| + |S_{\mathsf{Final}} - \frac{1}{2}| \leq Adv^{DDH1}_{\mathcal{G},\mathcal{B}}(\lambda) + q \cdot Adv^{DDH2}_{\mathcal{G},\mathcal{B}}(\lambda) + \frac{q}{p}$. □

6 Chosen-Ciphertext Security

Boneh et al. [6] proposed an well-known transformation from $\ell+1$-level CPA-secure HIBE (and one-time signature (OTS)) to ℓ-level CCA-secure HIBE. We cannot apply this transformation to a hierarchical IKE scheme *in a generic way* since it does not have delegating functionality. However, we can apply their techniques to the underlying Jutla–Roy HIBE of our hierarchical IKE, and therefore we obtain CCA-secure scheme. We show the detailed construction as follows. We assume a verification key vk is appropriately encoded as an element of $\mathbb{Z}_p$ when it is used in exponent of ciphertexts.

Let $\Pi_{OTS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ be an OTS scheme.[6] An ℓ-level hierarchical IKE scheme Π_{IKE} =(PGen, Gen, Δ-Gen, Upd, Enc, Dec) is constructed as follows.

– $\mathsf{PGen}(\lambda, \ell)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses x_0, y_0, $\{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}$, $\hat{x}_1, \hat{y}_1, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z = e(g_1, g_2)^{-x_0\alpha + y_0},\ u_{1,j} := g_1^{-x_{1,j}\alpha + y_{1,j}}\ (0 \leq j \leq \ell),$$
$$\hat{u}_1 := g_1^{-\hat{x}_1\alpha + \hat{y}_1},\ w_1 := g_1^{-x_2\alpha + y_2},\ h_1 := g_1^{-x_3\alpha + y_3}.$$

It outputs

$$pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, \hat{u}_1, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{\hat{x}_1}, g_2^{\hat{y}_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z),$$
$$msk := (x_0, y_0).$$

– $\mathsf{Gen}(mk, ID)$: It chooses $\beta_0, \ldots, \beta_{\ell-1}, r \xleftarrow{\$} \mathbb{Z}_p$, and let $B := \sum_{i=0}^{\ell-1} \beta_i$. It computes

$$R_j := g_2^{-\beta_j}\ (0 \leq j < \ell),$$
$$D_1 := (g_2^{y_2})^r,\ D_1' := g_2^{y_0}\left((g_2^{y_{1,\ell}})^{\mathtt{I}} g_2^{y_3}\right)^r,$$
$$D_2 := (g_2^{x_2})^{-r},\ D_2' := g_2^{-x_0}\left((g_2^{x_{1,\ell}})^{\mathtt{I}} g_2^{x_3}\right)^{-r},$$
$$D_3 := g_2^{r+B},$$
$$K_j := (g_2^{y_{1,j}})^r\ (0 \leq j \leq \ell-1),\ K_j' := (g_2^{x_{1,j}})^{-r}\ (0 \leq j \leq \ell-1),$$
$$K_{vk} := (g_2^{\hat{y}_1})^r,\ K_{vk}' := (g_2^{\hat{x}_1})^{-r}.$$

[6] The formal description of the OTS is given in Appendix A.

It outputs

$$dk_{\mathrm{I},0} := R_0,\ hk_{\mathrm{I},0}^{(i)} := R_i\ (1 \le i \le \ell - 1),$$
$$hk_{\mathrm{I},0}^{(\ell)} := (D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{i-1}, K_{vk}, K_{vk}').$$

– Δ-$\mathsf{Gen}(hk_{\mathrm{I},t_i}^{(i)}, \mathtt{time})$: If $t_i \neq T_i(\mathtt{time})$, it outputs $\perp$. Otherwise, parse $hk_{\mathrm{I},t_i}^{(i)}$ as $(R_i, D_1, D_1',\ D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{i-1}, K_{vk}, K_{vk}')$. It chooses $\hat{r} \leftarrow \mathbb{Z}_p$, and let $t_j := T_j(\mathtt{time})$ $(i-1 \le j \le \ell - 1)$. It computes

$$\hat{d}_1 := D_1 (g_2^{y_2})^{\hat{r}},\ \hat{d}_1' := D_1' (K_{i-1})^{t_{i-1}} \Big((g_2^{y_{1,\ell}})^{\mathrm{I}} \prod_{j=i-1}^{\ell-1} \big((g_2^{y_{1,j}})^{t_j} \big) g_2^{y_3} \Big)^{\hat{r}},$$

$$\hat{d}_2 := D_2 (g_2^{x_2})^{-\hat{r}},\ \hat{d}_2' := D_2' (K_{i-1}')^{t_{i-1}} \Big((g_2^{x_{1,\ell}})^{\mathrm{I}} \prod_{j=i-1}^{\ell-1} \big((g_2^{x_{1,j}})^{t_j} \big) g_2^{x_3} \Big)^{-\hat{r}},$$

$$\hat{d}_3 := D_3 g_2^{\hat{r}},$$
$$\hat{k}_j := K_j (g_2^{y_{1,j}})^{\hat{r}}\ (0 \le j \le i-2),\ \hat{k}_j' := K_j' (g_2^{x_{1,j}})^{-\hat{r}}\ (0 \le j \le i-2),$$
$$\hat{k}_{vk} := K_{vk} (g_2^{\hat{y}_1})^{\hat{r}},\ \hat{k}_{vk}' := K_{vk}' (g_2^{\hat{x}_1})^{\hat{r}}.$$

It outputs $\delta_{t_{i-1}}^{(i-1)} := (\hat{d}_1, \hat{d}_1', \hat{d}_2, \hat{d}_2', \hat{d}_3, \{(\hat{k}_j, \hat{k}_j')\}_{j=0}^{i-2}, \hat{k}_{vk}, \hat{k}_{vk}')$.

– $\mathsf{Upd}(hk_{\mathrm{I},t_i}^{(i)}, \delta_{\tau_i}^{(i)})$:
Parse $hk_{\mathrm{I},t_i}^{(i)}$ and $\delta_{\tau_i}^{(i)}$ as $(R_i, D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{i-1},\ K_{vk}, K_{vk}')$ and $(\hat{d}_1, \hat{d}_1', \hat{d}_2, \hat{d}_2', \hat{d}_3, \{(\hat{k}_j, \hat{k}_j')\}_{j=0}^{i-1}, \hat{k}_{vk}, \hat{k}_{vk})$, respectively. It computes $D_3 := \hat{d}_3 R_i$, and sets $(D_j, D_j') := (\hat{d}_j, \hat{d}_j')$ $(j = 1, 2)$, $(K_j, K_j') := (\hat{k}_j, \hat{k}_j')$ $(0 \le j \le i-1)$, and $(K_{vk}, K_{vk}') := (\hat{k}_{vk}, \hat{k}_{vk}')$. Finally, it outputs $hk_{\mathrm{I},\tau_i}^{(i)} := (R_i, D_1, D_1', D_2, D_2',\ D_3, \{(K_j, K_j')\}_{j=0}^{i-1}, K_{vk}, K_{vk}')$.

– $\mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: It first runs $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$. It chooses $s, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := M z^s,\ C_1 := g_1^s,\ C_2 := (g_1^{\alpha})^s,\ C_3 := \Big(\prod_{j=0}^{\ell-1} \big(u_{1,j}^{t_j} \big) u_{1,\ell}^{\mathrm{I}} \hat{u}_1^{vk} w_1^{\mathtt{tag}} h_1 \Big)^s,$$

where $t_j := T_j(\mathtt{time})$ $(0 \le j \le \ell - 1)$. It also runs $\sigma \leftarrow \mathsf{Sign}(sk, (C_0, C_1, C_2, C_3, \mathtt{tag}))$, and outputs $C := (vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma)$.

– $\mathsf{Dec}(dk_{\mathrm{I},t_0}, \langle C, \mathtt{time} \rangle)$: If $t_0 \neq T_0(\mathtt{time})$, then it outputs $\perp$. Otherwise, parse dk_{I,t_0} and C as $(R_0, D_1, D_1', D_2, D_2', D_3, K_{vk}, K_{vk}')$ and $(vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma)$, respectively. If $\mathsf{Ver}(vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma) \to 0$, then it outputs $\perp$. Otherwise, it computes

$$\hat{D}_1' := D_1' (K_{vk})^{vk},\ \hat{D}_2' := D_2' (K_{vk}')^{vk}.$$

Finally, it outputs

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} \hat{D}_1') e(C_2, D_2^{\mathtt{tag}} \hat{D}_2')}.$$

The correctness of the above IKE scheme Π_{IKE} can be checked as in our CPA-secure IKE scheme described in Sect. 4.

We obtain the following theorem. The proof is omitted since this theorem can be easily proved by combining Boneh et al.'s techniques [6] and our proof techniques of Theorem 1.

Theorem 2. *If the underlying OTS scheme Π_{OTS} is sUF-OT secure and the SXDH assumption holds, then the resulting ℓ-level hierarchical IKE scheme Π_{IKE} is IND-KE-CCA secure.*

7 Conclusion

In this paper, we first proposed hierarchical IKE scheme in the standard model. When the hierarchy is one, our scheme achieves constant-size parameters including public parameters, decryption and helper keys, and ciphertexts, and hence our scheme is more efficient than the existing scheme [32] in the sense of parameter sizes. Our scheme is based on the Jutla–Roy HIBE [22] (and its variant [25]) and techniques of threshold secret sharing schemes [4,27].

Acknowledgments. We would like to thank anonymous PKC 2016 referees for their helpful comments. The first author is supported by JSPS Research Fellowships for Young Scientists. This work (Yohei Watanabe) was supported by Grant-in-Aid for JSPS Fellows Grant Number 25·3998. This work (Junji Shikata) was partially conducted under the auspices of the MEXT Program for Promoting the Reform of National Universities.

A Definitions

We give the formal definitions of the CBDH and DBDH assumptions and OTS. In the following, we assume the Type-1 pairing (i.e., $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$).

Computational Bilinear Diffie–Hellman (CBDH) Assumption. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the CBDH problem as follows.

$$Adv^{CBDH}_{\mathcal{G},\mathcal{A}}(\lambda) := \Pr\left[T = e(g,g)^{c_1 c_2 c_3} \;\middle|\; \begin{array}{l} (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}, \\ c_1, c_2, c_3 \xleftarrow{\$} \mathbb{Z}_p, \\ T \leftarrow \mathcal{A}(\lambda, g, g^{c_1}, g^{c_2}, g^{c_3}) \end{array} \right].$$

Definition 6. *The CBDH assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv^{CBDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.*

Decisional Bilinear Diffie–Hellman (DBDH) Assumption. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DBDH problem as follows.

$$Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr \left[b' = b \;\middle|\; \begin{array}{l} (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}, \\ c_1, c_2, c_3 \xleftarrow{\$} \mathbb{Z}_p, \\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 1 \text{ then } W := \hat{e}(g,g)^{c_1 c_2 c_3}, \\ \text{else } W \xleftarrow{\$} \mathbb{G}_T, \\ b' \leftarrow \mathcal{A}(\lambda, g, g^{c_1}, g^{c_2}, g^{c_3}, W) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 7. *The DBDH assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.*

One-Time Signature. An OTS scheme Π_{OTS} consists of three-tuple algorithms (KGen, Sign, Ver) defined as follows.

- $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$: It takes a security parameter λ and outputs a pair of a public key and a secret key (vk, sk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: It takes the secret key sk and a message $m \in \mathcal{M}$ and outputs a signature σ.
- 1 or $0 \leftarrow \mathsf{Ver}(vk, m, \sigma)$: It takes the public key vk and a pair of a message and a signature (m, σ), and then outputs 1 or 0.

We assume that Π_{OTS} meets the following *correctness* property: For all $\lambda \in \mathbb{N}$, all $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$, and all $m \in \mathcal{M}$, it holds that $1 \leftarrow \mathsf{Ver}(vk, (m, \mathsf{Sign}(sk, m)))$.

We describe the notion of strong unforgeability against one-time attack ($\mathsf{sUF\text{-}OT}$). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against sUF-OT security is defined by

$$Adv^{sUF\text{-}OT}_{\Pi_{OTS},\mathcal{A}}(\lambda) := \Pr \left[1 \leftarrow \mathsf{Ver}(vk, m^*, \sigma^*) \wedge (m^*, \sigma^*) \neq (m, \sigma) \;\middle|\; \begin{array}{l} (vk, sk) \leftarrow \mathsf{KGen}(\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign(\cdot)}(vk) \end{array} \right].$$

$Sign(\cdot)$ is a *signing oracle* which takes a message m as input, and then returns σ by running $\mathsf{Sign}(sk, m)$. $\mathcal{A}$ is allowed to access to the above oracle only once.

Definition 8. *An OTS scheme Π_{OTS} is said to be sUF-OT secure if for all PPT adversaries $\mathcal{A}$, $Adv^{sUF\text{-}OT}_{\Pi_{OTS},\mathcal{A}}(\lambda)$ is negligible in λ.*

References

1. 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)
2. Bellare, M., Palacio, A.: Protecting against key-exposure: strongly key-insulated encryption with optimal threshold. Appl. Algebra Eng. Commun. Comput. **16**(6), 379–396 (2006)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. pp. 321–334. S&P 2007, May 2007
4. Blakley, G.: Safeguarding cryptographic keys. In: Proceedings of the 1979 AFIPS National Computer Conference. pp. 313–317. AFIPS Press, Montvale (1979)
5. den Boer, B.: A simple and key-economical unconditional authentication scheme. J. Comput. Secur. **2**, 65–72 (1993)
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen ciphertext security from identity based encryption. SIAM J. Comput. **36**(5), 1301–1328 (2007)
7. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
9. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings – the role of Ψ revisited. Discrete Appl. Math. **159**(13), 1311–1322 (2011)
10. Dodis, Y., Franklin, M., Katz, J., Miyaji, A.: Intrusion-resilient public-key encryption. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 19–32. Springer, Heidelberg (2003)
11. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: A generic construction for intrusion-resilient public-key encryption. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 81–98. Springer, Heidelberg (2004)
12. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
13. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
14. Dodis, Y., Luo, W., Xu, S., Yung, M.: Key-insulated symmetric key cryptography and mitigating attacks against cryptographic cloud software. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2012, pp. 57–58. ACM, New York (2012)
15. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008)
16. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
17. Hanaoka, G., Hanaoka, Y., Imai, H.: Parallel key-insulated public key encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 105–122. Springer, Heidelberg (2006)
18. Hanaoka, G., Weng, J.: Generic constructions of parallel key-insulated encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 36–53. Springer, Heidelberg (2010)

19. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-based hierarchical strongly key-insulated encryption and its application. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 495–514. Springer, Heidelberg (2005)
20. Itkis, G., Reyzin, L.: SiBIR: signer-base intrusion-resilient signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
21. Johansson, T., Kabatianskii, G.A., Smeets, B.J.M.: On the relation between A-codes and codes correcting independent errors. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 1–11. Springer, Heidelberg (1994)
22. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013)
23. Libert, B., Quisquater, J.-J., Yung, M.: Parallel key-insulated public key encryption without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 298–314. Springer, Heidelberg (2007)
24. Ramanna, S.C., Chatterjee, S., Sarkar, P.: Variants of waters' dual system primitives using asymmetric pairings. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 298–315. Springer, Heidelberg (2012)
25. Ramanna, S.C., Sarkar, P.: Efficient (anonymous) compact HIBE from standard assumptions. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 243–258. Springer, Heidelberg (2014)
26. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
27. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
28. Taylor, R.: An integrity check value algorithm for stream ciphers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 40–48. Springer, Heidelberg (1994)
29. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
30. 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)
31. Weng, J., Liu, S., Chen, K., Ma, C.: Identity-based parallel key-insulated encryption without random oracles: security notions and construction. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 409–423. Springer, Heidelberg (2006)
32. Weng, J., Liu, S., Chen, K., Zheng, D., Qiu, W.: Identity-based threshold key-insulated encryption without random oracles. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 203–220. Springer, Heidelberg (2008)

Signatures

Attribute-Based Signatures for Circuits from Bilinear Map

Yusuke Sakai(✉), Nuttapong Attrapadung, and Goichiro Hanaoka

AIST, Tsukuba, Japan
{yusuke.sakai,n.attrapadung,hanaoka-goichiro}@aist.go.jp

Abstract. In attribute-based signatures, each signer receives a signing key from the authority, which is associated with the signer's attribute, and using the signing key, the signer can issue a signature on any message under a predicate, if his attribute satisfies the predicate. One of the ultimate goals in this area is to support a wide class of predicates, such as the class of *arbitrary circuits*, with *practical efficiency* from *a simple assumption*, since these three aspects determine the usefulness of the scheme. We present an attribute-based signature scheme which allows us to use an arbitrary circuit as the predicate with practical efficiency from the symmetric external Diffie-Hellman assumption. We achieve this by combining the efficiency of Groth-Sahai proofs, which allow us to prove algebraic equations efficiently, and the expressiveness of Groth-Ostrovsky-Sahai proofs, which allow us to prove any NP relation via circuit satisfiability.

Keywords: Attribute-based signatures · Groth-Sahai proofs · Groth-Ostrovsky-Sahai proofs

1 Introduction

1.1 Attribute-Based Signatures

In an ordinary digital signature scheme, a signer has a signing key and publicizes its corresponding verification key. The verification is performed with respect to such a public key, and hence during the verification process, those who made the signature is uniquely determined. In other words, digital signatures provide nothing for privacy or anonymity requirements.

The concept of attribute-based signatures is introduced by Maji, Prabhakaran, and Rosulek [21], in order to relax this firm correspondence between a signer and a signature. In an attribute-based signature scheme, there is an attribute authority, and each signer receives from the authority a signing key associated with his attribute. Once a signer receives a signing key, he is able to issue a signature on any message, under a predicate satisfied by his attribute. The signature is *anonymous*, that is, the signature tells a verifier that the party

The first author is supported by a JSPS Fellowship for Young Scientists.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 283–300, 2016.
DOI: 10.1007/978-3-662-49384-7_11

who generates the signature has an attribute satisfying the predicate, but further information on the signer's identity or attribute is completely hidden from the verifier.

One of the active lines of research on attribute-based signatures is to support a larger class of predicates with practical efficiency. The state-of-the-art results along this line is the scheme by Okamoto and Takashima for non-monotone span programs from bilinear groups [24] and the scheme by Tang, Li, and Liang for any circuits from multilinear maps [27]. The ultimate goal in this line is achieving a large class of predicate, such as *the class of arbitrary circuits*, while keeping the scheme *practically efficient* and relying on *a simple assumption*, since these three aspects determine the usefulness of the scheme in practice. However, neither of above two schemes and in fact neither of any existing scheme does not achieve this ultimate goal.

Bellare and Fuchsbauer proposed a versatile cryptographic primitive called policy-based signatures [2]. They showed a generic construction of an attribute-based signature scheme from a policy-based signature scheme. There are two ways of instantiating their generic construction. Namely, the one is an instantiation with NIZK for general NP languages such as the Groth-Ostrovsky-Sahai proof system [13], and the other is an instantiation with NIZK for specific algebraic equations such as the Groth-Sahai proof system [14]. Although the authors of [2] did not explicitly mention (they only dealt with monotone predicates), the former may be extended to support the class of arbitrary circuits. However, it suffers from a large overhead of the signature size due to a Karp reduction to an NP-complete problem. The latter can be instantiated efficiently, but the supported class is restricted to conjunctions and disjunctions of pairing-product equations.

In summary, it still remains open whether it is possible to construct an attribute-based signature scheme that supports circuit predicates with practical efficiency from simple assumptions.

1.2 Efficient Non-interactive Zero-Knowledge

In this section we review non-interactive zero-knowledge (NIZK) proofs, which can be useful building blocks for constructing attribute-based signatures.

NIZK proofs allow us to prove that a secret information satisfies a public condition without revealing the secret beyond the truth of the condition. This primitive is extremely useful and widely studied in the area of cryptography. It has been an important research topic to expand the class of the predicate that proof systems support, as well as to improve the efficiency of proof systems.

Recent developments in zero-knowledge proofs include the proof system by Groth, Ostrovsky, and Sahai [13] and the one by Groth and Sahai [14]. The former can prove any NP relation via circuit satisfiability, but it suffers from large overhead due to a Karp reduction. The latter is very efficient, but the class of the relation is restricted to algebraic equations, and hence it cannot treat arbitrary NP relation in general.

A natural question is whether it is possible to construct a proof system which is as expressive as the Groth-Ostrovsky-Sahai proof system, and is at the same time as efficient as the Groth-Sahai proof system. In this paper, we investigate a case study of a fusion of Groth-Ostrovsky-Sahai and Groth-Sahai proofs in case of attribute-based signatures, and show that by this idea, we can construct a practical attribute-based signature for circuits from bilinear maps.

1.3 Our Contribution

In this paper, we present an attribute-based signature scheme for arbitrary circuits of unbounded size and depth with practical efficiency, from a simple assumption over bilinear groups. Our attribute-based signature scheme satisfies perfect privacy and adaptive unforgeability. The scheme is based on a witness indistinguishable and extractable non-interactive proof system and an existentially unforgeable signature scheme. All the building blocks can be instantiated solely from the symmetric external Diffie-Hellman (SXDH) assumption [14,16], and thus we can obtain a perfectly private and adaptively unforgeable scheme from the same assumption.

Our scheme is fairly practical. The signature size grows as around one kilobyte per each gate, which is comparable to the existing schemes such as the schemes by Maji et al. [21] and the scheme by Okamoto and Takashima [24]. We note that Maji et al.'s schemes and the Okamoto-Takashima scheme are less expressive than ours, namely, Maji et al.'s schemes support monotone span programs, while the Okamoto-Takashima scheme supports non-monotone span programs. In addition, our scheme drastically improves efficiency when we compare it with related schemes of Bellare and Fuchsbauer [2] and Tang, Li, and Liang [27]. As stated above, the former scheme is a generic construction of attribute-based signatures from policy-based signatures and the latter scheme is an attribute-based signature scheme for circuits from multilinear maps.

It would be interesting to note the contrast between our scheme and its encryption counterparts, namely, the attribute-based encryption schemes for circuits [9,11,12]. We highlight that our scheme only requires a simple and popular bilinear map assumption, namely the SXDH assumption to prove its security, whereas the encryption counterparts require powerful lattice assumptions or multilinear maps. This is reminiscent of the fact that an identity-based signature scheme can be constructed only from a standard digital signature scheme [3,17,22], while identity-based encryption requires a very strong assumption [5].

1.4 Technique

The basic idea behind our construction is simple: to sign anonymously, a signer receives a signature on his attribute from the authority, and proves the knowledge of this signature together with a proof that shows the signed attribute satisfies a public circuit. The signature that the signer receives works as a certificate,

which certifies the signer having the attribute, and forbids the third party from signing in the name of his attribute.

To implement this idea, we need to overcome two difficulties. The first difficulty is (1) *simultaneously and efficiently proving circuit satisfiability of the attribute and the validity of the certificate on that attribute*. The other difficulty is (2) *binding the proof from the first part to a message to be signed*. In the following we give more detailed explanations on these difficulties and our idea for overcoming them.

(1) Proving Circuit Satisfiability and Certificate Validity. The first difficulty is expressing circuit satisfiability of an attribute in zero-knowledge, while keeping the entire proof system efficient. We need to prove not only circuit satisfiability of an attribute, but also validity of a certificate. The Groth-Ostrovsky-Sahai proof system enables us to prove circuit satisfiability, but its direct use does not allow us to prove the validity of the certificate *efficiently*, since, if we were to use the Groth-Ostrovsky-Sahai proof system, we must represent validity of a certificate in a circuit via a Karp reduction, which is highly inefficient.

Nevertheless, our starting point is still the technique of Groth, Ostrovsky, and Sahai [13]. In this technique, to prove circuit satisfiability, the prover first computes commitments to assignments to each wire, and then proves that for each gate the incoming wires u and v and the outgoing wire w satisfy the NAND relation $\neg(u \wedge v) = w$.

We instantiate this idea with Groth-Sahai proofs. We need Groth-Sahai proofs, rather than a simple adoption of the Groth-Ostrovsky-Sahai proof system because we need to handle not only Boolean relations (for the NAND gates as above), but also algebraic equations at the same time. The need for algebraic equations comes from the necessity to certifying attributes. As stated above, the authority signs on attributes to certify that each signer can sign in the name of his attribute. Hence we need to prove the validity of the certificate, and for this purpose we employ Groth-Sahai proofs, together with structure-preserving signatures [1].

Therefore, we need to translate the idea of the Groth-Ostrovsky-Sahai proof system into the Groth-Sahai proof system. Namely, we need to translate the NAND relation $\neg(u \wedge v) = w$ into a bilinear equation, which is what the Groth-Sahai proofs can prove. We do this by *arithmetizing* the relation. That is, let u and v be the assignments to incoming wires then w be the assignment to the outgoing wire, and the prover proves the equation $1 - u \cdot v = w$ to prove the NAND relation.

(2) Binding the Proof to a Message. The other difficulty is binding the proof to a single message in order to resist chosen-message attacks. Although we want to prove knowledge of certificates to sign anonymously, this dose not suffice for resisting chosen-message attacks. This is because the proof is not bound to the message, and hence the adversary can reuse the signature (the proof) on some message to a signature on another message.

To overcome this difficulty, we introduce an OR-proof technique, following Maji et al. [21]. In this technique, the signer proves the knowledge of the certificate *or* a signature on a dummy attribute, which is an extra attribute unused in the real protocol, and differs message by message.

The point is that different messages have different dummy attributes. To be more specific, an (attribute-based) signature on message M proves the knowledge of a signature on some attribute *or* a signature on a dummy attribute x, while a signature on a different message M^* proves the knowledge of a signature on an attribute *or* a signature on another dummy attribute x^*. By this means, if an adversary sees a signature on M and forges a signature on M^*, then a reduction extracts a witness from the forgery and obtains a signature on x^* of the underlying signature scheme. With this x^* the reduction reduces the forgery for the attribute-based signature scheme to a forgery for the underlying signature scheme.

1.5 Related Work

Maji et al. [20,21] introduced the notion of attribute-based signatures, and presented three constructions which have perfect privacy and adaptive unforgeability. The first two schemes combine a digital signature scheme and Groth-Sahai proofs. These two schemes are instantiated respectively with the Boneh-Boyen signature scheme [4] and with the Waters signature scheme [29]. The third construction is proven secure in the generic group model. Following Maji et al.'s results, Li and Kim [19], Siamak and Safavi-Naini [26], and Li et al. [18] presented attribute-based signature schemes, which are proven secure only in the selective model of unforgeability. Another drawback of these schemes is relatively narrow class of the supported predicates. Namely Li and Kim's scheme [19] only supports conjunction predicates, while Siamak and Safavi-Naini's scheme [26] and Li et al.'s scheme [18] support threshold predicates. Escala, Herranz, and Morillo presented an attribute-based signature with adaptive unforgeability [8]. Okamoto and Takashima presented an attribute-based signature scheme which is adaptively unforgeable and supports non-monotone span programs as predicates [24]. Recently, Herranz et al. [15], followed by Chen et al. [6], presented attribute-based signature schemes with constant-size signatures for threshold predicates. The former has selective unforgeability while the latter has adaptive unforgeability. Wang and Chen [28] presented an attribute-based signature scheme from a lattice assumption with selective unforgeability. Tang, Li, and Liang [27] presented an attribute-based signature scheme for bounded-depth circuits from multilinear maps. Most recently, Mridul and Pandit presented various attribute-based signature schemes such as for Boolean formulas or for regular languages from q-type assumptions [23].

Escala, Herranz, and Morillo presented a traceable attribute-based signature scheme (under the name of "revocable" attribute-based signatures) [8], which allows a trusted authority to identify who made a signatures. Okamoto and Takashima presented a decentralized attribute-based signature scheme [25], which removes the necessity of any trusted setup in the system. Following these

works, El Kaafarani, Ghadafi, and Khader presented a decentralized traceable attribute-based signature scheme [7]. Ghadafi revisited the security notion of decentralized traceable attribute-based signatures, and introduced, among other things, a new security notion of non-frameability [10].

As for attribute-based encryption for circuits, Gorbunov, Vaikuntanathan, and Wee [11] presented the first attribute-based encryption scheme for circuits. After that, Garg et al. [9] presented an attribute-based encryption scheme for circuits from multilinear maps. Recently, Gorbunov, Vaikuntanathan, and Wee presented a predicate encryption scheme for circuits from a class of learning-with-errors assumptions [12].

2 Preliminary

We say that a function $f\colon \mathbb{N} \to \mathbb{R}$ is negligible if for all $c \in \mathbb{N}$ there exists $x_0 \in \mathbb{N}$ such that $f(x) \leq x^{-c}$ for all $x \geq x_0$.

Representation of Circuit. Here we explain notation for circuits, especially how we identify a circuit. Let C be a circuit with L-bit input and N gates. We assume C is entirely represented by NAND gates. We distinguish the input wires, the internal wires, and the output wire by indices $1, \ldots, L, L+1, \ldots, L+N$, where $1, \ldots, L$ are the input wires, $L+1, \ldots, L+N-1$ are the internal wires, and $L+N$ is the output wire. The topology of the circuit is specified by two functions $I_1, I_2\colon \{L+1, \ldots, L+N\} \to \{1, \ldots, L+N-1\}$. They map a non-input wire to its first and second incoming wires in which these three wires are connected by a NAND gate. We require that $I_1(i) < i$ and $I_2(i) < i$.

Bilinear Groups. Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that on input 1^k outputs a group description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$ where p is a prime, $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative groups generated by g and $\tilde{g}$, respectively, $\mathbb{G}_T$ is a multiplicative group of order p, and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear map.

We say that the decision Diffie-Hellman assumption on $\mathbb{G}_1$ holds if for any probabilistic polynomial-time adversary $\mathcal{A}$

$$\begin{aligned}&|\Pr[\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^k); x, y \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, g^x, g^y, g^{xy}) = 1] \\ &\quad - \Pr[\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^k); x, y, z \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, g^x, g^y, g^z) = 1]|\end{aligned}$$

is negligible. The decision Diffie-Hellman assumption on $\mathbb{G}_2$ is defined similarly. Namely, we say the decision Diffie-Hellman assumption holds on $\mathbb{G}_2$ if

$$\begin{aligned}&|\Pr[\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^k); x, y \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, \tilde{g}^x, \tilde{g}^y, \tilde{g}^{xy}) = 1] \\ &\quad - \Pr[\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^k); x, y, z \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, \tilde{g}^x, \tilde{g}^y, \tilde{g}^z) = 1]|\end{aligned}$$

is negligible. We say that the symmetric external Diffie-Hellman (SXDH) assumption holds if the decision Diffie-Hellman assumptions on both $\mathbb{G}_1$ and $\mathbb{G}_2$ hold.

Groth-Sahai and Groth-Ostrovsky-Sahai Proofs. A non-interactive proof system for the NP relation $R \subset \{0,1\}^* \times \{0,1\}^*$ is defined by following three algorithms $(\mathsf{WISetup}, \mathsf{WIProve}, \mathsf{WIVerify})$: the setup algorithm $\mathsf{WISetup}$ takes as input the security parameter 1^k and outputs a common reference string crs; the proof algorithm takes as input the common reference string crs, a statement x, and a witness w, and outputs a proof π; the verification algorithm $\mathsf{WIVerify}$ takes as input the common reference string crs, the statement x, and the proof π, and outputs 1 or 0 which indicate validity of the proof. As a correctness condition, we require that for all $k \in \mathbb{N}$, $(x,w) \in R$, and $\mathsf{crs} \leftarrow \mathsf{WISetup}(1^k)$, it holds that $\mathsf{WIVerify}(\mathsf{crs}, x, \mathsf{WIProve}(\mathsf{crs}, x, w)) = 1$.

We require a proof system to be perfectly witness indistinguishable (WI) and perfectly extractable. A proof system is perfectly witness indistinguishable if for any $\mathsf{crs} \leftarrow \mathsf{WISetup}(1^k)$, $x \in \{0,1\}^*$, $w_0 \in \{0,1\}^*$, $w_1 \in \{0,1\}^*$ such that (x,w_0), $(x,w_1) \in R$, the two distributions $\mathsf{WIProve}(\mathsf{crs}, x, w_0)$ and $\mathsf{WIProve}(\mathsf{crs}, x, w_1)$ distributes identically. The proof system is perfectly extractable if there are two algorithms $\mathsf{ExtSetup}$ and $\mathsf{Extract}$ that satisfy the following two properties: (1) for any probabilistic polynomial-time adversary $\mathcal{A}$,

$$|\Pr[\mathsf{crs} \leftarrow \mathsf{WISetup}(1^k) : \mathcal{A}(\mathsf{crs}) = 1] - \Pr[(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{ExtSetup}(1^k) : \mathcal{A}(\mathsf{crs}) = 1]|$$

is negligible, and (2) for any probabilistic polynomial-time adversary $\mathcal{A}$,

$$\Pr[(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{ExtSetup}(1^k); (x,\pi) \leftarrow \mathcal{A}(\mathsf{crs}); w \leftarrow \mathsf{Extract}(\mathsf{crs}, \mathsf{ek}, x, \pi) : \mathsf{WIVerify}(\mathsf{crs}, x, \pi) = 1 \text{ and } (x,w) \notin R] = 0.$$

The Groth-Sahai proof system [14] is a proof system which can prove satisfiability of a set of algebraic equations called pairing-product equations in a witness-indistinguishable and extractable manner under the SXDH assumption. In particular the Groth-Sahai proof system can prove satisfiability of a set of pairing-product equations, which are the equation of the form

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{j=1}^{m} e(\mathcal{X}_j, \mathcal{B}_j) \prod_{i=1}^{n} \prod_{j=1}^{m} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = T$$

in which $\mathcal{A}_i \in \mathbb{G}_1$, $\mathcal{B}_j \in \mathbb{G}_2$, $\gamma_{i,j} \in \mathbb{Z}_p$, and $T \in \mathbb{G}_T$ are public constants, and $\mathcal{X}_i \in \mathbb{G}_1$ and $\mathcal{Y}_j \in \mathbb{G}_2$ are private variables (witness). To prove the knowledge of a satisfying assignment, the prover first computes commitments to each witness (we call this the Groth-Sahai commitment), and then computes proofs demonstrating the witness satisfies the equations. The commitment consists of the two group elements in the same group as the witness. For proving a pairing-product equation, Groth-Sahai proofs require eight group elements, in particular four elements in $\mathbb{G}_1$ and four elements in $\mathbb{G}_2$, for each equation. In the case that $n = 0$ and thus the equation to be proved has the form

$$\prod_{j=1}^{m} e(\mathcal{X}_j, \mathcal{B}_j) = T,$$

it only requires two group elements in $\mathbb{G}_2$. See [14] for further detail.

Groth-Ostrovsky-Sahai proofs are the proof system which can prove satisfiability of a circuit which solely consists of NAND gates. The proof algorithm proceeds with a similar way to the Groth-Sahai proofs. Namely, the prover first computes commitments to the assignments to the wires, and then proves each triple (u, v, w) of wires connected by a NAND gate satisfies the NAND relation $\neg(u \wedge v) = w$. See [13] for further detail.

Structure-Preserving Signatures. A signature scheme consists of the following three algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify})$: the key generation algorithm takes as input a security parameter 1^k and outputs a pair $(\mathsf{vk}, \mathsf{sk})$ of the verification key and the signing key; the signing algorithm Sign takes as input the signing key sk and a message m and outputs a signature θ; the verification algorithm Verify takes as input the verification key vk, the message m, and the signature θ, and outputs 1 or 0 indicating validity of the signature. As the correctness condition, it is required to hold that for all $k \in \mathbb{N}$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^k)$, and $m \in \{0,1\}^*$, it $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$.

A signature scheme $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify})$ is said to be existentially unforgeable, if the probability $\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^k); (m^*, \theta^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk}) : \mathsf{Verify}(\mathsf{vk}, m^*, \theta^*) = 1 \wedge \mathrm{m}^* \text{ is not queried}]$ is negligible for all probabilistic polynomial-time adversaries $\mathcal{A}$.

Our scheme can be instantiated using any structure-preserving signature scheme. For concreteness, we employ the recent scheme by Kiltz, Pan, and Wee (KPW) [16], which is efficient and based on the SXDH assumption. For completeness, we describe the KPW signature scheme below. In the description, for a matrix $A = (a_{i,j}) \in {\mathbb{Z}_p}^{n \times m}$ we denote by $[A]_1$

$$[A]_1 = \begin{pmatrix} g^{a_{1,1}} & \cdots & g^{a_{1,m}} \\ \vdots & \ddots & \vdots \\ g^{a_{n,1}} & \cdots & g^{a_{n,m}} \end{pmatrix} \in {\mathbb{G}_1}^{n \times m},$$

and similarly for $[A]_2 \in {\mathbb{G}_2}^{n \times m}$ with generator $\tilde{g}$, and $[A]_T$ with generator $e(g, \tilde{g})$. For two matrices A and B, we denote $e([A]_1, [B]_2) = [AB]_T$.

$\mathsf{Kg}(\mathsf{gk}, 1^L)$**.** Given a description $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$ of bilinear groups and a message length L, choose $a, b \leftarrow \mathbb{Z}_p$, $K \leftarrow {\mathbb{Z}_p}^{(L+1)\times 2}$, let $A = (1|a)^\top \in {\mathbb{Z}_p}^{2\times 1}$, $B = (1|b)^\top \in {\mathbb{Z}_p}^{2\times 1}$, choose K_0, $K_1 \leftarrow {\mathbb{Z}_p}^{2\times 2}$, let $C \leftarrow KA$, $C_0 \leftarrow K_0 A$, $C_1 \leftarrow K_1 A$, $P_0 \leftarrow B^\top K_0$, $P_1 \leftarrow B^\top K_1$. Let $\mathsf{vk}_{\mathsf{Sign}} \leftarrow ([C_0]_2, [C_1]_2, [C]_2, [A]_2)$ and $\mathsf{sk}_{\mathsf{Sign}} \leftarrow (\mathsf{vk}_{\mathsf{Sign}}, K, [P_0]_1, [P_1]_1, [B]_1)$, and output $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}})$.

$\mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, [\boldsymbol{m}]_1)$**.** Given a signing key $\mathsf{sk}_{\mathsf{Sign}} \leftarrow (\mathsf{vk}_{\mathsf{Sign}}, K, [P_0]_1, [P_1]_1, [B]_1)$ and a message $[\boldsymbol{m}]_1 \in {\mathbb{G}_1}^L$, choose $\boldsymbol{r} \leftarrow {\mathbb{Z}_p}^2$ and $\tau \leftarrow \mathbb{Z}_p$, compute

$$\begin{aligned} \theta_1 &\leftarrow [(1|\boldsymbol{m}^\top)K + \boldsymbol{r}^\top(P_0 + \tau P_1)]_1 \in {\mathbb{G}_1}^{1\times 2}, \\ \theta_2 &\leftarrow [\boldsymbol{r}^\top B^\top]_1 \in {\mathbb{G}_1}^{1\times 2}, \\ \theta_3 &\leftarrow [\boldsymbol{r}^\top B^\top \tau]_1 \in {\mathbb{G}_1}^{1\times 2}, \\ \theta_4 &\leftarrow [\tau]_2 \in \mathbb{G}_2. \end{aligned}$$

Let $\theta \leftarrow (\theta_1, \theta_2, \theta_3, \theta_4)$ and output θ.

$\mathsf{Verify}(\mathsf{vk}_{\mathsf{Sign}}, \theta)$**.** Given the verification key $\mathsf{vk}_{\mathsf{Sign}} = ([C_0]_2, [C_1]_2, [C]_2, [A]_2)$, a message $[\boldsymbol{m}]_1 \in {\mathbb{G}_1}^L$, and a signature $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$, check

$$e(\theta_1, [A]_2) = e([(1|\boldsymbol{m})]_1, [C]_2)e(\theta_2, [C_0]_2)e(\theta_3, [C_1]_2),$$
$$e(\theta_2, \theta_4) = e(\theta_3, [1]_2).$$

If they hold, output 1. Otherwise output 0.

Collision-Resistant Hash Functions. A collision-resistant hash function family is defined as a pair $(\mathcal{H}, \mathsf{Hash})$ of two algorithms: the hash key generation algorithm $\mathcal{H}$ is a probabilistic polynomial-time algorithm that on input security parameter 1^k outputs a hash key hk; the hash algorithm Hash is a deterministic polynomial-time algorithm that on input the hash key hk and a message M outputs a hash value h; a collision-resistant hash function family is required to satisfy that for all probabilistic polynomial-time algorithms $\mathcal{A}$ the probability $\Pr[\mathsf{hk} \leftarrow \mathcal{H}(1^k); (M, M') \leftarrow \mathcal{A}(\mathsf{hk}) : \mathsf{Hash}(\mathsf{hk}, M) = \mathsf{Hash}(\mathsf{hk}, M')]$ is negligible in k. We assume that the length of the hash value h is determined by the security parameter 1^k and denote $\ell_{\mathcal{H}} = \ell_{\mathcal{H}}(k)$.

Attribute-Based Signatures. An attribute-based signature scheme is defined by the following four algorithms:

$\mathsf{AttrSetup}(1^k, 1^\ell) \to (\mathsf{pp}, \mathsf{msk})$**.** The setup algorithm takes as input the security parameter 1^k and the length ℓ of attributes, and outputs the public parameter pp and the master secret key msk.

$\mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x) \to \mathsf{sk}_x$**.** The signing key generation algorithm takes as input the public parameter pp, the master secret key msk, and the attribute x, and outputs the signing key sk_x for x.

$\mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}_x, M, C) \to \sigma$**.** The signing algorithm takes as input the public parameter pp, the signing key sk_x, the message M, and the circuit C, and outputs the signature σ.

$\mathsf{AttrVerify}(\mathsf{pp}, M, C, \sigma) \to 1/0$**.** The verification algorithm takes as input the public parameter pp, the message M, the circuit C, and the signature σ, and outputs 1 or 0 indicating the validity of the signature.

As the correctness condition, it is required to satisfy that for all $k, \ell \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{AttrSetup}(1^k, 1^\ell)$, $x \in \{0,1\}^\ell$, $\mathsf{sk}_x \leftarrow \mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x)$, $M \in \{0,1\}^*$, and C such that $C(x) = 1$, it holds that $\mathsf{AttrVerify}(\mathsf{pp}, M, C, \mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}_x, M, C)) = 1$.

We define two security notions for attribute-based signatures. The first notion is privacy, which requires the signature to not leak any information on the signer's identity and attribute beyond the fact that the attribute satisfies the predicate. The other notion is unforgeability, which requires any collusion of signers is unable to forge a new signature with a predicate which is not satisfied by any attribute in the collusion even if they see signatures on messages of their choice.

Definition 1. *An attribute-based signature scheme is perfectly private, if for all* k, $\ell \in \mathbb{N}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{AttrSetup}(1^k, 1^\ell)$, x_0, $x_1 \in \{0,1\}^\ell$, C *such that* $C(x_0) = C(x_1) = 1$, $\mathsf{sk}_0 \leftarrow \mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x_0)$, $\mathsf{sk}_1 \leftarrow \mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x_1)$, *and* $M \in \{0,1\}^*$, *the distribution* $\mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}_0, M, C)$ *and* $\mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}_1, M, C)$ *distributes identically.*

Definition 2. *An attribute-based signature scheme is adaptively unforgeable if the probability that the adversary wins in the following experiment is negligible in* k*:*

1. *The experiment sets up a public parameter and a master secret key as* $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{AttrSetup}(1^k, 1^\ell)$. *Then the experiment sends the adversary* pp.
2. *The adversary is allowed to access the key reveal oracle and the signing oracle: the former, given a query* x, *returns* $\mathsf{sk}_x \leftarrow \mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x)$*; the latter, given a query* (M, C), *returns* $\sigma \leftarrow \mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}, M, C)$ *with arbitrary* $\mathsf{sk} \leftarrow \mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x)$ *such that* $C(x) = 1$.
3. *The adversary halts with output* (M^*, C^*, σ^*).
4. *The adversary wins if the following three conditions hold: (i)* $\mathsf{AttrVerify}(\mathsf{pp}, M^*, C^*, \sigma^*) = 1$, *(ii) the adversary did not query* x *such that* $C^*(x) = 1$, *and (iii) the adversary did not query* (M^*, C^*) *to the signing oracle.*

3 Attribute-Based Signatures for Circuits

In this section we present our attribute-based signature scheme. We assume the input length ℓ is longer than or equal to the output length $\ell_{\mathcal{H}}$ of the hash function, i.e., $\ell \geq \ell_{\mathcal{H}}$. If it does not, we can simply think of a circuit that ignores the extra inputs.

Before presenting the concrete scheme, we explain an overview of the scheme.

As stated in the introduction, the basic idea is that the authority issues a signature (a certificate) on an attribute to certify that the corresponding signer is allowed to sign in the name of his attribute. This corresponds to the $\mathsf{AttrGen}$ algorithm, which computes a structure-preserving signature on the given attribute.

To sign anonymously, the signer proves the knowledge of the certificate received from the authority, as well as proves that the certified attribute satisfies the public circuit. To do this, the signer computes commitments to all the assignments to each wire. Then for each triple (u, v, w) which are connected by a NAND gate, the signer proves that the triple satisfies the NAND relation $1 - u \cdot v = w$. This is implemented by Eqs. (2), (5), and (6).

Since we are instantiating our scheme with a Type III pairing, for each wire we need two commitments in both $\mathbb{G}_1$ and $\mathbb{G}_2$. This is because we need to take a pairing of two wire assignments (Eqs. (5) and (6)) for proving the NAND relation of the three wires. This further requires the signer to prove that two commitments are commitments to the same message. This is done by proving Eqs. (3) and (4), which ensure that the exponents of W_i and $\tilde{W}_i$ are identical.

Lastly, the OR-proof technique is implemented by modifying the circuit C into $\hat{C}$ as in Eq. (1). This circuit ensures that the input $(X_2, \ldots, X_{l+1})$ is either a satisfying assignment of C or the hash value h. Equation (2) ensures that θ is a valid signature on $(X_2, \ldots, X_{l+1})$. They constitute a proof of knowledge of a signature on an attribute or a signature on the dummy attribute determined by the message.

The full description of our scheme is as follows.

$\mathsf{AttrSetup}(1^k, 1^\ell)$**.** Given a security parameter 1^k and an input size 1^ℓ for circuit, generate bilinear group parameter $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^k)$, a witness indistinguishable common reference string $\mathsf{crs} \leftarrow \mathsf{WISetup}(\mathsf{gk})$, a verification key and a signing key $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{Kg}(\mathsf{gk}, 1^{\ell+1})$ and a hash key $\mathsf{hk} \leftarrow \mathcal{H}(1^k)$. Set $\mathsf{pp} = (\ell, \mathsf{crs}, \mathsf{vk}_{\mathsf{Sign}}, \mathsf{hk})$ and $\mathsf{msk} \leftarrow \mathsf{sk}_{\mathsf{Sign}}$, and output $(\mathsf{pp}, \mathsf{msk})$.

$\mathsf{AttrGen}(\mathsf{pp}, \mathsf{msk}, x)$**.** Parse x as $(x_1, \ldots, x_\ell)$. Generate a structure-preserving signature θ on the message

$$(g^0, g^{x_1}, \ldots, g^{x_\ell}) \in {\mathbb{G}_1}^{\ell+1}.$$

Set $\mathsf{sk}_x \leftarrow (x, \theta)$ and output sk_x.

$\mathsf{AttrSign}(\mathsf{pp}, \mathsf{sk}_x, M, C)$**.** Parse sk_x into $((x_1, \ldots, x_\ell), \theta)$ and proceed as follows:

1. Let $h \leftarrow \mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle)$. Expand the circuit C into a larger circuit $\hat{C}$ with $\ell + 1$-bit input as

$$\begin{aligned} \hat{C}(X_1, X_2, \ldots, X_{\ell+1}) = 1 \\ \iff \Big(X_1 = 0 \wedge C(X_2, \ldots, X_{\ell+1}) = 1\Big) \\ \vee \Big(X_1 = 1 \wedge X_2 \| \cdots \| X_{\ell_{\mathcal{H}+1}} = h\Big) \end{aligned} \quad (1)$$

   where the hash value h is hard-wired into $\hat{C}$. Let N be the number of gates in $\hat{C}$ and I_1 and I_2 be the functions that specify the topology of $\hat{C}$.
2. Let $X_1 \leftarrow 0$, $X_2 \leftarrow x_1$, …, $X_{\ell+1} \leftarrow x_\ell$, and then compute the assignment to each non-input wires in $\hat{C}$: for all $i = (\ell+1)+1$, …, $(\ell+1)+(N-1)$

$$X_i \leftarrow 1 - X_{I_1(i)} \cdot X_{I_2(i)}.$$

3. For all $i = 1$, …, $(\ell+1)+(N-1)$, let

$$W_i \leftarrow g^{X_i}, \qquad \tilde{W}_i \leftarrow \tilde{g}^{X_i}.$$

4. Compute a Groth-Sahai commitment com_θ to θ.
5. For all $i = 1$, …, $(\ell+1)+(N-1)$, compute Groth-Sahai commitments com_{W_i} to W_i and $\mathsf{com}_{\tilde{W}_i}$ to $\tilde{W}_i$.
6. Generate a proof π_{Sign} for the verification equation

$$\mathsf{Verify}(\mathsf{vk}_{\mathsf{Sign}}, (W_1, \ldots, W_{\ell+1}), \theta) = 1. \quad (2)$$

7. For all $i = 1, \ldots, \ell + 1$, generate proofs π_i proving the equation
$$e(g, \tilde{W}_i) = e(W_i, \tilde{g}). \tag{3}$$
8. For all $i = (\ell + 1) + 1, \ldots, (\ell + 1) + (N - 1)$, generate proofs π_i proving the equations
$$e(g, \tilde{W}_i) = e(W_i, \tilde{g}), \tag{4}$$
$$e(W_{I_1(i)}, \tilde{W}_{I_2(i)})e(W_i, \tilde{g}) = e(g, \tilde{g}). \tag{5}$$
9. Generate a proofs $\pi_{(\ell+1)+N}$ proving
$$e(W_{I_1((\ell+1)+N)}, \tilde{W}_{I_2((\ell+1)+N)}) = 1. \tag{6}$$
10. Let
$$\sigma = (\mathsf{com}_\theta, \mathsf{com}_{W_1}, \ldots, \mathsf{com}_{W_{(\ell+1)+(N-1)}}, \mathsf{com}_{\tilde{W}_1}, \ldots, \mathsf{com}_{\tilde{W}_{(\ell+1)+(N-1)}}, \pi_{\mathsf{Sign}}, \pi_1, \ldots, \pi_{(\ell+1)+N})$$
and output σ.

$\mathsf{AttrVerify}(\mathsf{pp}, M, C, \sigma)$**.** Verify the proofs with respect to the circuit $\hat{C}$ in Eq. (1) and its topology I_1, I_2 defined by given M and C. Output 1 if all the proofs are verified as valid. Otherwise output 0.

Theorem 1. *Provided the proof system is perfectly witness indistinguishable, the above attribute-based signature scheme is perfectly private. Provided the proof system is perfectly extractable and perfectly witness indistinguishable, the signature scheme is existentially unforgeable, and the hash function family is collision resistant, the above attribute-based signature scheme is adaptively unforgeable.*

Proof. Perfect privacy directly followed from witness indistinguishability of the proof system.

For adaptive unforgeability, the proof proceeds with the following sequence of games:

Game 1**.** This game is identical to the experiment for adaptive unforgeability.

Game 2**.** In this game, the behavior of the signing oracle is modified as follows. Given a signing query (M, C), the experiment computes the hash value $h \leftarrow \mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle)$, let $(h_1 \| \cdots \| h_{\ell_\mathcal{H}}) \leftarrow h$, compute a signature θ on the message
$$(g^1, g^{h_1}, \ldots, g^{h_{\ell_\mathcal{H}}}, 1, \ldots, 1) \in {\mathbb{G}_2}^{\ell+1}$$
with the master secret key $\mathsf{msk} = \mathsf{sk}_{\mathsf{Sign}}$, and then use θ as the witness to compute a signature σ.

Game 3. In this game, the common reference string crs in pp is switched to the extractable common reference string crs generated by the $\mathsf{ExtSetup}$ algorithm as $(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{ExtSetup}(1^k)$.

We denote by succ_i the event that the adversary wins in Game i. We hereafter bound $\Pr[\mathsf{succ}_1]$ to be negligible. From the triangle inequality,

$$\begin{aligned}\Pr[\mathsf{succ}_1] &= \Pr[\mathsf{succ}_1] - \Pr[\mathsf{succ}_2] + \Pr[\mathsf{succ}_2] - \Pr[\mathsf{succ}_3] + \Pr[\mathsf{succ}_3] \\ &\leq |\Pr[\mathsf{succ}_1] - \Pr[\mathsf{succ}_2]| + |\Pr[\mathsf{succ}_2] - \Pr[\mathsf{succ}_3]| + \Pr[\mathsf{succ}_3].\end{aligned}$$

We bound these three terms. The first term $|\Pr[\mathsf{succ}_1] - \Pr[\mathsf{succ}_2]|$ is negligible, due to the witness indistinguishability of the Groth-Sahai proof system. The second term $|\Pr[\mathsf{succ}_2] - \Pr[\mathsf{succ}_3]|$ is also negligible, because the two types of common reference string are indistinguishable.

For the last term, we introduce an event coll. The event coll denotes the event that $\mathsf{Hash}(\mathsf{hk}, \langle M^*, C^* \rangle)$ collides to some of $\mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle)$ where (M, C) is one of the signing queries. Now we have that

$$\Pr[\mathsf{succ}_3] = \Pr[\mathsf{succ}_3 \wedge \mathsf{coll}] + \Pr[\mathsf{succ}_3 \wedge \neg\mathsf{coll}].$$

The probability $\Pr[\mathsf{succ}_3 \wedge \mathsf{coll}]$ is negligible due to the collision-resistance of the hash function. For a formal proof, we construct a simulator that attacks the collision resistance of the hash function family.

Setup. The simulator receives a hash key hk from the experiment. The simulator then generates an extractable common reference string as $(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{ExtSetup}(1^k)$ and verification and signing keys $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \leftarrow \mathsf{Kg}(1^k)$, and then sets $\mathsf{pp} \leftarrow (\ell, \mathsf{crs}, \mathsf{vk}, \mathsf{hk})$ and sends pp to the adversary.

Key reveal query. When the adversary requests the signing key for $x = (x_1, \ldots, x_\ell)$, the simulator runs the signing algorithm to obtain a signature $\theta \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (g^0, g^{x_1}, \ldots, g^{x_\ell}))$. The simulator responds with $\mathsf{sk}_x = (x, \theta)$.

Signing query. When the adversary requests a signature on M under a circuit C, the simulator computes the hash value $h \leftarrow \mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle)$, lets $(h_1 \| \cdots \| h_{\ell_{\mathcal{H}}}) \leftarrow h$, then further computes the signature $\theta \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, (g^1, g^{h_1}, \ldots, g^{h_{\ell_{\mathcal{H}}}}, 1, \ldots, 1))$, the circuit $\hat{C}$ as in Eq. (1), and proof π using θ as the witness. The simulator responds with $\sigma = \pi$.

Forgery. When the adversary outputs a tuple (M^*, C^*, σ^*), the simulator searches for a signing query (M, C) that satisfies $\mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle) = \mathsf{Hash}(\mathsf{hk}, \langle M^*, C^* \rangle)$. If it is found and the winning condition (i)–(iii) in Definition 2 is satisfied, the simulator outputs $(\langle M, C \rangle, \langle M^*, C^* \rangle)$ as a collision. Otherwise, the simulator outputs $(\perp, \perp)$.

The simulator successfully outputs a collision, if the event $\mathsf{succ}_3 \wedge \mathsf{coll}$ occurs. In particular, whenever the simulator outputs $(\langle M, C \rangle, \langle M^*, C^* \rangle)$, we have that $\langle M, C \rangle \neq \langle M^*, C^* \rangle$. This is because the winning condition forbids the adversary to output M^* and C^* which are queried to the signing oracle, and thus (M, C) differs from (M^*, C^*). Hence $\Pr[\mathsf{succ}_3 \wedge \mathsf{coll}]$ is negligible.

For $\Pr[\mathsf{succ}_3 \wedge \neg\mathsf{coll}]$, we construct a simulator that attacks the existential unforgeability of the underlying signature scheme. The construction of the simulator is as follows.

Setup. The simulator is given a verification key $\mathsf{vk}_{\mathsf{Sign}}$ of the signature scheme. The simulator sets up the extractable common reference string of the proof system as $(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{ExtSetup}(1^k)$. The simulator sends $\mathsf{pp} = (\ell, \mathsf{crs}, \mathsf{vk}_{\mathsf{Sign}}, \mathsf{hk})$ to the adversary.

Key reveal query. When the adversary requests the signing key for an attribute $x = (x_1, \ldots, x_\ell)$, the simulator requests, to its signing oracle, a signature on the message

$$(g^0, g^{x_1}, \ldots, g^{x_\ell}) \in {\mathbb{G}_1}^{\ell+1}.$$

Then the simulator receives a signature θ. The simulator sends $\mathsf{sk}_x = \theta$ to the adversary.

Signing query. When the adversary requests a signature on a message M under the circuit C, the simulator computes the hash value $h = (h_1 \| \cdots \| h_{\ell_{\mathcal{H}}}) \leftarrow \mathsf{Hash}(\mathsf{hk}, \langle M, C \rangle)$, then requests a signature on the message

$$(g^1, g^{h_1}, \ldots, g^{h_{\ell_{\mathcal{H}}}}, 1, \ldots, 1) \in {\mathbb{G}_1}^{\ell+1}$$

to its signing oracle. The simulator receives a signature θ. The simulator computes a proof π using the signature θ as the witness. The simulator sends $\sigma = \pi$ to the adversary.

Forgery. When the adversary outputs a forgery (M^*, C^*, σ^*), the simulator extracts the witness

$$\theta, W_1, \ldots, W_{(\ell+1)+(N-1)}, \tilde{W}_1, \ldots, \tilde{W}_{(\ell+1)+(N-1)}.$$

Due to the extractability of the Groth-Sahai proof system, we can assume that the witness satisfies Eqs. (2)–(6).

Now below we argue that the pair

$$((W_1, \ldots, W_{\ell+1}), \theta)$$

constitutes a legitimate forgery for the underlying signature scheme. We have three cases to be dealt with.

1. Assume that $(W_1, \ldots, W_{\ell+1})$ is of the form

$$(g^{X_1}, \ldots g^{X_{\ell+1}}) \in {\mathbb{G}_2}^{\ell+1} \text{ where } X_1 = 0 \text{ and } X_2, \ldots, X_{\ell+1} \in \{0, 1\}.$$

In this case, due to Eqs. (3)–(6), we have that $\hat{C}(X_1, \ldots, X_{\ell+1}) = 1$, and hence we also have that $C(X_2, \ldots X_{\ell+1}) = 1$. Because the experiment forbids the adversary to query such $(X_2, \ldots, X_{\ell+1})$ as a key reveal query, we can conclude that the simulator has not queried $(g^0, g^{X_2}, \ldots g^{X_{\ell+1}})$ to its signing oracle. Hence, due to the equation Eq. (2), the pair $((g^0, g^{X_2}, \ldots, g^{X_{\ell+1}}), \theta)$ constitutes a legitimate forgery to the signature scheme.

2. Assume that $(W_1, \ldots, W_{\ell+1})$ is of the form

$$(g^{X_1}, \ldots, g^{X_{\ell+1}}) \in {\mathbb{G}_2}^{\ell+1}$$
$$\text{where } X_1 = 1, X_2, \ldots, X_{\ell_{\mathcal{H}}+1} \in \{0, 1\}, X_{\ell_{\mathcal{H}}+2} = \cdots = X_{\ell+1} = 0$$

In this case, due to Eqs. (3)–(6), we have that $(X_2 \| \cdots \| X_{\ell_{\mathcal{H}}+1}) = \mathsf{Hash}(\mathsf{hk}, \langle C^*, M^* \rangle)$. Since we are now considering the event $\neg\mathsf{coll}$, we have that the adversary has not queried (C, M) such that $\mathsf{Hash}(\mathsf{hk}, \langle C, M \rangle) = \mathsf{Hash}(\mathsf{hk}, \langle C^*, M^* \rangle)$ to the signing oracle. Therefore the simulator has not queried $(g^1, g^{X_2}, \cdots g^{X_{\ell_{\mathcal{H}}}}, 1, \ldots, 1)$ to its signing oracle, and thus $((g^1, g^{X_2}, \cdots g^{X_{\ell_{\mathcal{H}}}}, 1, \ldots, 1), \theta)$ constitutes a legitimate forgery.

3. Assume that $(W_1, \ldots, W_{\ell+1})$ is neither of the above two forms. In this case, the simulator does not issue any query of this form at all, and thus $((W_1, \ldots, X_{\ell+1}), \theta)$ is a legitimate forgery.

In any case, the pair $((W_1, \ldots, X_{\ell+1}), \theta)$ constitutes the forgery, and thus the simulator outputs this pair as a forgery.

The above construction shows that whenever the event $\mathsf{succ}_3 \wedge \neg\mathsf{coll}$ occurs, the simulator succeeds in producing the forgery of the signature scheme. It implies that $\Pr[\mathsf{succ}_3 \wedge \neg\mathsf{coll}]$ is negligible. □

4 Performance

In this section we compare our scheme with the Maji et al. (MPR11) schemes [21] and the Okamoto-Takashima (OT11) scheme [24]. Table 1 shows a brief comparison among the existing schemes and our scheme. In the table the three MPR11 schemes (1)–(3) are respectively the Boneh-Boyen signature based scheme, the Waters signature based scheme, and the scheme proven secure in the generic group model. For the first two schemes, we show the performance in the SXDH setting. Our scheme is instantiated with the Kiltz-Pan-Wee structure-preserving signature scheme from the SXDH assumption [16] and the Groth-Sahai proof system in the SXDH setting [14]. We also note that all the five schemes in the table are instantiated in prime order groups.

Table 2 shows a detailed calculation of the signature size of our scheme. The center and right columns respectively show the number of the group elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ that is required for each component of a signature.

Table 1. Comparison among pairing-based attribute-based signature schemes.

Scheme	Signature size	Assumption	Predicate
MPR11 (1) [21]	$36s + 2t + 24ks$	q-SDH, SXDH	Monotone span program
MPR11 (2) [21]	$28s + 2t + 12k + 8$	SXDH	Monotone span program
MPR11 (3) [21]	$s + t + 2$	Generic group	Monotone span program
OT11 [24]	$9s + 11$	DLIN	Non-monotone span program
Ours	$12\ell + 20N + 26$	SXDH	Non-monotone circuit

k: The security parameter
$s \times t$: The size of the monotone span program
ℓ: The input length of the circuit
N: The number of the gate in the circuit

Table 2. Signature size of our scheme.

	$\mathbb{G}_1$	$\mathbb{G}_2$
com_θ	12	2
com_{W_i}	$2(\ell+N)$	
$\mathsf{com}_{\tilde{W}_i}$		$2(\ell+N)$
π_{Sign}	4	8
$\pi_1, \ldots, \pi_{\ell+1}$	$4(\ell+1)$	$4(\ell+1)$
$\pi_{(\ell+1)+1}, \ldots, \pi_{(\ell+1)+(N-1)}$	$8(N-1)$	$8(N-1)$
$\pi_{(\ell+1)+N}$	4	4
Total	$6\ell+10N+16$	$6\ell+10N+10$

ℓ: The input length of the circuit
N: The number of the gates in the circuit

As the table shows, our scheme achieves a comparable performance with the existing schemes, while the class of supported predicates is drastically wider than the existing schemes. In addition, the assumption from which the scheme is proven secure is also comparable with or in some case identical to the existing schemes.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
2. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014)
3. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology **21**(2), 149–177 (2008)
5. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity based encryption on trapdoor permutations. In: 49th Annual Symposium on Foundations of Computer Science, pp. 283–292. IEEE (2008)
6. Chen, C., Chen, J., Lim, H.W., Zhang, Z., Feng, D., Ling, S., Wang, H.: Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 50–67. Springer, Heidelberg (2013)
7. El Kaafarani, A., Ghadafi, E., Khader, D.: Decentralized traceable attribute-based signatures. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 327–348. Springer, Heidelberg (2014)
8. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 224–241. Springer, Heidelberg (2011)

9. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013)
10. Ghadafi, E.: Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16715-2
11. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, pp. 545–554. ACM (2013)
12. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015)
13. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM **59**(3), 11: 1–11: 35 (2012)
14. Groth, J., Sahai, A.: Efficient noninteractive proof systems for bilinear groups. SIAM J. Comput. **41**(5), 1193–1232 (2012)
15. Herranz, J., Laguillaumie, F., Libert, B., Ràfols, C.: Short attribute-based signatures for threshold predicates. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 51–67. Springer, Heidelberg (2012)
16. Kiltz, E., Pan, J., Wee, H.: Structure-preserving signatures from standard assumptions, revisited. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 275–295. Springer, Heidelberg (2015)
17. Kurosawa, K., Heng, S.-H.: From digital signature to ID-based identification/signature. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 248–261. Springer, Heidelberg (2004)
18. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 60–69. ACM (2010)
19. Li, J., Kim, K.: Attribute-based ring signatures. Cryptology ePrint Archive, Report 2008/394 (2008). http://eprint.iacr.org/
20. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-based signatures: Achieving attribute-privacy and collu sion-resistance. Cryptology ePrint Archive, Report 2008/328 (2008). http://eprint.iacr.org/
21. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011)
22. Maurer, U.M., Yacobi, Y.: Non-interative public-key cryptography. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 498–507. Springer, Heidelberg (1991)
23. Nandi, M., Pandit, T.: On the power of pair encodings: Frameworks for predicate cryptographic primitives. Cryptology ePrint Archive, Report 2015/955 (2015). http://eprint.iacr.org/
24. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011)
25. Okamoto, T., Takashima, K.: Decentralized attribute-based signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 125–142. Springer, Heidelberg (2013)
26. Shahandashti, S.F., Safavi-Naini, R.: Threshold attribute-based signatures and their application to anonymous credential systems. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (2009)

27. Tang, F., Li, H., Liang, B.: Attribute-based signatures for circuits from multilinear maps. In: Chow, S.S.M., Camenisch, J., Hui, L.C.K., Yiu, S.M. (eds.) ISC 2014. LNCS, vol. 8783, pp. 54–71. Springer, Heidelberg (2014)
28. Wang, Q., Chen, S.: Attribute-based signature for threshold predicates from lattices. Secur. Commun. Netw. **8**(5), 811–821 (2015)
29. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys

Nils Fleischhacker(✉), Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin

CISPA, Saarland University, Saarbrücken, Germany
fleischhacker@cs.uni-saarland.de

Abstract. In a sanitizable signature scheme the signer allows a designated third party, called the sanitizer, to modify certain parts of the message and adapt the signature accordingly. Ateniese et al. (ESORICS 2005) introduced this primitive and proposed five security properties which were formalized by Brzuska et al. (PKC 2009). Subsequently, Brzuska et al. (PKC 2010) suggested an additional security notion, called unlinkability which says that one cannot link sanitized message-signature pairs of the same document. Moreover, the authors gave a generic construction based on group signatures that have a certain structure. However, the special structure required from the group signature scheme only allows for inefficient instantiations.

Here, we present the first efficient instantiation of unlinkable sanitizable signatures. Our construction is based on a novel type of signature schemes with re-randomizable keys. Intuitively, this property allows to re-randomize both the signing and the verification key separately but consistently. This allows us to sign the message with a re-randomized key and to prove in zero-knowledge that the derived key originates from either the signer or the sanitizer. We instantiate this generic idea with Schnorr signatures and efficient Σ-protocols, which we convert into non-interactive zero-knowledge proofs via the Fiat-Shamir transformation. Our construction is at least one order of magnitude faster than instantiating the generic scheme of Brzuska et al. with the most efficient group signature schemes.

1 Introduction

Sanitizable signature schemes were introduced by Ateniese et al. [1] and similar primitives were concurrently proposed by Steinfeld et al. [42], by Miyazaki et al. [36], and by Johnson et al. [34]. The basic idea of this primitive is that the signer specifies parts of a (signed) message such that a dedicated third party, called the sanitizer, can change the message and adapt the signature accordingly. Sanitizable signatures have numerous applications, such as the anonymization of medical data, replacing commercials in authenticated media streams, or updates of reliable routing information [1]. After the first introduction of sanitizable signatures in [1], the desired security properties were later formalized by

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 301–330, 2016.
DOI: 10.1007/978-3-662-49384-7_12

Brzuska et al. [11]. At PKC 2010, Brzuska et al. [12] identified an important missing property called unlinkability. Loosely speaking, this notion ensures that one cannot link sanitized message-signature pairs of the same document. This property is essential in applications like the sanitization of medical records because it prevents the attacker from combining information of several sanitized versions of a document in order to reconstruct (parts of) the original document. The authors also showed that unlinkable sanitizable signatures can be constructed from group signatures [4] having the property that the keys of the signers can be computed independently, and in particular before the keys of the group manager. However, to this date, no efficient group signature scheme *that has the required properties* is known, which also means that no efficient unlinkable sanitizable signature scheme is known. This leaves us in an unsatisfactory situation. Either we use efficient sanitizable signature schemes that only achieve a subset of the security properties [1,11] or we have to rely on an inefficient black-box construction of unlinkable sanitizable signatures.

In this work, we close this gap by presenting the first efficient unlinkable sanitizable signature scheme that achieves all security properties. The instantiation of our scheme only requires 15 exponentiations for signing, 17 for the verification, and 14 for sanitizing a message-signature pair. This is at least one order of magnitude faster than the fastest previously known construction. For a detailed performance comparison, refer to Sect. 1.2.

1.1 Overview of Our Construction

In this section, we describe the main idea of our construction and the underlying techniques. Our solution is based on a novel type of digital signature schemes called *signatures with perfectly re-randomizable keys*. This type of signature schemes allows to re-randomize both the signing and the verification key separately. It is required that the re-randomization is perfect, meaning that re-randomized keys must have the same distribution as the original key. The new unforgeability notion for this type of signature scheme requires that it is infeasible for an attacker to output a forgery under either the original or a re-randomized key, even if the randomness is controlled by the attacker.

We show that this notion does not trivially follow from the regular notion of unforgeability. In fact, only a few signature schemes having this property achieve our notion of unforgeability under re-randomizable keys. We demonstrate this fact by showing concrete attacks against some well known unforgeable signature schemes that have re-randomizable keys. In particular, we show that the signature scheme of Boneh and Boyen [6] and the one of Camenisch and Lysyanskaya [15] have re-randomizable keys, but are insecure with respect to our stronger security notion. We stress that these attacks have no implications on the original security proof, but that they cannot be used as an instantiation. On the positive side, we prove that Schnorr's signature scheme [40,41] has re-randomizable keys and fulfills our security notion. It is well known that Schnorr's signature scheme [40,41] is one of the most efficient signature schemes based on the discrete logarithm assumption. Moreover, we also propose an instantiation of

signature schemes with re-randomizable keys in the standard model by slightly modifying the signature scheme of Hofheinz and Kiltz [31,32].

Apart from their usefulness in constructing highly efficient sanitizable signatures, this primitive may also be of independent interest. A second possible application of signature schemes with re-randomizable keys are stealth addresses [27] in Bitcoin or other cryptocurrencies. On a very high level, Bitcoin replaces bank accounts with keys of a signature scheme. Money transactions in Bitcoin transfer money from one public key to another and are only valid if they are signed with the secret key of the payer. All transactions are logged in a public log data structure, the block chain, which can be used to verify the validity of new transactions as well as to track money flow in Bitcoin. Our signatures with re-randomizable keys provide a conceptually very simple solution for so called stealth addresses. Consider a Bitcoin donation address on a website to support the host of the website or donate money to the website for a good cause. A donor may be unwilling to donate money if he can be linked to the website or other donors by the block chain. Using signatures with re-randomizable keys a donor can take the donation address, re-randomize it, and pay the money to the re-randomized address and transmit the re-randomization factor to the recipient through a non-public channel, such as email. The recipient can use the given re-randomization factor to re-randomize his corresponding secret key to further transfer the received money. Such addresses that are related in some invisible way to the recipient are called stealth addresses. For a more detailed treatment of Bitcoin and the existing stealth address mechanism see [27].

Construction of Unlinkable Sanitizable Signature Schemes. Our construction is based on signature schemes that have perfectly re-randomizable keys. To sign a message m, the signer first splits the message into the parts that cannot be modified by the sanitizer and those that may be changed. Subsequently, the signer authenticates the entire messages using a signature scheme with re-randomized keys. However, the signer cannot sign this part directly as this would reveal the identity of the signer. Instead, the signer chooses a randomness ρ, re-randomizes their key-pair, and then proves, in zero-knowledge, that the derived public key is a re-randomization of either the signer's or the sanitizer's key.

Sanitizing a message follows the same idea: the sanitizer modifies the message and signs it with a re-randomized version of their key pair and appends a zero-knowledge proof for the same language.

To turn this idea into an efficient scheme, we propose an efficient sigma protocol tailored to our problem that we then convert via the Fiat-Shamir transformation [24] into an efficient non-interactive zero-knowledge proof. The main observation is that our zero-knowledge proofs prove only simple statements about the keys and *not* about encrypted signatures that verify under either the signer or the sanitizers public-key. Since the corresponding language is much simpler than this standard "encrypt-and-proof" approach, it has much shorter statements and thus the resulting zero-knowledge proofs are significantly more efficient.

1.2 Evaluation and Comparison

To demonstrate the efficiency of our approach, we compare both the computational and the storage complexity of our construction to the one of Brzuska et al. [12], where we use the currently most efficient instantiations of the underlying (group) signature scheme. Somewhat surprisingly, only a few group signature schemes have the property that the user keys can be generated independently of and, in particular, before the group manager's key — a property that is required by [12]. This property originates from the definitions of Bellare et al. [4] and only very few group signature schemes, such as [29,30], can be adapted to have this property and at the same time fulfill all security requirements needed in [12]. In most cases the group member's keys depend on some information published by the group manager. Finally, we instantiate the signature scheme in [12] using a deterministic version of Schnorr's signature scheme. Thus, in our comparison shown in Table 1, we instantiate [12] with the group signature schemes of Groth [30] and of Furukawa and Yonezawa [29], which are to the best of our knowledge the two most efficient group signature schemes that can be adapted to allow an instantiation of [12]. Our comparison shows that in the most important algorithms, i.e., signing, sanitizing, and verification, our construction is at least one order of magnitude faster than both instantiations of [12]. Similarly, Table 2 provides an overview of the storage complexity of the different constructions. Although our keys are slightly larger than the other instances, it also shows that our signatures are significantly smaller than the ones of the other instances. Note that both the number of exponentiations and the number of group elements for Furukawa and Yonezawa's group signature scheme depend linearly on the security parameter. In our comparison, the scheme is instantiated with 100 bit security.

Thus, it is easy to see that our solutions is the first scheme that is efficient enough to be used in practice today.

1.3 Related Work

Ateniese et al. [1] first introduced sanitizable signatures and gave an informal description of the following properties: *Unforgeability* ensures that only the honest signer and sanitizer can create valid signatures. *Immutability* says that the

Table 1. Comparison of the dominant operations in our construction instantiated as described in Sect. 5 with the construction of Brzuska et al. [12] instantiated with Schnorr signatures and the group signature schemes of Groth [30] and Furukawa and Yonezawa [29] respectively. E and P stand for group exponentiations and pairing evaluations respectively.

	KGen_{sig}	KGen_{san}	Sign	Sanit	Verify	Proof	Judge
This paper	7E	1E	15E	14E	17E	23E	6E
[12] using [30]	1E	1E	194E + 2P	186E + 1P	207E + 62P	14E + 1P	1E + 2P
[12] using [29]	1E	4E	2831E	2814E	2011E	18E	2E

Table 2. Comparison of the key, signature, and proof sizes in our construction instantiated as described in Sect. 5 with the construction of Brzuska et al. [12] instantiated with Schnorr signatures and the group signature schemes of Groth [30] and Furukawa and Yonezawa [29] respectively. Here pk_{sig}, sk_{sig}, pk_{san}, and sk_{san} refer to the signer's and sanitizer's public and secret keys, while σ refers to the signature, and π refers to the proof that is used to determine accountability. The sizes are measured in group elements. For the sake of simplicity we do not distinguish between elements of different groups such as $\mathbb{Z}_q$ and $\mathbb{G}$. This simplification slightly favors [12] using [30], since group signatures in this scheme consist exclusively of $\mathbb{G}$-elements.

	pk_{sig}	sk_{sig}	pk_{san}	sk_{san}	σ	π
This paper	7	14	1	1	14	4
[12] using [30]	1	1	1	1	69	1
[12] using [29]	1	1	5	1	1620	3

(malicious) sanitizer can only modify designated parts of the message. *Transparency* guarantees that signatures computed by the signer and the sanitizer are indistinguishable. *Accountability* demands that, with the help of the signer, a proof of authorship can be generated, such that neither the malicious signer nor the malicious sanitizer can deny authorship of the message. These properties were later formalized by Brzuska et al. [11] and the *Unlinkability* property was introduced by Brzuska et al. in [12]. Later, in [13], Brzuska et al. introduce the notion of non-interactive public accountability, which allows a third party, without help from the signer, to determine, whether a message originates from the signer or the sanitizer. In [14], the same authors provide a slightly stronger unlinkability notion and an instantiation that has non-interactive public accountability and achieves their new unlinkability notion. However, non-interactive accountability and transparency are mutually exclusive. That is, no scheme can fulfill both properties at the same time. In this work we focus on schemes that have (interactive) accountability and transparency. Another line of research initiated by Klonowski and Lauks [35] and continued by Canard and Jambert [16] considers different methods for limiting the allowed operations of the sanitizer. That is, they show how to limit the set of possible modifications on one single block and how to enforce the same modifications on different message blocks. In [17], Canard et al. extend sanitizable signatures to the setting with multiple signers and sanitizers. Recently, Derler and Slamanig suggested a security notion that is stronger than privacy but weaker than unlinkability [23].

Other closely related types of malleable signature schemes, such as homomorphic signatures [2,8,18,28,33,34] or redactable signatures [10,19,34,37,42], where parts of the signed message can be removed, are closely related to sanitizable signatures, but aim to solve related but different problems, have different security notions, and are not directly applicable to solve the problem of efficient unlinkable sanitizable signatures. In [5] Boldyreva et al. deal with proxy signature schemes for delegating signing rights. In such signature schemes a designator can delegate signing rights to a proxy signer, who can then sign messages

on behalf of the designator. However, in such a scheme the proxy signatures are publicly distinguishable from signatures created by the designator. This would break the transparency property of sanitizable signature schemes. Policy-based signatures [3] allows a signer to delegate signing rights in connection with a policy that specifies, which messages can be signed with the delegated signing key. In addition, they require that they delegation policy shall remain hidden. In a similar vein to [3] in [9] the authors explore the possibilities of delegating signing keys for arbitrary functions. That is, using the delegated signing key one can sign functions of the message that correspond to the key. These works show theoretical solutions to the discussed problems, but are too slow for practical use due to the cryptographic tools they use.

2 Sanitizable Signatures

Sanitizable signature schemes allow the delegation of signing capabilities to a designated third party, called the sanitizer. These delegation capabilities are realized by letting the signer "attach" a description of the admissible modifications $\textsc{Adm}$ for this particular message and sanitizer. The sanitizer may then change the message according to some modification $\textsc{Mod}$ and update the signature using their private key. More formally, the signer holds a key pair $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig})$ and signs a message m and the description of the admissible modifications $\textsc{Adm}$ for some sanitizer pk_{san} with its private key sk_{sig}. The sanitizer, having a matching private key sk_{san}, can update the message according to some modification $\textsc{Mod}$ and compute a signature using his secret key sk_{san}. In case of a dispute about the origin of a message-signature pair, the signer can compute a proof π (using an algorithm Proof) from previously signed messages that proves that a signature has been created by the sanitizer. The verification of this proof is done by an algorithm Judge (that only decides the origin of a valid message-signature pair in question; for invalid pairs such decisions are in general impossible).

Admissible Modifications. Following [11,12] closely, we assume that $\textsc{Adm}$ and $\textsc{Mod}$ are (descriptions of) efficient deterministic algorithms such that $\textsc{Mod}$ maps any message m to the modified message $m' = \textsc{Mod}(m)$, and $\textsc{Adm}(\textsc{Mod}) \in \{0,1\}$ indicates if the modification is admissible and matches $\textsc{Adm}$, in which case $\textsc{Adm}(\textsc{Mod}) = 1$. By $\textsc{Fix}_{\textsc{Adm}}$ we denote an efficient deterministic algorithm that is uniquely determined by $\textsc{Adm}$ and which maps m to the immutable message part $\textsc{Fix}_{\textsc{Adm}}(m)$, e.g., for block-divided messages $\textsc{Fix}_{\textsc{Adm}}(m)$ is the concatenation of all blocks not appearing in $\textsc{Adm}$. We require that admissible modifications leave the fixed part of a message unchanged, i.e., $\textsc{Fix}_{\textsc{Adm}}(m) = \textsc{Fix}_{\textsc{Adm}}(\textsc{Mod}(m))$ for all $m \in \{0,1\}^*$ and all $\textsc{Mod}$ with $\textsc{Adm}(\textsc{Mod}) = 1$. Analogously, to avoid choices like $\textsc{Fix}_{\textsc{Adm}}$ having empty output, we also require that the fixed part must be "maximal" given $\textsc{Adm}$, i.e., $\textsc{Fix}_{\textsc{Adm}}(m') \neq \textsc{Fix}_{\textsc{Adm}}(m)$ for $m' \notin \{\textsc{Mod}(m) \mid \textsc{Mod} \text{ with } \textsc{Adm}(\textsc{Mod}) = 1\}$.

2.1 Definition of Sanitizable Signatures

The following definition of sanitizable signature schemes is taken in verbatim from [11,12].

Definition 1 (Sanitizable Signature Scheme). *A sanitizable signature scheme* $\mathrm{SanS} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *consists of seven algorithms:*

Key Generation. *There are two key generation algorithms, one for the signer and one for the sanitizer. Both create a pair of keys, a private and the corresponding public key:*

$$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa) \qquad \textit{and} \qquad (\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa).$$

Signing. *The signing algorithm takes as input a message* $m \in \{0,1\}^*$*, a signer secret key* sk_{sig}*, a sanitizer public key* pk_{san}*, as well as a description* Adm *of the admissible modifications to* m *by the sanitizer and outputs a signature* σ*. We assume that* Adm *can be recovered from any signature:*

$$\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \mathrm{Adm}).$$

Sanitizing. *The sanitizing algorithm takes as input a message* $m \in \{0,1\}^*$*, a description* Mod *of the desired modifications to* m*, a signature* σ*, the signer's public key* pk_{sig}*, and a sanitizer secret key* sk_{san}*. It modifies the message* m *according to the modification instruction* Mod *and outputs a new signature* σ' *for the modified message* $m' = \mathrm{Mod}(m)$ *or possibly* $\perp$ *in case of an error:*

$$\{(m', \sigma'), \perp\} \leftarrow \mathsf{Sanit}(m, \mathrm{Mod}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san}).$$

Verification. *The verification algorithm takes as input a message* m*, a candidate signature* σ*, a signer public key* pk_{sig}*, as well as a sanitizer public key* pk_{san} *and outputs a bit* b*:*

$$b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}).$$

Proof. *The proof algorithm takes as input a signer secret key* sk_{sig}*, a message* m*, a signature* σ*, and a sanitizer public key* pk_{san} *and outputs a proof* π*:*

$$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m, \sigma, \mathsf{pk}_{san}).$$

Judge. *The judge algorithm takes as input a message* m*, a signature* σ*, signer and sanitizer public keys* $\mathsf{pk}_{sig}, \mathsf{pk}_{san}$*, and proof* π*. It outputs a decision* $d \in \{\mathtt{Sign}, \mathtt{San}\}$ *indicating whether the message-signature pair was created by the signer or the sanitizer:*

$$d \leftarrow \mathsf{Judge}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi).$$

For a sanitizable signature scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted and that a genuinely created proof by the signer leads the judge to decide in favor of the signer. For a formal approach to correctness see [11].

2.2 Security of Sanitizable Signatures

In this section we recall the security notions of sanitizable signatures given by Brzuska et al. [11,12] and we follow their description closely. The authors defined unforgeability, privacy, immutability, accountability, transparency, and unlinkability and showed that signer and sanitizer accountability together implies unforgeability and that unlinkability implies privacy. Therefore, we only focus on the necessary definitions and omit unforgeability and privacy.

Immutability. Informally, this property says that a malicious sanitizer cannot change inadmissible blocks. This is formalized in a model where the malicious sanitizer $\mathcal{A}$ interacts with the signer to obtain signatures σ_i for messages m_i, descriptions ADM_i and keys $\mathsf{pk}_{san,i}$ of its choice. Eventually, the attacker stops, outputting a valid pair $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$ such that message m^* is not a "legitimate" transformation of one of the m_i's under the same key $\mathsf{pk}^*_{san} = \mathsf{pk}_{san,i}$. The latter is formalized by requiring that for each query $\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i}$ or $m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}_i(\text{MOD}) = 1\}$ for the value ADM_i in σ_i. This requirement enforces that for block-divided messages m^* and m_i differ in at least one inadmissible block. Observe that this definition covers also the case where the adversary interact with several sanitizers simultaneously, because it can query the signer for several sanitizer keys $\mathsf{pk}_{san,i}$.

Definition 2 (Immutability). *A sanitizable signature scheme* SanS *is said to be* immutable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligible (in κ), where*

Experiment $\mathsf{Immut}^{\text{SanS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
$(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
letting $(m_i, \text{ADM}_i, \mathsf{pk}_{san,i})$ *and* σ_i *denote the queries and answers to and from oracle* Sign.
Output 1 if $\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1$ and *for all i the following holds:* $\mathsf{pk}^*_{san} \neq \mathsf{pk}_{san,i}$ *or* $m^* \notin \{\text{MOD}(m_i) \mid \text{MOD with } \text{ADM}_i(\text{MOD}) = 1\}$
Else output 0.

Accountability. This property demands that the origin of a (possibly sanitized) signature should be undeniable. We distinguish between *sanitizer-accountability* and *signer-accountability* and formalize each security property in the following. *Signer-accountability* says that, if a message and its signature have not been sanitized, then even a malicious signer should not be able to make the judge accuse the sanitizer.

In the sanitizer-accountability game let $\mathcal{A}_{\mathsf{Sanit}}$ be an efficient adversary playing the role of the malicious sanitizer. Adversary $\mathcal{A}_{\mathsf{Sanit}}$ has access to a Sign and Proof oracle and it succeeds if it outputs a valid message signature pair such that m^*, σ^*, together with a key pk^*_{san} (with $(\mathsf{pk}^*_{san}, m^*)$ such that the output is different from pairs $(\mathsf{pk}_{san,i}, m_i)$ previously queried to the Sign oracle). Moreover, it is required that the proof produced by the signer via Proof still leads the judge to decide "Sign", i.e., that the signature has been created by the signer.

Definition 3 (Sanitizer-Accountability). *A sanitizable signature scheme* SanS *is* sanitizer-accountable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment* $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in κ), where*

Experiment $\mathsf{San\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^{\kappa})$
$(\mathsf{pk}^*_{san}, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig})$
letting $(m_i, \textsc{Adm}_i, \mathsf{pk}_{san,i})$ *and* σ_i *denote the queries and answers to and from oracle* Sign
$\pi \leftarrow \mathsf{Proof}(\mathsf{sk}_{sig}, m^*, \sigma^*, \mathsf{pk}^*_{san})$
Output 1 if for all i the following holds:
$(\mathsf{pk}^*_{san}, m^*) \neq (\mathsf{pk}_{san,i}, m_i)$ and
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}) = 1$ and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}, \pi) \neq \mathtt{San}$

In the signer-accountability game a malicious signer $\mathcal{A}_{\mathsf{Sign}}$ gets a public sanitizing key pk_{san} as input and has access to a sanitizing oracle, which takes as input tuples $(m_i, \textsc{Mod}_i, \sigma_i, \mathsf{pk}_{sig,i})$ and returns (m'_i, σ'_i). Eventually, the adversary $\mathcal{A}_{\mathsf{Sign}}$ outputs a tuple $(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*)$ and is considered successful if Judge accuses the sanitizer for the new key-message pair pk^*_{sig}, m^* with a valid signature σ^*.

Definition 4 (Signer-Accountability). *A sanitizable signature scheme* SanS *is said to be* signer-accountable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment* $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in κ), where*

Experiment $\mathsf{Sig\text{-}Acc}^{\mathrm{SanS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^n)$
$(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san})}(\mathsf{pk}_{san})$
letting $(m_i, \textsc{Mod}_i, \sigma_i, \mathsf{pk}_{sig,i})$ *and* (m'_i, σ'_i) *denote the queries and answers to and from oracle* Sanit.
Output 1 if for all i the following holds:
$(\mathsf{pk}^*_{sig}, m^*) \neq (\mathsf{pk}_{sig,i}, m'_i)$ and
$\mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}) = 1$ and
$\mathsf{Judge}(m^*, \sigma^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}, \pi^*) \neq \mathtt{Sign}$
else output 0.

Transparency. Informally, this property says that one cannot decide whether a signature has been sanitized or not. Formally, this is defined in a game where an adversary $\mathcal{A}$ has access to Sign, Sanit, and Proof oracles with which the adversary can create signatures for (sanitized) messages and learn proofs. In addition, $\mathcal{A}$ gets access to a $\mathsf{Sanit}/\mathsf{Sign}$ box which contains a secret random bit $b \in \{0, 1\}$ and which, on input a message m, a modification information $\textsc{Mod}$ and a description $\textsc{Adm}$ behaves as follows:

- for $b = 0$ runs the signer algorithm to create $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{sig}, \text{ADM})$, then runs the sanitizer algorithm and returns the sanitized message m' with the new signature σ', and
- for $b = 1$ acts as in the case $b = 0$ but also signs m' from scratch with the signing algorithm to create a signature σ' and returns the pair (m', σ').

Adversary $\mathcal{A}$ eventually produces an output a, the guess for b. A sanitizable signature is now *transparent* if for all efficient algorithms $\mathcal{A}$ the probability for a right guess $a = b$ in the above game is negligibly close to $\frac{1}{2}$. Below we also define a relaxed version called *proof-restricted transparency*.

Definition 5 ((Proof-Restricted) Transparency). *A sanitizable signature scheme* SanS *is said to be* proof-restrictedly transparent *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment $\mathsf{Trans}^{\text{SanS}}_{\mathcal{A}}(\kappa)$ evaluates to 1 is negligibly bigger than $1/2$ (in κ), where*

Experiment $\mathsf{Trans}^{\text{SanS}}_{\mathcal{A}}(\kappa)$
$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
$(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa)$
$b \leftarrow \{0, 1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\ \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{Sanit/Sign}(\cdot,\cdot,\cdot)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
letting $M_{\mathsf{Sanit/Sign}}$ and M_{Proof} denote the sets of messages output by the Sanit/Sign *and queried to the* Proof *oracle respectively.*
Output 1 if $\big(a = b$ and $M_{\mathsf{Sanit/Sign}} \cap M_{\mathsf{Proof}} = \emptyset\big)$
Else output 0

Unlinkability. This security notion demands that it is not feasible to use the signatures to identify sanitized message-signature pairs originating from the same source. This should even hold if the adversary itself provides the two source message-signature pairs and modifications of which one is sanitized. It is required that the two modifications yield the same sanitized message, because otherwise predicting the source is easy, of course. This, however, is beyond the scope of signature schemes: the scheme should only prevent that *signatures* can be used to link data. In the formalization of [12], the adversary is given access to a signing oracle and a sanitizer oracle (and a proof oracle since this step depends on the signer's secret key and may leak valuable information). The adversary is also allowed to query a left-or-right oracle LoRSanit which is initialized with a secret random bit b and keys $\mathsf{pk}_{sig}, \mathsf{sk}_{san}$. The adversary may query this oracle on tuples $((m_0, \text{MOD}_0, \sigma_0), (m_1, \text{MOD}_1, \sigma_1))$ and returns $\mathsf{Sanit}(m_b, \text{MOD}_b, \sigma_b, \mathsf{pk}_{sig}, \mathsf{sk}_{san})$ if $\mathsf{Verify}(m_i, \sigma_i, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 1$ for $i = 0, 1$, $\text{ADM}_0 = \text{ADM}_1$ and if the modifications map to the same message, i.e., $\text{ADM}_0(\text{MOD}_0) = 1$, $\text{ADM}_1(\text{MOD}_1) = 1$ and $\text{MOD}_0(m_0) = \text{MOD}_1(m_1)$. Otherwise, the oracle returns $\perp$. The adversary should eventually predict the bit b significantly better than with the guessing probability of $\frac{1}{2}$.

Definition 6 (Unlinkability). *A sanitizable signature scheme* SanS *is* unlinkable *if for all PPT adversaries $\mathcal{A}$ the probability that the experiment* $\mathsf{Link}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$ *evaluates to 1 is negligibly bigger than* $1/2$ *(in* κ*), where*

Experiment $\mathsf{Link}_{\mathcal{A}}^{\mathrm{SanS}}(\kappa)$
$(\mathsf{sk}_{sig}, \mathsf{pk}_{sig}) \leftarrow \mathsf{KGen}_{sig}(1^\kappa)$
$(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{KGen}_{san}(1^\kappa)$
$b \leftarrow \{0,1\}$
$a \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\mathsf{sk}_{sig},\cdot,\cdot),\mathsf{Sanit}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_{san}),\ \mathsf{Proof}(\mathsf{sk}_{sig},\cdot,\cdot,\cdot),\mathsf{LoRSanit}(\cdot,\cdot)}(\mathsf{pk}_{sig}, \mathsf{pk}_{san})$
if $a = b$ *then output* 1*, else output* 0.

3 Signatures Schemes with Re-randomizable Keys

In this section, we introduce signature schemes that have re-randomizable keys and which serve as the main building block for our construction. Signature schemes with this property have the advantage that one can re-randomize the key-pair $(\mathsf{sk}, \mathsf{pk})$ to a key-pair $(\mathsf{sk}', \mathsf{pk}')$ and sign a message m with a seemingly unrelated key. Jumping ahead, this property allows us to sign messages with a fresh key and prove, in zero-knowledge, the origin of the key. For one of the signature schemes we require bilinear maps, which are defined as follows. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ be an efficient, non-degenerate bilinear map, for system-wide available groups, where g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

3.1 Defining Signature Schemes with Re-randomizable Keys

To define this property and the corresponding security notion formally, we denote by $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify})$ a standard digital signature scheme, where $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\kappa), (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa), \sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m), b \leftarrow \mathsf{SVerify}(\mathsf{pk}, m, \sigma)$ are the standard algorithms of a digital signature scheme.

Definition 7 (Signatures with Perfectly Re-randomizable Keys). *A signature scheme* $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify})$ *has perfectly re-randomizable keys if there exist two PPT algorithms* $(\mathsf{RandSK}, \mathsf{RandPK})$ *and a randomness space* χ *such that:*

$\mathsf{RandSK}(\mathsf{sk}, \rho)$: *The secret key re-randomization algorithm takes as input a secret key* sk *and a randomness* $\rho \in \chi$ *and outputs a new secret key* sk'.

$\mathsf{RandPK}(\mathsf{pk}, \rho)$: *The public key re-randomization algorithm takes as input a public key* pk *and a randomness* $\rho \in \chi$ *and outputs a new public key* pk'.

Correctness. *The scheme is* correct *if and only if all of the following holds:*

1. *For all* $\kappa \in \mathbb{N}$*, all key-pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all messages* $m \in \{0,1\}^*$*, and all signatures* $\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m)$*, it holds that* $\mathsf{SVerify}(\mathsf{pk}, m, \sigma) = 1$.

2. *For all* $\kappa \in \mathbb{N}$*, all key-pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$*, all randomness* $\rho \in \chi$*, all messages* $m \in \{0,1\}^*$*, and* $\sigma \leftarrow \mathsf{SSign}(\mathsf{RandSK}(\mathsf{sk}, \rho), m)$*, it holds that* $\mathsf{SVerify}(\mathsf{RandPK}(\mathsf{pk}, \rho), m, \sigma) = 1$.
3. *For all key pairs* $(\mathsf{sk}, \mathsf{pk})$*, and a uniformly chosen randomness* $\rho \in \chi$*, the distribution of* $(\mathsf{sk}', \mathsf{pk}')$ *and* $(\mathsf{sk}'', \mathsf{pk}'')$ *is identical, where* $\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$*,* $\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$*, and* $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{SGen}(1^\kappa)$

3.2 Security of Signature Schemes with Re-randomizable Keys

The security of signature scheme with re-randomizable keys is defined analogously to the unforgeability of regular signature schemes, but allows the adversary to learn message/signature pairs under re-randomized keys. This should even hold if the randomness to re-randomize the keys is chosen by the attacker. In this definition, the adversary has access to two oracles. The first one, denoted by $\mathcal{O}_1$ is a regular signing oracle. The second one, denoted by $\mathcal{O}_2$ is an oracle that takes as input a message m and some randomness ρ. It then re-randomizes the private key according to ρ and signs the message using this key.

Definition 8 (Unforgeability under Re-randomized Keys). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* unforgeable under re-randomized keys *(*UFRK*) if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where*

Experiment $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$:
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
$Q := \emptyset$
$(m^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot), \mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
Output 1 if one of the two conditions is fulfilled
1. *If* $\mathsf{SVerify}(\mathsf{pk}, m^*, \sigma^*) = 1$ *and* $m^* \notin Q$
2. *If* $\mathsf{SVerify}(\mathsf{RandPK}(\mathsf{pk}, \rho^*), m^*, \sigma^*) = 1$ *and* $m^* \notin Q$

else output 0

$\mathcal{O}_1(\mathsf{sk}, m)$:
$Q := Q \cup \{m\}$
$\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}, m)$
output σ

$\mathcal{O}_2(\mathsf{sk}, m, \rho)$:
$Q := Q \cup \{m\}$
$\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
$\sigma \leftarrow \mathsf{SSign}(\mathsf{sk}', m)$
output σ

Given this definition of unforgeability, one can easily obtain the "standard" notion of existential unforgeability by giving the adversary only access to $\mathcal{O}_1$ and only checking the first condition.

Definition 9 (Existential Unforgeability). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is said to be* existentially unforgeable under chosen message attacks *(*EUF*) if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *is defined as* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$*, but the adversary only gets access to* $\mathcal{O}_1$ *and wins if the first condition is fulfilled.*

For our construction, we also need signature schemes that are strongly unforgeable, meaning that it is computationally hard to compute a *new* signature σ^* on a message m, i.e., the adversary is allowed to submit m to the oracle and learn a signature σ and wins the game if σ^* is valid but different from σ.

Definition 10 (Strong Existential Unforgeability). *A signature scheme with perfectly re-randomizable keys* $\Sigma = (\mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is* strongly existentially unforgeable under chosen message attacks *(*$\mathsf{s\text{-}EUF}$*) if for all PPT adversaries* $\mathcal{A}$ *the probability that the experiment* $\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligible (in* κ*), where* $\mathsf{s\text{-}EUF}^{\Sigma}_{\mathcal{A}}(\kappa)$ *is defined as* $\mathsf{UFRK}^{\Sigma}_{\mathcal{A}}(\kappa)$*, but the adversary only gets access to* $\mathcal{O}_1$ *and* $\mathcal{O}_1$ *maintains* $Q := Q \cup \{m, \sigma\}$*. The adversary wins only if the following condition is fulfilled:* $\mathsf{SVerify}(\mathsf{pk}, m^*, \sigma^*) = 1$ *and* $(m^*, \sigma^*) \notin Q$*.*

3.3 Counter Examples

In this section, we show that unforgeability under re-randomizable keys (Definition 8) does not trivially follow from regular unforgeability (Definition 9). In fact, very few standard model signatures, that have re-randomizable keys, are unforgeable under re-randomizable keys. We demonstrate this by giving concrete attacks against some well known schemes, such as the Boneh and Boyen [7] and Camenisch and Lysyanskaya [15] signature schemes. We remark that these attacks have no implications on the original security proof and that our attacks are outside of the regular unforgeability model.

Boneh-Boyen Signature Scheme. The scheme of Boneh and Boyen [7] works in a bilinear groups setting and is existentially unforgeable under the q-SDH assumption. The scheme works as follows: The secret key consists of $x, y \in \mathbb{Z}_q^*$ and the public key consists of the corresponding $\mathbb{G}_2$ elements $u := g_2^x$ and $v := g_2^y$. To sign a message $m \in \mathbb{Z}_q^*$, the signer chooses a random $r \leftarrow \mathbb{Z}_q^*$, computes $s := g_1^{1/(x+m+yr)}$, and outputs the signature $\sigma = (r, s)$. To verify that a signature is valid, the verifier checks that $e(s, u \cdot g_2^m \cdot v^r) = e(g_1, g_2)$ holds. The keys of the scheme can be re-randomized additively, i.e., given randomness $(\rho_1, \rho_2) \in \mathbb{Z}_q^2$, secret keys are randomized as $(x', y') := (x + \rho_1, y + \rho_2)$ and public keys are randomized as $(u', v') := (u \cdot g_2^{\rho_1}, v \cdot g_2^{\rho_2})$.

Even though this scheme is existentially unforgeable under the q-SDH assumption and has perfectly re-randomizable keys, it is forgeable under re-randomized keys. The attack is as follows: The adversary $\mathcal{A}$ on input the public key (u, v) chooses a random message $m \in \mathbb{Z}_q^*$ as well as a random value $\rho_1 \in \mathbb{Z}_q^*$. It then queries $(m, (\rho_1, 0))$ to its signing oracle receiving back a signature $\sigma = (r, s)$. Then, it computes $m' := m + \rho_1$ and outputs $\sigma, m', (0, 0)$ as a forgery. It is easy to verify, that the verification equation actually holds for the output of $\mathcal{A}$:

$$
\begin{aligned}
& e(s, u \cdot g_2^{m'} \cdot v^r) = e(g_1, g_2) \\
\Leftrightarrow \quad & e(s, g_2^{x+m+\rho_1+yr}) = e(g_1, g_2) \\
\Leftrightarrow \quad & e(g_1^{\frac{1}{(x+\rho_1)+m+yr}}, g_2^{x+\rho_1+m+yr}) = e(g_1, g_2) \\
\Leftrightarrow \quad & e(g_1, g_2)^{\frac{x+\rho_1+m+yr}{x+\rho_1+m+yr}} = e(g_1, g_2) \\
\Leftrightarrow \quad & e(g_1, g_2) = e(g_1, g_2)
\end{aligned}
$$

Furthermore, the adversary is efficient and the only message queried to the signing oracle is m, and $m' \neq m$. Therefore, it follows that $\mathcal{A}$ breaks the unforgeability under re-randomizable keys with probability 1.

Camenisch-Lysyanskaya Signature Scheme. The signature scheme of Camenisch and Lysyanskaya [15] works in a symmetric bilinear groups setting and is existentially unforgeable under the LRSW assumption. The scheme works as follows: The secret key consists of $x, y \in \mathbb{Z}_q$ and the public key consists of the corresponding group elements $X := g^x$ and $Y := g^y$. To sign a message $m \in \mathbb{Z}_q$, the signer chooses a random $a \leftarrow \mathbb{G}$, computes $b := a^y$ and $c := a^{x+mxy}$, and outputs the signature $\sigma = (a, b, c)$. To verify that a signature is valid, the verifier checks that $e(a, Y) = e(g, b)$ and $e(X, a) \cdot e(X, b)^m = e(g, c)$ hold. The keys of the scheme can be re-randomized multiplicatively[1]. I.e., given randomness $(\rho_1, \rho_2) \in \mathbb{Z}_q^2$, secret keys are randomized as $(x', y') := (x \cdot \rho_1, y \cdot \rho_2)$ and public keys are randomized as $(X', Y') := (X^{\rho_1}, Y^{\rho_2})$.

This scheme is also existentially unforgeable and has perfectly re-randomizable keys. Nevertheless it also is forgeable under re-randomized keys and the corresponding attack works as follows: The adversary $\mathcal{A}$ on input the public key (X, Y) chooses a random message $m \in \mathbb{Z}_q^*$ as well as a random value $\rho_2 \in \mathbb{Z}_q^* \backslash \{1\}$. It then queries $(m, (1, \rho_2))$ to its signing oracle receiving back a signature $\sigma = (a, b, c)$. It it finally computes $m' := m \cdot \rho_2$ and $b' := b^{(\rho_2^{-1})}$ and outputs $(a, b', c), m', (1, 1)$ as a forgery. It is easy to verify, that the verification equation actually holds for the output of $\mathcal{A}$. For the first check equation we have:

$$
\begin{aligned}
& e(a, Y) = e(g, b') \\
\Leftrightarrow \quad & e(a, g^y) = e(g, b^{(\rho_2^{-1})}) \\
\Leftrightarrow \quad & e(g^y, a) = e(g, a^{(y\rho_2)\cdot\rho_2^{-1}}) \\
\Leftrightarrow \quad & e(g^y, a) = e(g, a^y) \\
\Leftrightarrow \quad & e(g, a)^y = e(g, a)^y.
\end{aligned}
$$

[1] The keys can also be re-randomized additively, however in that case neither a proof of security nor an attack are apparent.

For the second verification equation we have:

$$
\begin{aligned}
& e(X,a)\cdot e(X,b')^{m'} = e(g,c) \\
\Leftrightarrow\quad & e(g^x,a)\cdot e(g^x,b^{\rho_2^{-1}})^{m\cdot\rho_2} = e(g,a^{x+mxy\rho_2}) \\
\Leftrightarrow\quad & e(g,a)^x\cdot e(g^x,a^{y\rho_2\rho_2^{-1}})^{m\rho_2} = e(g,a)^{x+mxy\rho_2} \\
\Leftrightarrow\quad & e(g,a)^x\cdot e(g,a)^{mxy\rho_2} = e(g,a)^{x+mxy\rho_2} \\
\Leftrightarrow\quad & e(g,a)^{x+mxy\rho_2} = e(g,a)^{x+mxy\rho_2}.
\end{aligned}
$$

Furthermore, the adversary is efficient and the only message queried to the signing oracle is m, and $m' \neq m$, since $\rho_2 \neq 1$. Therefore, it follows that $\mathcal{A}$ wins the unforgeability game with re-randomizable keys with probability 1.

3.4 Instantiations

In this section, we show that our security notion is achievable in the random oracle and the standard model. In the random oracle model, we prove that Schnorr's signature scheme [40,41] is unforgeable under re-randomized keys and in the standard model we show that a slightly modified version of the signature scheme due to Hofheinz and Kiltz [31,32] satisfies our notion.

Random Oracle Model. We show that Schnorr's signature scheme [40,41] is unforgeable under re-randomized keys. Our proof technique relies on an idea that was previously observed by Fischlin and Fleischhacker [25] in the context of an impossibility result. The core of this technique, that we call randomness switching technique, allows moving a signature from one public key to another one knowing only the difference between the two corresponding secret keys.

Definition 11 (Schnorr Signature Scheme). *Let $\mathbb{G}$ be a cyclic group of prime order q with generator g and let $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. The Schnorr signature scheme* SSS*, working over $\mathbb{G}$, is defined as follows:*

$\mathsf{SGen}(1^\kappa)$: *Pick* $\mathsf{sk} \leftarrow \mathbb{Z}_q$ *at random, compute* $\mathsf{pk} := g^{\mathsf{sk}}$*, and output* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{SSign}(\mathsf{sk}, m)$: *Pick* $r \leftarrow \mathbb{Z}_q$ *at random and compute* $R := g^r$*, compute* $c := \mathcal{H}(R, m)$ *and* $y := r + \mathsf{sk}\cdot c \mod q$*. Output* $\sigma := (c, y)$.

$\mathsf{SVerify}(\mathsf{pk}, m, \sigma)$: *Parse* σ *as* (c, y)*. If* $c = \mathcal{H}(\mathsf{pk}^{-c} g^y, m)$*, then output* 1*, otherwise output* 0.

$\mathsf{RandSK}(\mathsf{sk}, \rho)$: *Compute* $\mathsf{sk}' := \mathsf{sk} + \rho \mod q$ *and output* sk'.

$\mathsf{RandPK}(\mathsf{pk}, \rho)$: *Compute* $\mathsf{pk}' := \mathsf{pk}\cdot g^\rho$ *and output* pk'.

Obviously all three correctness conditions hold. It remains to show that SSS is unforgeable under re-randomized keys.

Theorem 1 (Unforgeability of Schnorr Signatures Under Re-randomized Keys). *The signature scheme* SSS *(Definition 11) is unforgeable under re-randomized keys (Definition 8) in the random oracle model if the discrete logarithm problem in $\mathbb{G}$ is hard.*

Proof. Assume towards contradiction that there exists an efficient adversary $\mathcal{A}$ against the unforgeability under re-randomized keys. Then, we construct an adversary $\mathcal{B}$ against the existential unforgeability of SSS, which runs $\mathcal{A}$ as a black-box and simulates both oracles with its own signing oracle. More precisely, $\mathcal{B}$ answers all queries to $\mathcal{O}_1(\mathsf{sk}, m)$ with its own signing oracle and it simulates $\mathcal{O}_2(\mathsf{sk}, \rho, m)$ by first querying its own signing oracle on m, obtaining a signature (c, y), and then adapting the signatures by adding the value $\rho \cdot c$ to y. Eventually, the adversary $\mathcal{A}$ outputs a forgery (σ^*, m^*, ρ^*) with $\sigma^* = (c, y)$. The reduction $\mathcal{B}$ adapts the signature in order to serve as a forgery under the key pk by subtracting $\rho^* \cdot c$ from y. A formal description of the adversary and the simulation of the oracle $\mathcal{O}_2(\mathsf{sk}, \rho, m)$ is given in the following:

$\underline{\mathcal{B}^{\mathcal{O}_1(\mathsf{sk},\cdot)}(\mathsf{pk}):}$
$(\sigma^*, m^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
Parse σ^* as (c, y)
$y' := y - \rho^* c$
output $(c, y'), m^*$

$\underline{\mathcal{O}_2(\mathsf{sk}, \rho, m):}$
$(c, y) \leftarrow \mathcal{O}_1(\mathsf{sk}, m)$
$y' := y + \rho c$
output (c, y')

For the analysis, let us assume that $\mathcal{A}$'s success probability in the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}$ is greater than $1/\mathsf{poly}(\kappa)$. It is easy to see that $\mathcal{B}$ is efficient and that the simulation of $\mathcal{A}$'s signing oracle $\mathcal{O}_1$ is perfect. Now, we show that $\mathcal{B}$ also provides a perfect simulation of the oracle $\mathcal{O}_2$. The signature under pk received by $\mathcal{O}_2$ consists of c and y. The c value is independent of the signing key, therefore only the y value needs to be adapted. The adapted value is computed as

$$y' = y + \rho c = r + \mathsf{sk} \cdot c + \rho c = r + (\mathsf{sk} + \rho) \cdot c.$$

Obviously (c, y') is therefore a signature on m under $\mathsf{pk} \cdot g^{\rho}$ with the same randomness as (c, y). It follows that the answers to signing queries are distributed exactly as in the $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment.

Similarly the output of $\mathcal{B}$ is computed from the output of $\mathcal{A}$. Whenever $\mathcal{A}$ outputs a valid signature, message, randomness triple (σ^*, m^*, ρ^*), we have that $\sigma^* = (c, y)$ where $c = \mathcal{H}(g^r, m)$ and $y = r + (\mathsf{sk} + \rho^*) \cdot c$ for some $r \in \mathbb{Z}_q$. We therefore have

$$y' := y - \rho^* c = r + (\mathsf{sk} + \rho^*) \cdot c - \rho^* c = r + \mathsf{sk} \cdot c$$

and thus (c, y') is a valid signature on m under pk. Further, in answering signing queries for $\mathcal{A}$, the adversary $\mathcal{B}$ queries the exact same messages as $\mathcal{A}$ and therefore whenever $\mathcal{A}$ wins in the $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment, $\mathcal{B}$ wins in the $\mathsf{EUF}_{\mathcal{A}}^{\mathsf{SSS}}(\kappa)$ experiment. Combining this with the well known proof of existential unforgeability of Schnorr signatures by Pointcheval and Stern [38,39] rules out the existence of $\mathcal{A}$ under the discrete logarithm assumption in the random oracle model.

Standard Model. In the following we show that a modified version of the signature schemes due to Hofheinz and Kiltz [31,32] is unforgeable under re-randomized keys. The original construction of Hofheinz and Kiltz works on type 1 and type 2 pairings and the element s in their scheme is a random bit string. However, in our case we choose s as a random element from $\mathbb{Z}_q$. This modification slightly increases the signature's size, but does not influence the original functionality or security proof. To prove the security formally, we adapt the randomness switching technique to this setting, which allows us to reduce the unforgeability under re-randomized keys to standard existential unforgeability. The scheme of Hofheinz and Kiltz requires a programmable hash function [31,32], but since security properties of programmable hash functions are not relevant to our proofs, we omit them here and refer the interested reader to [31,32].

Definition 12 (Programmable Hash Function [31,32]). *A programmable hash function* $(\mathsf{Gen}, \mathsf{Eval})$ *consists of two algorithms:*

$k \leftarrow \mathsf{Gen}(1^\kappa)$: *The key generation algorithm takes as input the security parameter* 1^κ *and generates a public key* k.

$y \leftarrow \mathsf{Eval}(k, m)$: *The deterministic evaluation algorithm takes as input a key* k *and a message* $m \in \{0,1\}^\ell$ *and outputs a hash value* y.

Given the definition of programmable hash functions, we define the slightly modified signature scheme due to Hofheinz Kiltz and define the re-randomization algorithms.

Definition 13 (Hofheinz Kiltz Signature Scheme [31,32]). *Let* $\mathsf{PHF} = (\mathsf{Gen}, \mathsf{Eval})$ *be a programmable hash function with domain* $\{0,1\}^*$ *and range* $\mathbb{G}_1$. *The signature scheme* HKSS *is defined as follows:*

$\mathsf{SSetup}(1^\kappa)$: *Generate a key for* PHF *as* $k \leftarrow \mathsf{Gen}(1^\kappa)$ *and output* $\mathsf{pp} = k$.

$\mathsf{SGen}(1^\kappa)$: *Pick* $\mathsf{sk} \leftarrow \mathbb{Z}_q$ *at random, compute* $\mathsf{pk} := g_2^{\mathsf{sk}}$, *and output* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{SSign}(\mathsf{sk}, m)$: *Parse* k *from* pp. *Pick* $s \leftarrow \mathbb{Z}_q$ *uniformly at random and compute* $y := \mathsf{Eval}(k, m)^{\frac{1}{\mathsf{sk}+s}}$. *Output* $\sigma := (s, y)$.

$\mathsf{SVerify}(\mathsf{pk}, m, \sigma)$: *Parse* σ *as* (s, y). *If* $e(y, \mathsf{pk} \cdot g_2^s) = e(\mathsf{Eval}(k, m), g_2)$ *then output* 1, *otherwise output* 0.

$\mathsf{RandSK}(\mathsf{sk}, \rho)$: *Compute* $\mathsf{sk}' := \mathsf{sk} + \rho \mod q$ *and output* sk'.

$\mathsf{RandPK}(\mathsf{pk}, \rho)$: *Compute* $\mathsf{pk}' := \mathsf{pk} \cdot g_2^\rho$ *and output* pk'.

Obviously all three correctness conditions hold. It remains to show that HKSS is unforgeable under re-randomized keys.

Theorem 2 (Unforgeability of HKSS Under Re-randomized Keys). *The signature scheme* HKSS *as defined in Definition 13 is unforgeable under re-randomized keys (Definition 8) in the standard model, if* HKSS *is unforgeable under chosen message attacks (Definition 9).*

Proof. Assume towards contradiction that there exists an efficient adversary $\mathcal{A}$ against the unforgeability under re-randomizable keys. Then, we construct an adversary $\mathcal{B}$ against the existential unforgeability of the underlying signature scheme, which runs $\mathcal{A}$ as a black-box. The algorithm $\mathcal{B}$ simulates the oracle $\mathcal{O}_1$ by simply forwarding the query to its own signing oracle and it uses the randomness switching technique for the simulation of $\mathcal{O}_2$. That is, whenever $\mathcal{A}$ sends a message-randomness pair (m, ρ) to $\mathcal{O}_2$, then $\mathcal{A}$ queries its signing oracle on m and adjusts the key by subtracting ρ from s. The formal description of $\mathcal{B}$ and the oracle $\mathcal{O}_2$ is given in the following:

$\underline{\mathcal{B}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pk}):}$
$(\sigma^*, m^*, \rho^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$
Parse σ^* as (s, y)
$s' := s + \rho^*$
output $(s', y), m^*$

$\underline{\mathcal{O}_2(\mathsf{sk}, \rho, m):}$
$(s, y) \leftarrow \mathcal{O}(m)$
$s' := s - \rho$
output (s', y)

For the analysis, let us assume that $\mathcal{A}$'s success probability in the experiment $\mathsf{UFRK}_{\mathcal{A}}^{\mathsf{HKSS}}(\kappa)$ is bigger than $1/\mathsf{poly}(\kappa)$. It is easy to see that $\mathcal{B}$ is efficient and that the simulation of $\mathcal{A}$'s signing oracle $\mathcal{O}_1$ is perfect. Now, we show that $\mathcal{B}$ also provides a perfect simulation of the oracle $\mathcal{O}_2$. Whenever $\mathcal{A}$ sends (ρ, m) to $\mathcal{O}_2$, then $\mathcal{B}$ returns a signature (s', y) for which it holds that $e(y, \mathsf{pk} \cdot g_2^{\rho} \cdot g_2^{s'}) = e(\mathsf{Eval}(k, m)^{\frac{1}{\mathsf{sk}+s}}, g_2^{\mathsf{sk}+\rho+(s-\rho)}) = e(\mathsf{Eval}(k, m), g_2)$, which has obviously the correct distribution.

Finally, we argue that $\mathcal{B}$ outputs a valid signature whenever $\mathcal{A}$ outputs a valid forgery. To see this, note that $(s' = s + \rho^*, y)$ for m^* under pk, whenever $\mathcal{A}$ returns a valid signature (s, y) for m^* under the re-randomized key $\mathsf{pk} \cdot g_2^{\rho}$, since $e(y, (\mathsf{pk} \cdot g_2^{\rho}) \cdot g_2^{s}) = e(y, \mathsf{pk} \cdot g_2^{\rho+s}) = e(y, \mathsf{pk} \cdot g_2^{s'})$. Combining this with the proof of existential unforgeability of the modified version of the Hofheinz Kiltz signature schemes from [31,32] rules out the existence of $\mathcal{A}$.

4 Efficient Sanitizable Signatures

In this section we show how to build efficient unlinkable sanitizable signatures from signatures with perfectly re-randomizable keys.

4.1 Preliminaries

We recall the definitions and security notions of the other building blocks required for our construction of sanitizable signatures. Namely we recall the definitions of CCA secure public key-encryption and non-interactive zero-knowledge proof systems.

CCA Secure Public-Key Encryption. A public key encryption scheme $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of a key generation algorithm $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa)$, an encryption algorithm $c \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$, and a decryption algorithm $m \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$. We omit the standard correctness condition and recall the standard notion of CCA security.

Definition 14 (Indistinguishability under Chosen Ciphertext Attacks). *A public key encryption scheme* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under chosen ciphertext attacks *(*$\mathsf{IND\text{-}CCA}$*) if for all (possibly stateful) PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *the probability that the experiment* $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\kappa)$ *evaluates to 1 is negligibly bigger than* $1/2$ *(in* κ*), where*

Experiment $\mathsf{IND\text{-}CCA}^{\mathcal{E}}_{\mathcal{A}}(\kappa)$:
$(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa)$
$b \leftarrow \{0,1\}$
$m_0, m_1 \leftarrow \mathcal{A}_0^{\mathsf{Dec}(\mathsf{dk},\cdot)}(\mathsf{ek})$
$c_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$
$a \leftarrow \mathcal{A}_1^{\mathsf{Dec}'(\mathsf{dk},c_b,\cdot)}(c_b)$
if $a = b$*, then output* 1
else output 0

$\mathsf{Dec}'(\mathsf{dk}, c_b, c)$:
if $c \neq c_b$
then output $\mathsf{Dec}(\mathsf{dk}, c)$
else output $\perp$

Non-interactive Zero-Knowledge Proof System. We recall the definitions of non-interactive zero-knowledge proof systems. A non-interactive zero-knowledge proof system $(\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ for a language $\mathcal{L}$ with the corresponding relation $\mathcal{R}$ consists of a setup algorithm $\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$ that generates a common reference string, a prover algorithm $\pi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}, x, w)$ that takes as input the common reference string crs, a statement x, and a witness w and outputs a zero-knowledge proof π; and a verification algorithm $b \leftarrow \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi)$ that outputs 1 iff $x \in \mathcal{L}$ and 0 otherwise. We omit the standard definition of correctness and recall the definitions of (perfect) soundness, zero-knowledge, and proof of knowledge.

Definition 15 (Perfect Soundness). *A NIZK scheme has* perfect soundness *if and only if for all* $\kappa \in \mathbb{N}$ *and all adversaries* $\mathcal{A}$ *it holds that*

$$\Pr[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi) = 0 \mid x \notin \mathcal{L}\,] = 1$$

Definition 16 (Zero-knowledge). *A NIZK scheme has* computational zero-knowledge *if for all* $\kappa \in \mathbb{N}$ *there exists an efficient simulator* $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1)$ *such that for all adversaries* $\mathcal{A}$ *it holds that*

$$\left| \begin{array}{l} \Pr\left[\,\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa) : \mathcal{A}^{\mathsf{P}_{\mathsf{ZK}}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \\ -\Pr\left[\,(\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{S}_0(1^\kappa) : \mathcal{A}^{\mathsf{S}'(\mathsf{crs},\mathrm{T},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \end{array} \right| \leq \mathsf{negl}(\kappa),$$

where $\mathsf{S}'(\mathsf{crs}, \mathrm{T}, x, w) = \mathsf{S}_1(\mathsf{crs}, \mathrm{T}, x)$ *if* $(x, w) \in \mathcal{R}$ *and outputs failure otherwise.*

Definition 17 (Proof of Knowledge). *A NIZK scheme is a* proof of knowledge *if there exists an efficient extractor* $\mathsf{Ext} = (\mathsf{Ext}_0, \mathsf{Ext}_1)$ *such that the following conditions hold:*

For all polynomial time adversaries $\mathcal{A}$ *it holds that*

$$\left| \begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1] \\ -\Pr[(\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{Ext}_0(1^\kappa) : \mathcal{A}(\mathsf{crs}) = 1] \end{array} \right| \leq \mathsf{negl}(\kappa).$$

For all polynomial time adversaries $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{c} (\mathsf{crs}, \mathrm{T}) \leftarrow \mathsf{Ext}_0(1^\kappa); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}); \\ w \leftarrow \mathsf{Ext}_1(\mathsf{crs}, \mathrm{T}, x, \pi) : (x, w) \in \mathcal{R} \end{array} \middle| \, \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \pi) = 1 \right] \geq \frac{1}{\mathsf{poly}(\kappa)}.$$

4.2 Our Construction

In the following, we describe our construction of a sanitizable signature scheme based on signatures with re-randomizable keys. Similar to previous constructions [11,12], we sign the parts of the message that cannot be changed by the sanitizer and a description of valid modifications ADM with a separate signature scheme. The main part of our construction, and which is very different from all previous schemes, is the computation of the signature on the parts that can be modified by the sanitizer. The basic idea here is that we compute this signature using a signature scheme with re-randomizable keys. That is, we compute this signature using a re-randomized private and public key-pair $(\mathsf{sk}', \mathsf{pk}')$, which was either re-randomized by the signer or the sanitizer. To allow for an easy Proof and Judge algorithm and avoid rewinding in the proof, we have to provide a way to check that pk' is in fact the re-randomization of the signer's or the sanitizer's public key. Therefore, we also include an encryption of the actual public key. In the Proof algorithm the signer can then decrypt and return this public key along with a proof of correct decryption.

In the following, for the sake of brevity all algorithms are assumed to implicitly take the public parameters as input.

Construction 1. Let $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ be a signature scheme with perfectly re-randomizable keys, $\Sigma_{\text{FIX}} = (\mathsf{SSetup}_{\text{FIX}}, \mathsf{SGen}_{\text{FIX}}, \mathsf{SSign}_{\text{FIX}}, \mathsf{SVerify}_{\text{FIX}})$ be a deterministic signature scheme, $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme, and $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ as well as $\Pi_{ZK} = (\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ be two non-interactive zero-knowledge proof systems for the languages $\mathcal{L}_1$ and $\mathcal{L}_2$, where the language $\mathcal{L}_1$, used in Sign, Sanit, and Verify, contains tuples $(\mathsf{ek}, c, \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$ for which there exists witness $w = (\omega, \rho)$ such that

$$c = \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega) \quad \wedge \quad \mathsf{pk}' = \mathsf{RandPK}(\mathsf{pk}, \rho)$$

or

$$c = \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega) \quad \wedge \quad \mathsf{pk}' = \mathsf{RandPK}(\mathsf{pk}_{san}, \rho).$$

The second language $\mathcal{L}_2$, used in Proof and Judge, contains tuples $(\mathsf{ek}, c, \widehat{\mathsf{pk}})$ for which there exists witness $w = (\psi, \mathsf{dk})$ such that

$$(\mathsf{ek}, \mathsf{dk}) = \mathsf{EGen}(1^\kappa; \psi) \quad \wedge \quad \widehat{\mathsf{pk}} = \mathsf{Dec}(\mathsf{dk}, c).$$

Define our sanitizable signature scheme $\mathrm{SanS} = (\mathsf{KGen}_{sig}, \mathsf{KGen}_{san}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ as follows:

Setup and Key Generation. The setup algorithm generates two common reference strings for the two different zero-knowledge proofs (of knowledge) and the key generation algorithm the required keys. They are formally defined as follows:

$\underline{\mathsf{Setup}(1^\kappa)}$:
- $\mathsf{crs}_{\mathsf{PoK}} \leftarrow \mathsf{Setup}_{\mathsf{PoK}}(1^\kappa)$
- $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{Setup}_{\mathsf{ZK}}(1^\kappa)$
- $\mathsf{pp}_s \leftarrow \mathsf{SSetup}(1^\kappa)$
- $\mathsf{pp} = (\mathsf{crs}_{\mathsf{PoK}}, \mathsf{crs}_{\mathsf{ZK}}, \mathsf{pp}_s)$
- output pp

$\underline{\mathsf{KGen}_{san}(1^\kappa)}$:
- $(\mathsf{sk}_{san}, \mathsf{pk}_{san}) \leftarrow \mathsf{SGen}(1^\kappa)$
- output $(\mathsf{sk}_{san}, \mathsf{pk}_{san})$

$\underline{\mathsf{KGen}_{sig}(1^\kappa)}$:
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SGen}(1^\kappa)$
- $(\mathsf{sk}_{\mathrm{FIX}}, \mathsf{pk}_{\mathrm{FIX}}) \leftarrow \mathsf{SGen}_{\mathrm{FIX}}(1^\kappa)$
- $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{EGen}(1^\kappa; \psi)$
- $\mathsf{sk}_{sig} := \begin{pmatrix} \mathsf{sk}_{\mathrm{FIX}}, \mathsf{sk}, \mathsf{dk}, \\ \mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi \end{pmatrix}$
- $\mathsf{pk}_{sig} := (\mathsf{pk}_{\mathrm{FIX}}, \mathsf{pk}, \mathsf{ek})$
- output $(\mathsf{sk}_{sig}, \mathsf{pk}_{sig})$

Signing and Sanitizing. The signing and sanitizing algorithms first parse their inputs and Sanit further checks that Mod is actually an admissible modification and modifies the message accordingly. The Sign algorithm now signs the fixed part with $\mathsf{sk}_{\mathrm{FIX}}$, while Sanit can simply reuse the σ_{FIX} of the input signature. The remainder of the two algorithms proceeds identically, by re-randomizing the respective key, encrypting the original key, proving that sk' is indeed a re-randomization and signing the full message together with signer's and sanitizer's public keys as seen in the following:

$\underline{\mathsf{Sign}(m, \mathsf{sk}_{sig}, \mathsf{pk}_{san}, \textsc{Adm})}$:

Parse sk_{sig} as
$(\mathsf{sk}_{\textsc{Fix}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\textsc{Fix}}, \mathsf{pk}, \mathsf{ek}, \psi)$
$\mathsf{pk}_{sig} := (\mathsf{pk}_{\textsc{Fix}}, \mathsf{pk}, \mathsf{ek})$
$m_{\textsc{Fix}} := (\textsc{Fix}_{\textsc{Adm}}(m), \textsc{Adm}, \mathsf{pk}_{san})$
$\sigma_{\textsc{Fix}} := \mathsf{SSign}_{\textsc{Fix}}(\mathsf{sk}_{\textsc{Fix}}, m_{\textsc{Fix}})$
$\rho \leftarrow \chi$
$\mathsf{sk}' \leftarrow \mathsf{RandSK}(\mathsf{sk}, \rho)$
$\mathsf{pk}' \leftarrow \mathsf{RandPK}(\mathsf{pk}, \rho)$
$c \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}; \omega)$
$x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
$\tau \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}, x, (\rho, \omega))$
$\sigma' := \mathsf{SSign}(\mathsf{sk}', (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}))$
output $\sigma = (\sigma_{\textsc{Fix}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$

$\underline{\mathsf{Sanit}(m, \textsc{Mod}, \sigma, \mathsf{pk}_{sig}, \mathsf{sk}_{san})}$:

Parse pk_{sig} as $(\mathsf{pk}_{\textsc{Fix}}, \mathsf{pk}, \mathsf{ek})$
Parse σ as $(\sigma_{\textsc{Fix}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$
If $\textsc{Adm}(\textsc{Mod}) = 0$
output $\perp$
$\widehat{m} := \textsc{Mod}(m)$
$\rho \leftarrow \chi$
$\widehat{\mathsf{sk}'} \leftarrow \mathsf{RandSK}(\mathsf{sk}_{san}, \rho)$
$\widehat{\mathsf{pk}'} \leftarrow \mathsf{RandPK}(\mathsf{pk}_{san}, \rho)$
$\widehat{c} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathsf{pk}_{san}; \omega)$
$x := (\widehat{c}, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \widehat{\mathsf{pk}'})$
$\widehat{\tau} \leftarrow \mathsf{P}_{\mathsf{PoK}}(\mathsf{crs}, x, (\rho, \omega))$
$\widehat{\sigma}' := \mathsf{SSign}(\widehat{\mathsf{sk}}', (\widehat{m}, \mathsf{pk}_{sig}, \mathsf{pk}_{san}))$
output $(\widehat{m}, \widehat{\sigma} = (\sigma_{\textsc{Fix}}, \widehat{\sigma}', \textsc{Adm}, \widehat{\mathsf{pk}'}, \widehat{c}, \widehat{\tau}))$

Verification. The verification algorithm checks that both signatures and the proof of knowledge verify:

$\underline{\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san})}$:

Parse pk_{sig} as $(\mathsf{pk}_{\textsc{Fix}}, \mathsf{pk}, \mathsf{ek})$.
Parse σ as $(\sigma_{\textsc{Fix}}, \sigma', \textsc{Adm}, \mathsf{pk}', c, \tau)$.
$m_{\textsc{Fix}} := (\textsc{Fix}_{\textsc{Adm}}(m), \textsc{Adm}, \mathsf{pk}_{san})$
$x := (c, \mathsf{ek}, \mathsf{pk}, \mathsf{pk}_{san}, \mathsf{pk}')$
$$\text{if} \begin{pmatrix} \mathsf{SVerify}_{\textsc{Fix}}(\mathsf{pk}_{\textsc{Fix}}, m_{\textsc{Fix}}, \sigma_{\textsc{Fix}}) = 1 \\ \text{and } \mathsf{SVerify}(\mathsf{pk}', (m, \mathsf{pk}_{sig}, \mathsf{pk}_{san}), \sigma') = 1 \\ \text{and } \mathsf{V}_{\mathsf{PoK}}(\mathsf{crs}, x, \tau) = 1 \end{pmatrix}$$
then output 1
else output 0

Proving and Judging. The algorithm Proof first verifies that the given signature is indeed valid. It then parses its inputs and decrypts the ciphertext c, thus revealing who computed the signature. Moreover, it computes a zero-knowledge proof asserting that the decryption was performed correctly. The Judge checks whether the proof of decryption is correct. If the proof π contains pk_{san}, then the Judge algorithm outputs `San`. In all other cases, Judge returns `Sign`.

$\underline{\mathsf{Proof}(\mathsf{sk}_{sig}, m, \sigma, \mathsf{pk}_{san})}$:

If $\mathsf{Verify}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}) = 0$
  output $\perp$
Parse sk_{sig} as
  $(\mathsf{sk}_{\text{FIX}}, \mathsf{sk}, \mathsf{dk}, \mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek}, \psi)$
Parse σ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$
$\widehat{\mathsf{pk}} \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$
$x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
$\phi \leftarrow \mathsf{P}_{\mathsf{ZK}}(\mathsf{crs}, x, (\psi, \mathsf{dk}))$
output $(\widehat{\mathsf{pk}}, \phi)$

$\underline{\mathsf{Judge}(m, \sigma, \mathsf{pk}_{sig}, \mathsf{pk}_{san}, \pi)}$:

Parse pk_{sig} as $(\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$
Parse σ as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$
Parse π as $(\widehat{\mathsf{pk}}, \phi)$
$x := (\mathsf{ek}, c, \widehat{\mathsf{pk}})$
if $\begin{pmatrix} \mathsf{pk}_{san} = \widehat{\mathsf{pk}} \\ \text{and } \mathsf{V}_{\mathsf{ZK}}(\mathsf{crs}, x, \phi) = 1 \end{pmatrix}$
then output `San`
else output `Sign`

4.3 Security Proof

We are now ready to state the main theorem about the security of the construction described above.

Theorem 3. *If* $\Sigma = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SVerify}, \mathsf{RandSK}, \mathsf{RandPK})$ *is a signature scheme that is unforgeable under re-randomized keys,* $\Sigma_{\text{FIX}} = (\mathsf{SSetup}_{\text{FIX}}, \mathsf{SGen}_{\text{FIX}}, \mathsf{SSign}_{\text{FIX}}, \mathsf{SVerify}_{\text{FIX}})$ *is a signature scheme that is strongly existentially unforgeable,* $\Pi_{PoK} = (\mathsf{Setup}_{\mathsf{PoK}}, \mathsf{P}_{\mathsf{PoK}}, \mathsf{V}_{\mathsf{PoK}})$ *is a computationally zero-knowledge perfectly sound proof of knowledge system,* $\Pi_{ZK} = (\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{P}_{\mathsf{ZK}}, \mathsf{V}_{\mathsf{ZK}})$ *is a computationally zero-knowledge perfectly sound proof system,* $\mathcal{E} = (\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a CCA-secure public key encryption scheme, then Construction 1 is sanitizer-accountable, signer-accountable, immutable, (proof-restrictedly) transparent, and unlinkable.*

We sketch the basic ideas of the proofs here. The full proofs for each security property are deferred to the [26].

Sanitizer Accountability. Consider an efficient adversary $\mathcal{A}$ against the sanitizer accountability of SanS, whose final output is a tuple $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$. The signature σ^* can be parsed as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$ and the public-key as $\mathsf{pk}_{sig} = (\mathsf{pk}_{\text{FIX}}, \mathsf{pk}, \mathsf{ek})$. Whenever $\mathcal{A}$ wins, then $\mathcal{A}$ never queried $(\mathsf{pk}^*_{san}, m^*)$ to its sign oracle Sign, the signature verifies, and Judge outputs `Sign`. This implies that

$$\mathsf{SVerify}(\mathsf{pk}', (m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}), \sigma') = 1$$

and

$$\widehat{\mathsf{pk}} = \mathsf{pk}.$$

Since $(\mathsf{pk}^*_{san}, m^*)$ is fresh and the re-randomization factor ρ^* can be extracted from the proof τ, it follows that $(m^*, \mathsf{pk}_{sig}, \mathsf{pk}^*_{san}), \sigma', \rho^*$ is a valid forgery under a re-randomization of pk, which contradicts the unforgeability under re-randomized keys of Σ.

Signer Accountability. Let $\mathcal{A}$ be an efficient adversary against the signer accountability of SanS, whose final output is a tuple $(\mathsf{pk}^*_{sig}, m^*, \sigma^*, \pi^*)$, where pk^*_{sig} can be parsed as $(\mathsf{pk}_{\text{FIX}}{}^*, \mathsf{pk}^*, \mathsf{ek}^*)$, the signature σ^* as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$, and π^* as $(\widehat{\mathsf{pk}}, \phi)$. Whenever $\mathcal{A}$ wins, then $\mathcal{A}$ never queried $(\mathsf{pk}^*_{sig}, m^*)$ to its sanitizer oracle Sanit, the signature is valid, and Judge outputs $\mathtt{San}$. This implies that

$$\mathsf{SVerify}(\mathsf{pk}', (m^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}), \sigma') = 1$$

and

$$\widehat{\mathsf{pk}} = \mathsf{pk}_{san}.$$

Since $(\mathsf{pk}^*_{sig}, m^*)$ is fresh and the re-randomization factor ρ^* can be extracted from the proof τ, it follows that $(m^*, \mathsf{pk}^*_{sig}, \mathsf{pk}_{san}), \sigma', \rho^*$ is a valid forgery under a re-randomization of pk_{san}, which contradicts the unforgeability under re-randomized keys of Σ.

Immutability. Let $\mathcal{A}$ be an efficient adversary against the immutability of SanS, whose final output is tuple $(\mathsf{pk}^*_{san}, m^*, \sigma^*)$. The signature σ^* can be parsed as $(\sigma_{\text{FIX}}, \sigma', \text{ADM}, \mathsf{pk}', c, \tau)$. From the winning conditions of $\mathcal{A}$ we can conclude, that the tuple $(\text{FIX}_{\text{ADM}}(m^*), \text{ADM}, \mathsf{pk}^*_{san})$ is different from any such tuple corresponding to one of the Sanit queries and that

$$\mathsf{SVerify}_{\text{FIX}}(\mathsf{pk}_{\text{FIX}}, (\text{FIX}_{\text{ADM}}(m^*), \text{ADM}, \mathsf{pk}^*_{san}), \sigma_{\text{FIX}}).$$

However, then it follows that $(\text{FIX}_{\text{ADM}}(m^*), \text{ADM}, \mathsf{pk}^*_{san}), \sigma_{\text{FIX}}$ is a valid forgery under pk_{FIX}, which contradicts the strong existential unforgeability of Σ_{FIX}.

(Proof Restricted) Transparency. The proof of transparency is the most involved one and proceeds in several game-hops. We start with the transparency game with the bit $b = 0$. Then, first, we use the simulatability of the zero knowledge proofs, to switch to a game, where all proofs are simulated. We can then change the $\mathsf{Sanit}/\mathsf{Sign}$ oracle to no longer encrypt the re-randomized public key, but an independently chosen public key instead. The answers of Proof queries can be changed accordingly. The difference between the two games can be bounded by reducing it to the CCA security of the encryption scheme. Next, the bit b is flipped to 1. Due to the simulated proofs, the outputs of $\mathsf{Sanit}/\mathsf{Sign}$ are distributed identically before and after the switch. The outputs of the Proof oracle, however, may differ, if the attacker manages to ask a valid query, such that the signature reuses one of the ciphertexts computed by $\mathsf{Sanit}/\mathsf{Sign}$. This leads to two different cases. If the attacker uses a new pk'_{san} in its query, then it must also compute a new proof of knowledge, which means that it has to know the content of the ciphertext, leading to a trivial reduction to CCA security (or even one-wayness) of the encryption scheme. In the other case, the fact that the signature must verify, leads to a forgery under a re-randomized key, which would contradict the unforgeability under re-randomized keys of Σ. Finally, we can switch back to real ciphertexts instead of random ones and undo the simulation

of the zero knowledge proofs, thus arriving at the transparency game with the bit $b = 1$.

Since the distances between all hops can be bounded by negligible functions, the difference between the two cases of the game is also negligible.

Unlinkability. Let $\mathcal{A}$ be an efficient adversary against the signer unlinkability of SanS and consider a query

$$((m_i^0, \text{MOD}_i^0, \sigma_i^0), (m_i^1, \text{MOD}_i^1, \sigma_i^1))$$

by $\mathcal{A}$ to the LoRSanit oracle. We parse the signature σ_i^b as $(\sigma_{\text{FIX},i}^b, \sigma_i'^b, \text{ADM}_i^b, \mathsf{pk}'^b_i, c_i^b, \tau_i^b)$ and denote by (m_b^*, σ_b^*) the answer to this query depending on the choice of b in the experiment. The signature σ_b^* can be parsed as $(\sigma_{\text{FIX},b}, \sigma_b', \text{ADM}_b, \mathsf{pk}_b', c_b, \tau_b)$. The conditions required for the LoRSanit oracle to provide such an answer implies that the distribution of $(\sigma_0', \text{ADM}_0, \mathsf{pk}_0', c_0, \tau_0)$ and $(\sigma_1', \text{ADM}_1, \mathsf{pk}_1', c_1, \tau_1)$ are identical. Therefore, the only way to distinguish between the two cases is if it holds that $\sigma_{\text{FIX},i}^0 \neq \sigma_{\text{FIX},i}^1$. However, since Σ_{FIX} is deterministic, such a query would imply that one of $(m_{\text{FIX}}^0, \sigma_{\text{FIX}}^0)$ and $(m_{\text{FIX}}^1, \sigma_{\text{FIX}}^1)$ must necessarily be a valid forgery under pk_{FIX}, which contradicts the strong existential unforgeability of Σ_{FIX}.

5 Instantiating the Construction

We instantiate our generic construction with compatible and efficient instantiations in the random oracle model. For the two signature schemes, we choose standard Schnorr signatures as defined in Definition 11 for Σ, as well as a derandomized[2] version of Schnorr signatures for Σ_{FIX}[3]. The encryption scheme and proof systems are instantiated with the Cramer Shoup encryption scheme [22], and Σ-protocols that we convert into a non-interactive zero-knowledge proof via the Fiat-Shamir transform [24]. The Cramer Shoup encryption scheme is defined as follows:

Definition 18 (Cramer Shoup Encryption Scheme). *Let $\mathbb{G}$ be a cyclic group of prime order q with two random generators g_1, g_2 and let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function. The Cramer Shoup encryption scheme, working over $\mathbb{G}$, is defined as follows:*

$\mathsf{EGen}(1^\kappa)$: *The key generation algorithm proceeds as follows: Pick $x, y, a, b, a', b' \leftarrow \mathbb{Z}_q$ uniformly at random, compute $h := g_1^x g_2^y$, $h := g_1^a g_2^b$, $h := g_1^{a'} g_2^{b'}$, set $\mathsf{dk} := (x, y, a, b, a', b')$ and $\mathsf{ek} := (h, c, d)$ and output $(\mathsf{dk}, \mathsf{ek})$.*

[2] The randomness is generated by a PRF.

[3] Note, that while the original security proof [38,39] for Schnorr signatures only proves standard existential unforgeability, it can be easily adapted to prove strong existential unforgeability.

$\mathsf{Enc}(\mathsf{ek}, m)$: *The encryption algorithm proceeds as follows: Parse* ek *as* (h, c, d) *and choose* $r \leftarrow \mathbb{Z}_q$ *uniformly at random. Compute* $\alpha := \mathcal{H}(g_1^r, g_2^r, h^r \cdot m)$ *and* $C := (g_1^r, g_2^r, h^r \cdot m, (cd^\alpha)^r)$*. Output* C*.*

$\mathsf{Dec}(\mathsf{dk}, C)$: *The decryption algorithm proceeds as follows: Parse* dk *as* (x, y, a, b, a', b') *and* C *as* (u, v, w, e)*. Compute* $\alpha := \mathcal{H}(u, v, w)$ *and check if* $u^{a+\alpha a'} \cdot v^{b+ab'} = e$ *holds. If it holds output* $w/(u^x \cdot v^y)$*. Otherwise output* $\perp$*.*

The remaining building blocks for our construction are two non-interactive zero-knowledge proof systems that we instantiate with specific Fiat-Shamir transformed [24] Σ-protocols. The first proof system is for the language $\mathcal{L}_1$ and the statement that we want to prove in our concrete instantiation looks as follows:

$$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$$

$$PoK\left\{(\omega, \rho): \begin{array}{c} g_1^\omega = c_1 \ \wedge \ g_2^\omega = c_2 \ \wedge \ (cd^\alpha)^\omega = c_4 \\ \wedge \ \frac{h^\omega}{g^\rho} = \frac{c_3}{\mathsf{pk}'} \ \wedge \ \left(g_1^\rho = \frac{\mathsf{pk}'}{\mathsf{pk}} \ \vee \ g_1^\rho = \frac{\mathsf{pk}'}{\mathsf{pk}_{san}}\right) \end{array}\right\}.$$

$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \mathsf{pk}', \mathsf{pk}_{san}, \mathsf{pk})$

Prover (x, i, ω, ρ)		Verifier (x)
$(T_i, z_i, s_i) \leftarrow \mathsf{S}_\Sigma(1^\kappa)$		
$u_{1-i}, u_2 \leftarrow \mathbb{Z}_q$		
$T_{1-i} := g_1^{u_{1-i}}$		
$T_2 := g_1^{u_2} \quad T_3 := g_2^{u_2}$		
$T_4 := (cd^\alpha)^{u_2} \quad T_5 := h^{u_2}$	$\xrightarrow{T_0, T_1, T_2, T_3, T_4, T_5}$	
		$z \leftarrow \mathbb{Z}_q$
	$\xleftarrow{z}$	
$z_{1-i} := z - z_i$		
$s_{1-i} := \rho \cdot z_{1-i} + u_{1-i}$		
$s_2 := \omega \cdot z + u_2$	$\xrightarrow{s_0, s_1, s_2, z_0}$	
		$z_1 := z - z_0$
		$g_1^{s_0} \stackrel{?}{=} T_0 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}})^{z_0}$
		$g_1^{s_1} \stackrel{?}{=} T_1 \cdot (\frac{\mathsf{pk}'}{\mathsf{pk}_{san}})^{z_1}$
		$g_1^{s_2} \stackrel{?}{=} T_2 \cdot c_1^z$
		$g_2^{s_2} \stackrel{?}{=} T_3 \cdot c_2^z$
		$(cd^\alpha)^{s_2} \stackrel{?}{=} T_4 \cdot c_4^z$
		$\frac{h^{s_2}}{g_1^{s_0+s_1}} \stackrel{?}{=} \frac{T_5}{T_0 T_1} \cdot (\frac{c_3}{\mathsf{pk}'})^z$

Fig. 1. Σ-Protocol for Encryption of Public Key

Note that the statement that we are proving can be expressed as a logical combination of discrete logarithm proofs of knowledge. For the design of each single discrete logarithm proofs we deploy Schnorr's Σ-protocols from [40]. We then formulate the complete proof using standard parallel composition techniques, first introduced in [20,21]. The complete protocol is depicted in Fig. 1. It is worth mentioning that, in order to express the logical disjunction of our statement, the prover must run the simulator S provided by the zero-knowledge property (Definition 16). For the specific case of Σ-protocols S_Σ works by randomly sampling z_i, s_i from $\mathbb{Z}_q$ and computing T_i as $g_1^{s_i}/(\frac{\mathsf{pk}'}{\mathsf{pk}})^{z_i}$ (or $g_1^{s_i}/(\frac{\mathsf{pk}'}{\mathsf{pk}_{san}})^{z_i}$, respectively). Finally, as mentioned above, the protocol can be made non-interactive by using the Fiat-Shamir transformation. Note that this allow us to drop the first tuple of elements $(T_0, \ldots, T_5)$ since they can be simply recomputed from the public parameters and the further messages of the protocol and their integrity can be checked by recomputing the hash function.

In the following, we show how to instantiate the proof of knowledge for the language $\mathcal{L}_2$. We prove the following statement:

$$x := (\mathsf{ek} := (g_1, g_2, h, c, d), C := (c_1, c_2, c_3, c_4), \hat{\mathsf{pk}})$$

$$ZK\left\{(\chi, \psi) : g_1^\chi g_2^\psi = h \ \wedge \ c_1^\chi c_2^\psi = \frac{c_3}{\hat{\mathsf{pk}}}\right\}.$$

Again, for the concrete instantiation in Figure 2 we deploy parallel composition of Σ-protocols made non-interactive via the Fiat-Shamir transformation. Combining these building blocks yields a highly efficient sanitizable signature scheme.

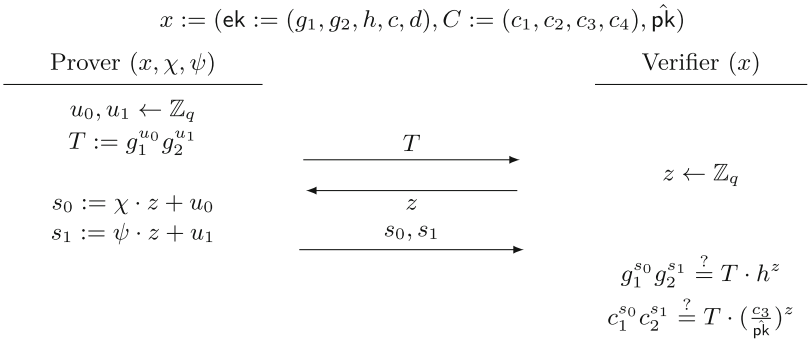


Fig. 2. Σ-Protocol for Proof of Decryption

6 Conclusion

In this paper, we formalized the novel notion of signature schemes that are unforgeable under re-randomized keys. Furthermore, we showed that Schnorr's

signature scheme [40,41] is unforgeable under re-randomized keys in the random oracle model and that Hofheinz' and Kiltz' signature scheme [31,32] is unforgeable under re-randomized keys in the standard model.

Based on signature schemes with re-randomizable keys we then gave a construction of unlinkable sanitizable signatures and an instantiation, which is at least one order of magnitude faster than all previously known schemes.

Acknowledgments. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – www.cispa-security.org) and the project PROMISE. Moreover, it was supported by the Initiative for Excellence of the German federal and state governments through funding for the Saarbrücken Graduate School of Computer Science and the DFG MMCI Cluster of Excellence. Part of this work was also supported by the German research foundation (DFG) through funding for the collaborative research center 1223. Dominique Schröder was also supported by an Intel Early Career Faculty Honor Program Award.

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
3. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
5. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. Cryptology ePrint Archive, Report 2003/096 (2003). http://eprint.iacr.org/2003/096
6. 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)
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Crypt. **21**(2), 149–177 (2008)
8. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
9. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
10. Brzuska, C., Busch, H., Dagdelen, O., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable signatures for tree-structured data: definitions and constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)

11. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
12. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
13. Brzuska, C., Pöhls, H.C., Samelin, K.: Non-interactive public accountability for sanitizable signatures. In: De Capitani di Vimercati, S., Mitchell, C. (eds.) EuroPKI 2012. LNCS, vol. 7868, pp. 178–193. Springer, Heidelberg (2013)
14. Brzuska, C., Pöhls, H.C., Samelin, K.: Efficient and perfectly unlinkable sanitizable signatures without group signatures. In: Katsikas, S., Agudo, I. (eds.) EuroMPI 2013. LNCS, vol. 8341, pp. 12–30. Springer, Heidelberg (2014)
15. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
16. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
17. Canard, S., Jambert, A., Lescuyer, R.: Sanitizable signatures with several signers and sanitizers. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 35–52. Springer, Heidelberg (2012)
18. Catalano, D.: Homomorphic signatures and message authentication codes. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 514–519. Springer, Heidelberg (2014)
19. Chang, E.-C., Lim, C.L., Xu, J.: Short redactable signatures using random trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
20. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
21. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof 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)
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)
23. Derler, D., Slamanig, D.: Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 455–474. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26059-4_25
24. 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)
25. Fischlin, M., Fleischhacker, N.: Limitations of the meta-reduction technique: the case of Schnorr signatures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 444–460. Springer, Heidelberg (2013)
26. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatureswith rerandomizable keys. Cryptology ePrint Archive, Report 2015/395 (2015). http://eprint.iacr.org/2015/395
27. Franco, P.: Understanding Bitcoin: Cryptography: Engineering and Economics. Wiley, Chichester (2015)

28. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
29. Furukawa, J., Yonezawa, S.: Group signatures with separate and distributed authorities. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 77–90. Springer, Heidelberg (2005)
30. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
31. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
32. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. J. Crypt. **25**(3), 484–527 (2012)
33. Johnson, R., Walsh, L., Lamb, M.: Homomorphic signatures for digital photographs. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 141–157. Springer, Heidelberg (2012)
34. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
35. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
36. Mitsunari, S., Saka, R., Kasahara, M.: A new traitor tracing. IEICE Trans. **E85–A**(2), 481–484 (2002)
37. Pöhls, H.C., Samelin, K.: On updatable redactable signatures. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 457–475. Springer, Heidelberg (2014)
38. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
39. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Crypt. **13**(3), 361–396 (2000)
40. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
41. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Crypt. **4**(3), 161–174 (1991)
42. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)

Fault-Tolerant Aggregate Signatures

Gunnar Hartung(✉), Björn Kaidel, Alexander Koch,
Jessica Koch, and Andy Rupp

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{gunnar.hartung,bjoern.kaidel,alexander.koch,
jessica.koch,andy.rupp}@kit.edu

Abstract. Aggregate signature schemes allow for the creation of a short aggregate of multiple signatures. This feature leads to significant reductions of bandwidth and storage space in sensor networks, secure routing protocols, certificate chains, software authentication, and secure logging mechanisms. Unfortunately, in all prior schemes, adding a single *invalid* signature to a valid aggregate renders the whole aggregate invalid. Verifying such an invalid aggregate provides no information on the validity of any individual signature. Hence, adding a single faulty signature destroys the proof of integrity and authenticity for a possibly large amount of data. This is largely impractical in a range of scenarios, e.g. secure logging, where a single tampered log entry would render the aggregate signature of all log entries invalid.

In this paper, we introduce the notion of fault-tolerant aggregate signature schemes. In such a scheme, the verification algorithm is able to determine the subset of all messages belonging to an aggregate that were signed correctly, provided that the number of aggregated faulty signatures does not exceed a certain bound.

We give a generic construction of fault-tolerant aggregate signatures from ordinary aggregate signatures based on cover-free families. A signature in our scheme is a small vector of aggregated signatures of the underlying scheme. Our scheme is bounded, i.e. the number of signatures that can be aggregated into one signature must be fixed in advance. However the length of an aggregate signature is logarithmic in this number. We also present an unbounded construction, where the size of the aggregate signature grows linearly in the number of aggregated messages, but the factor in this linear function can be made arbitrarily small.

The additional information encoded in our signatures can also be used to speed up verification (compared to ordinary aggregate signatures) in cases where one is only interested in verifying the validity of a single message in an aggregate, a feature beyond fault-tolerance that might be of independent interest. For concreteness, we give an instantiation using a suitable cover-free family.

Keywords: Aggregate signatures · Fault-tolerance · Cover-free family

This work was supported by the German Federal Ministry of Education and Research within the framework of the project KASTEL_IoE in the Competence Center for Applied Security Technology (KASTEL).

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 331–356, 2016.
DOI: 10.1007/978-3-662-49384-7_13

1 Introduction

Aggregate signature schemes allow anyone to aggregate multiple signatures by different signers into a single combined signature, which is considerably smaller than the size of the individual signatures. This type of digital signature schemes was first proposed and instantiated by Boneh, Gentry, Lynn and Shacham [Bon+03], and has since evolved into a diverse and active research area.

Applications of Aggregate Signatures. The main motivation for aggregate signature schemes is to save bandwidth and storage space. Therefore, their applications are manifold [AGH10].

A well-known field of application are *sensor networks*, which consist of several small sensors that measure an aspect of their physical environment and send their findings to a central base station. Digital signatures ensure the integrity and authenticity of the measurements during transfer from the sensors to the base station. Using a conventional digital signature scheme, the verifying base station would need to receive each signature separately, which is bandwidth-intensive. However, if the signatures were aggregated beforehand using an aggregate signature scheme, the bandwidth consumption on the side of the base station is reduced drastically. Also, verifying an aggregate signature is typically considerably faster than verifying all individual signatures.

Another application is *secure logging*. Log files are used to record events like user actions, system errors, failed log-in attempts as well as general information, and play an important role in computer security by providing, for example, accountability and a basis for intrusion detection. Log files are usually kept for very long periods of time, which means that thousands or even millions of log entries need to be stored. Digital signatures are used to ensure the integrity of the log data. For aggregate signature schemes, it is sufficient to store one single aggregate signature over all log entries, instead of an individual signature per log entry as with a normal digital signature scheme. Whenever a new log entry is added to the log file, one simply calculates a signature for the new entry and aggregates it into the already existing aggregate signature.

Aggregate signatures can also be useful for *authenticating software*. To ensure the validity of software libraries and programs it has become common to sign their code and/or compiled binaries. Mobile operating systems often only allow signed programs to be executed. Again, it is advantageous to use an aggregate signature to save download bandwidth and verification overhead upon execution, e.g. if all programs are verified at boot time. Like in the logging scenario, aggregate signatures allow for installation of new applications without having to store the individual signatures of all installed programs.

Problem Statement. In all known aggregate signature schemes an aggregate signature is invalid (i.e., verification fails) if just one invalid message–signature pair is contained in the aggregate. Note that either this pair was already invalid (i.e., the individual signature was not valid for this particular message) when the

aggregate was created or a "wrong" message is included for verification. In any case, the verification algorithm can give no information about which message–signature pair is the reason for the failure or if other message–signature pairs were valid. This essentially renders the aggregate useless after an invalid signature is added, even though the majority of the messages might have been correctly signed.

For sensor networks, this means that the measurements of all sensors are lost even if only a single sensor sends an invalid signature, for example because of calculation glitches or transmission errors. Usually it is not feasible for computationally weak sensors to ensure the validity of their signature before sending it, since many signature schemes use expensive operations like pairings for verification. An aggregator could ensure the validity of the individual signatures before aggregation, but this would undo one of the advantages of the aggregate signature scheme altogether.

The same problem occurs in the logging scenario: If one log entry is not correctly signed, is tampered with, or is lost (for example through hard disk errors or crashes), the signature for the whole log file becomes invalid. In [MT09] Ma and Tsudik state that this is one of the reasons why they still need to store individual signatures for every log entry, although they use an aggregated signature for the complete log. This is undesirable, since one of the motivations for using the aggregate signature schemes for logging is to save storage space.

This problem also affects the software authentication scenario. If a new program is installed and its signature is invalid or the code of an already installed program gets changed (through hard disk problems etc.), the whole aggregated signature used for authenticating the software becomes invalid. In the worst case this would mean that no program can be executed anymore, because the operating system might block every unauthenticated program.

Contribution. To solve the above mentioned problems, we introduce the concept of fault-tolerant aggregate signature schemes, which are able to tolerate a specific number of invalid (or faulty) signatures while aggregating. In such a scheme, the verification algorithm does not output boolean values like "valid" and "invalid" but instead outputs a list of validly signed messages and will leave out all messages that are invalid.

Note that in contrast to ordinary aggregate signatures, fault-tolerant aggregate signatures cannot offer an aggregate signature size which is independent of the number of individual signatures to be aggregated. In other words, we cannot hope to aggregate an unlimited number of individual signatures using a constant-size aggregate. This easily follows from an information-theoretic argument: Let us assume we fix the size of an aggregate signature to l bits. This l-bit string then needs to be used by the verification algorithm (as the only "source of information") to determine which of its input messages are valid. Hence, based on the l-bit string, the algorithm can distinguish at most 2^l different outputs. However, considering n messages and corresponding individual signatures, d of which are invalid, there are $\binom{n}{n-d}$ possible different subsets (and thus outputs)

which should be distinguishable by the verification algorithm by considering this string. So n is upper bounded by $\binom{n}{n-d} \leq 2^l$. (For a more formal argument using the notation of fault-tolerant aggregate signature introduced later, refer to Appendix A.)

Besides a formal framework for fault-tolerant aggregate signatures, we also present a generic construction which can be used to turn any aggregate signature scheme into a fault-tolerant scheme. This construction makes use of cover-free families [KS64] to provide fault-tolerance and comes with a tight security reduction to the underlying signature scheme. For concreteness, we explicitly describe how to instantiate our scheme with a cover-free family based on polynomials over a finite field [KRS99], which has a compact representation. (We generalize the known family to multivariate polynomials in Appendix B.) This leads to an instantiation featuring short aggregate signatures relative to the number n of individual signatures that are aggregated (provided that the maximal number of faults the scheme should tolerate is relatively small compared to n).

As an additional feature, our construction allows the verification of an individual signature in a fashion that is more efficient (e.g., saving a number of costly pairing operations in the case of pairing-based aggregate signatures) than verifying the complete aggregate. This provides a level of flexibility to the signature scheme as demanded by certain applications such as secure logging [MT09].

As a shortcoming of our scheme, we need to *assume* that aggregates may only contain a previously fixed upper bound d of invalid individual signatures. If for some reason this bound is exceeded, the faulty signatures may affect the verifiability of other messages, as is the case for common aggregate signatures. This is also analogous to error-correction codes (which are related to cover-free families), where only a specific number of errors can be located.

Basic Idea of Our Construction. To get a glimpse of our generic construction of a fault-tolerant aggregate signature scheme, we now informally illustrate the basic idea. Let an ordinary aggregate signature scheme (e.g., BGLS) and n individual signatures $\sigma_1, \ldots, \sigma_n$ generated using this scheme be given. Our goal is to detect $d = 1$ faulty individual signatures. To achieve this, our approach is to choose m subsets $T_1, \ldots, T_m \subset \{\sigma_1, \ldots, \sigma_n\}$ of individual signatures and aggregate the signatures of each subset, thereby yielding aggregate signatures $\tau_1, \ldots, \tau_m$, such that

1. m is (significantly) smaller than n and
2. even if one of the individual signatures is faulty and the corresponding aggregate signatures τ_i will be invalid, all other individual signatures σ_j are aggregated into at least one different, valid signature τ_k.

For example, consider the following binary 4×6 matrix for which $n = 6$ and $m = 4$.

$$A := (a_{i,j}) := \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

A describes a solution to the above mentioned problem as follows: The 1 entries in column j indicate in which T_i the individual signature σ_j is contained. Consequently, the 1-entries in row i indicate the σ_j contained in T_i. More precisely, $T_i := \{\sigma_j \colon a_{i,j} = 1\}$ and τ_i is the aggregate of all $\sigma_j \in T_i$. Observe that while it is usually unnecessary for the verification to know the order of the claims from the aggregation process, in our fault-tolerant scheme, specifying the correct order is inevitable.

For the matrix above, an aggregate signature $\tau = (\tau_1, \tau_2, \tau_3, \tau_4)$ for individual signatures $\sigma_1, \ldots, \sigma_6$ would be formed in the following manner:

$$\tau = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{pmatrix} = \begin{pmatrix} \mathsf{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathsf{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathsf{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathsf{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix} \hat{=} \begin{pmatrix} \sigma_1 & & & \sigma_4 & & \sigma_6 \\ \sigma_1 & \sigma_2 & & & \sigma_5 & \\ & \sigma_2 & \sigma_3 & \sigma_4 & & \\ & & \sigma_3 & & \sigma_5 & \sigma_6 \end{pmatrix},$$

where Agg informally denotes the aggregation function of the underlying aggregate signature scheme.

Let us assume that only one signature σ_j is faulty. Then all τ_i are faulty where $a_{i,j} = 1$. However, because all other σ_k were also aggregated into at least one different τ_i, we can still derive the validity of σ_k.

For a concrete example, suppose σ_1 is faulty. Then τ_1 and τ_2 will be faulty, whereas τ_3 and τ_4 are valid. We see that σ_2, σ_3 and σ_4 occur in τ_3, and σ_5, σ_6 occur in τ_4, and so we may be sure that the corresponding messages were signed.

The matrix A defined above has the property that it can tolerate one faulty signature, i.e., if just one signature is faulty, then all other messages can still be verified. Unfortunately, this is not possible if two or more faulty signatures are aggregated. Lets assume that σ_1 and σ_2 are faulty. In this case, τ_1, τ_2 and τ_3 become invalid and τ_4 is the only valid signature. We could still derive the validity of $\sigma_3, \sigma_5, \sigma_6$, because τ_4 is valid. However, the validity of σ_1, σ_2 and σ_4 can no longer be verified, since they were never aggregated to τ_4.

Note that our scheme does not support fully flexible aggregation: Each column of A can only be used to hold one individual signature, as can be seen in the example above. Two aggregate signatures where the same column is used can not be aggregated further without losing the guarantee of fault-tolerance. However, our scheme still supports a notion of aggregation which is only slightly restricted: Individual signatures can always be aggregated, while aggregate signatures can only be aggregated if no column is used in both. As long as this requirement is met, signatures can be aggregated in any order. This notion is sufficient for many use cases, we discuss this further in Sect. 3.

The construction of matrices that can tolerate $d > 1$ faulty signatures is more intricate, but incidence matrices belonging to d-cover-free families turned out to imply the desired property. Informally speaking, in such a matrix the "superposition" $\boldsymbol{s}$ of up to d arbitrary column vectors $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_d}}$, i.e., the vector $\boldsymbol{s}$ which has a 1 at position ℓ if at least one of the vectors $\boldsymbol{a_{i_1}}, \ldots, \boldsymbol{a_{i_d}}$ has a 1 at this position, does not "cover" any other distinct column vector $\boldsymbol{a_j}$ ($j \notin \{i_1, \ldots, i_d\}$). In other words, there is at least one position ℓ such that $\boldsymbol{a_j}$ has a 1 at this position but $\boldsymbol{s}$ shows a 0. This implies that if at most d individual signatures (each

belonging to one column) are invalid, then each distinct individual signature is contained in at least one valid aggregate signature, and the corresponding message can therefore be trusted. Hence applying such a matrix, as sketched above, implies that any subset of faulty individual signatures of size up to d will not compromise the trustworthiness of any other message. There are different constructions of d-cover-free families for d and n of unlimited size (where these parameters need to satisfy certain conditions depending on the family) featuring $m \ll n$ for choices of the parameters n, d with $d \ll n$.

Related Work. The first full aggregate scheme was constructed by Boneh et al. [Bon+03] in the random oracle model. Full aggregate schemes allow any user to aggregate signatures of different signers, i.e., aggregation is a public operation. Furthermore it is possible to aggregate individual signatures as well as already aggregated signatures in any order. In [HSW13] Hohenberger, Sahai and Waters give the first construction of such a scheme in the standard model using multilinear maps. Recently, Hohenberger, Koppula, and Waters [HKW15] have constructed a "universal signature aggregator" based on indistinguishability obfuscation. A universal signature aggregator can aggregate signatures from any set of signing algorithms, even if they use different algebraic settings. In [ZS11] Zaverucha and Stinson construct an aggregate one-time-signature.

Since it has proven difficult to construct full aggregate schemes in the standard model, a lot of research was focused on signature schemes with some form of *restricted* aggregation. One major type of restricted aggregation is *sequential* aggregation, as proposed by Lysyanskaya et al. [Lys+04]. In these schemes, the aggregate is sequentially sent from signer to signer and each signer can add new information to the aggregate. Multiple constructions are known, both in the random oracle [Lys+04, Nev08, Bol+07, Ger+12] and the standard model [Lu+06, Sch11, LLY15]. Another type of aggregation is *synchronized* aggregation, as proposed by Gentry and Ramzan [GR06]. Here, a special synchronization information, like the current time period, is used while signing. All signatures sharing the same synchronizing information behave like signatures of a full aggregate scheme, i.e., both individual and aggregated signatures can be aggregated in any order. Again, schemes in the random oracle [AGH10, GR06] and standard model [AGH10] are known. Other authors considered aggregate signature schemes that need interaction between the signers [BN07, BJ10] or can only partially aggregate the signatures [Her06, BGR14].

Our construction is based on cover-free families, which are a combinatorial structure that was first introduced by Kautz and Singleton [KS64] in the language of coding theory. They have several applications in cryptography, for example group testing [STW97], multireceiver authentication codes [SW99], encryption [Cra+07, Dod+02, HK04] and traitor-tracing [TS06]. There are multiple constructions of signature schemes using cover-free families. Hofheinz, Jager, and Kiltz [HJK11] use cover-free families to construct a $(m, 1)$-programmable hash function. They then use this hash function to construct conventional digital signature schemes from weak assump-

tions. Zaverucha and Stinson [ZS11] construct an aggregate one-time-signature using cover-free families.

Outline. Section 2 introduces some notations, conventions, and preliminary definitions. Section 3 presents a general definition of fault-tolerant aggregate signature schemes and some properties of such schemes, such as the security definition. Our construction is presented and analyzed in Sect. 4. Afterwards, we discuss an instantiation of our scheme with a specific class of cover-free families in Sect. 5.

2 Preliminaries

Let $[n] := \{1, \ldots, n\}$. The *multiplicity* of an element m in a multiset M is the number of occurrences of m in M. For two multisets M_1, M_2, the union $M_1 \cup M_2$ is defined as the multiset where the multiplicity of each element is the sum of the multiplicities in M_1, M_2.

If v is a vector or a tuple, $v[i]$ refers to the i-th entry of v. If M is a matrix, $\text{rows}(M)$ and $\text{cols}(M)$ denote the number of rows and columns of M, respectively. For $i \in [\text{rows}(M)], j \in [\text{cols}(M)]$, $M[i, j]$ is the entry in the i-th row and j-th column of M.

Throughout the paper, $\kappa \in \mathbb{N}$ is the security parameter. We say an algorithm A is probabilistic polynomial time (PPT) if the running time of A is polynomial in κ and A is a probabilistic algorithm. All algorithms are implicitly given 1^κ as input, even when not noted explicitly.

In this work, σ usually refers to signatures of standard aggregate signature schemes, whereas τ mostly refers to signatures of a fault-tolerant aggregate signature scheme.

2.1 Aggregate Signatures

Let us quickly review the definition of aggregate signature schemes and the associated security notion, as defined in [Bon+03]. An *aggregate signature scheme* is a tuple of four PPT algorithms:

- $\mathsf{KeyGen}(1^\kappa)$ creates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ creates a signature for message m under secret key sk.
- $\mathsf{Agg}(C_1, C_2, \sigma_1, \sigma_2)$ takes as input two multisets of public-key and message pairs C_1 and C_2 and corresponding signatures σ_1 and σ_2 and creates an aggregate signature σ, certifying the validity of the messages in $C_1 \cup C_2$ under the corresponding public keys.
- $\mathsf{Verify}(C, \sigma)$ takes as input a multiset of public-key and message pairs C and an aggregate signature σ for C and outputs 1, if the signature is valid, and 0 otherwise.

For *correctness* we require that any signature that is generated by the signature scheme by applications of Sign and Agg using key pairs of the scheme, is valid, i.e. Verify outputs 1.

Security Notion for Aggregate Signatures. The security experiment for aggregate signatures consists of three phases [Bon+03]:

- *Setup Phase.* The challenger generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KeyGen}(1^\kappa)$ and gives the public key pk to the adversary.
- *Query Phase.* The adversary $\mathcal{A}$ may (adaptively) issue signature queries m_i to the challenger, who responds with $\sigma_i := \mathsf{Sign}(\mathsf{sk}, m_i)$.
- *Forgery Phase.* Finally, $\mathcal{A}$ outputs a multiset of public-key and message pairs C^* and a signature σ^*.

The adversary *wins* the experiment iff there is a message m^* such that $c^* = (\mathsf{pk}, m^*)$ is in C^*, $\mathsf{Verify}(C^*, \sigma^*) = 1$, and m^* has never been submitted to the signature oracle.

An aggregate signature scheme Σ is (t, q, ε)-*secure* if there is no adversary $\mathcal{A}$ running in time at most t, making at most q queries to the signature oracle and winning in the above experiment with probability at least ε.

2.2 Cover-Free Families

For our construction of a fault-tolerant aggregate signature scheme in Sect. 3 we need a d-cover-free family, which allows us to detect up to d invalid individual signatures in our aggregate signature.

Definition 1. *A* d-cover-free family $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ *(denoted by* d*-CFF) consists of a set* $\mathcal{S}$ *of* m *elements and a set* $\mathcal{B}$ *of* n *subsets of* $\mathcal{S}$*, where* $d < m < n$*, such that: For any* d *subsets* $B_{i_1}, \ldots, B_{i_d} \in \mathcal{B}$ *and all distinct* $B \in \mathcal{B} \setminus \{B_{i_1}, \ldots, B_{i_d}\}$*, it holds that*

$$|B \setminus \bigcup_{k=1}^{d} B_{i_k}| \geq 1.$$

So, it is not possible to cover a single subset with at most d different subsets. To get a better representation of a d-CFF and to simplify the handling of it, we will use a matrix in the following way:

Definition 2. *For a* d*-CFF* $\mathcal{F} = (\mathcal{S}, \mathcal{B})$*, where the elements of* $\mathcal{S}$ *and* $\mathcal{B}$ *have a well-defined order, such that we can write* $\mathcal{S} = \{s_1, \ldots, s_m\}$*,* $\mathcal{B} = \{B_1, \ldots, B_n\}$*, we define its* incidence matrix $\mathcal{M}$ *as follows:*

$$\mathcal{M}[i, j] = \begin{cases} 1, & \text{if } s_i \in B_j, \\ 0, & \text{otherwise.} \end{cases}$$

The i-th row *of* $\mathcal{M}$ *is denoted by* $\mathcal{M}_i \in \{0,1\}^n$*, for* $i \in [m]$.

So, $s_i \in \mathcal{S}$ corresponds to row i and $B_j \in \mathcal{B}$ corresponds to column j, i.e. $\mathcal{M}$ has m rows and n columns.

3 Fault-Tolerant Aggregate Signatures

Claims and Claim Sequences. As a notational convenience, we introduce the concept of claims. A *claim* c is simply a pair (pk, m) of a public key and a message, conveying the meaning that the owner of pk has authenticated the message m. In this sense, a signature σ for m that is valid under pk is a proof for the claim c. This definition allows for a more compact representation of our algorithms.

The signature scheme we introduce in Sect. 4 critically requires an order among the claims. While the actual order is arbitrary, it must be maintained by the aggregation and verification algorithms. We therefore define the fault-tolerant signature schemes based on sequences of claims, instead of multisets.

More precisely, when an individual signature τ' for a claim c is first aggregated into an aggregate signature τ, one must assign a unique "position" j to c. If one wishes to verify τ, one must call Verify with a sequence of claims C that has c at its j-th position, i.e. $C[j] = c$. Therefore, two aggregate signatures τ_1, τ_2 for two sequences of claims C_1, C_2 can not be aggregated if $C_1[j] \neq C_2[j]$ for some j.

Thus, our scheme does not support *fully* flexible, arbitrary aggregation. However, if the signers agree in advance on the positions j of their claims, they can aggregate all their signatures into a single combined signature τ. This prerequisite can easily be fulfilled in many applications. In wireless sensor networks for example, one only has to configure each sensor to use a different position j. Moreover, it is always possible to use our scheme as a sequential aggregate signature scheme, since the position j of a claim needs only be determined when it is first aggregated. Our scheme is therefore suitable for all applications where sequential aggregate signatures are sufficient, too, such as secure logging [MT09].

For the general aggregation setting, we will have to deal with "incomplete" claim sequences, i.e. if a claim sequence does not yet contain a claim at position j. We therefore assume the existence of a *claim placeholder* $\perp$ that may be contained in claim sequences. When aggregating the signatures of two such incomplete claim sequences C_1, C_2, the claim sequences will be *merged*, meaning that claim placeholders in C_1 are replaced by actual claims from C_2, for each position j where $C_1[j] = \perp$ and $C_2[j] \neq \perp$, and vice versa. (This merging operation replaces the multiset union used by common aggregate signature schemes.)

For technical reasons, we also require that there is no position where C_1 and C_2 both contain a claim, even if the claims are identical. As a consequence, if a signature τ is aggregated into two different aggregate signatures τ_1, τ_2 using the same position j, τ_1 and τ_2 can not be aggregated later. Note, however, that this does not preclude the possibility to aggregate τ into τ_1 and τ_2 at different positions.

We now move to the formal definition. A *claim sequence* is a tuple of claims and claim placeholders $\perp$. The multiset of elements of a claim sequence C excluding $\perp$ is denoted by $\mathsf{elem}(C)$. Two claim sequences C_1, C_2 are *mergeable* if for all $i \in [\min(|C_1|, |C_2|)]$ it holds that $C_1[i] = \perp$ or $C_2[i] = \perp$ or $C_1[i] = C_2[i]$.

C_1, C_2 are called *exclusively mergeable*, if for all such i it holds that $C_1[i] = \perp$ or $C_2[i] = \perp$. (In particular, two exclusively mergeable sequences are mergeable.) For example, for distinct claims c_1, c_2, c_3, define $C_1 = (\perp, c_2, c_3)$, $C_2 = (c_1, \perp, \perp)$, $C_2' = (c_1, c_2, \perp)$, $C_2'' = (c_1, c_3, c_2)$. Then, C_1, C_2 are exclusively mergeable, C_1, C_2' are mergeable, but not exclusively mergeable, and C_1, C_2'' are not mergeable.

Let C_1 and C_2 be two mergeable claim sequences of length k and l, respectively. Without loss of generality, assume $k \geq l$. Then the *merged claim sequence* $C_1 \sqcup C_2$ is $(c_1, \ldots, c_k)$, where

$$c_i := \begin{cases} C_1[i], & \text{if } C_2[i] = \perp,\ C_2[i] = C_1[i] \text{ or } i > l, \\ C_2[i], & \text{otherwise.} \end{cases}$$

The *empty signature* λ is a signature valid for exactly the claim sequences containing only $\perp$ and the empty claim sequence.

Subsequences. Let $C = (c_1, \ldots, c_n)$ be a tuple and $b \in \{0,1\}^n$ be a bit sequence specifying a selection of indices. Then $C[b]$ is the subsequence of C containing exactly the elements c_j where $b[j] = 1$, replacing all other claims by $\perp$. In particular, if $\mathcal{M}$ is an incidence matrix of a cover-free family, then $C[\mathcal{M}_i]$ is the subsequence containing all c_j where $\mathcal{M}[i,j] = 1$ and $\perp$ at all other positions.

Syntax of Fault-Tolerant Signature Schemes. We are now ready to define fault-tolerant aggregate signature schemes. The intuitive difference of such a scheme to an ordinary aggregate signature scheme is that its verification algorithm does not only output a boolean value 1 or 0 that determines if either all claims are valid or at least one claim is invalid, but it gives (some) information on which claims in C are valid. In particular, it outputs the set of valid claims. If the signature contains more errors than the scheme can cope with, Verify may output just a subset of the valid claims. Other claims may be clearly false or just not certainly true. (The verification algorithm ought to be conservative and reject a claim in case of uncertainty.)

The aggregation algorithm is called with two claim *sequences*, hence, before aggregating, a single claim c must be converted to a claim sequence $C = (\perp, \ldots, \perp, c)$ by assigning a position to c.

Definition 3. *An* aggregate signature scheme with list verification[1] *is a tuple of four PPT algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Agg}, \mathsf{Verify})$*, where*

- $\mathsf{KeyGen}(1^\kappa)$ *creates a key pair* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ *creates a signature for message* m *under secret key* sk.
- $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$ *takes as input two exclusively mergeable claim sequences* C_1 *and* C_2 *and corresponding signatures* τ_1 *and* τ_2 *and creates an aggregate signature* τ*, certifying the validity of the claim sequence* $C_1 \sqcup C_2$.

[1] The name "list verification" is chosen to indicate the changes in syntax, in particular that the verification algorithm outputs a multiset (list) instead of just 1 or 0.

– $\mathsf{Verify}(C, \tau)$ *takes as input a claim sequence* C *and an aggregate signature* τ *for* C *and outputs a multiset of claims* $C_{\text{valid}} \subseteq \mathsf{elem}(C)$ *specifying the valid claims in* τ*. Note that this may be a proper subset of* $\mathsf{elem}(C)$*, or even empty, if none of the claims can be derived from* τ *(for certain). Again, here,* C *may contain* $\perp$ *as a claim placeholder.*

Σ *is required to be* correct *as defined in the following paragraphs.*

Regular Signatures. Informally, a signature is regular if it is created by running the algorithms of Σ. More formally, let C be a claim sequence and τ be a signature. We recursively define what it means for τ to be *regular for* C:

– If $(\mathsf{pk}, \mathsf{sk})$ is in the image of $\mathsf{KeyGen}(1^\kappa)$ and $C = ((\mathsf{pk}, m))$ for a message m, and if τ is in the image of $\mathsf{Sign}(\mathsf{sk}, m)$, then τ is said to be regular for C and for any claim sequence obtained by prepending any number of $\perp$ symbols to C.
– If τ_1 is regular for a claim sequence C_1, τ_2 is regular for another claim sequence C_2, and C_1, C_2 are exclusively mergeable, then τ is regular for $C_1 \sqcup C_2$ if τ is in the image of $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$.
– The empty signature λ is regular for the claim sequences containing only $\perp$ and the empty claim sequence $()$.

If a signature τ is not regular for a claim sequence C, it is called *irregular* for C.

Fault Tolerance. Let $M = \{(c_1, \tau_1), \ldots, (c_n, \tau_n)\}$ be a multiset of claim and signature pairs, which is partitioned into two multisets M_{irreg} and M_{reg}, containing the pairs for which τ_i is irregular for $C = (c_i)$ and regular for C, respectively.[2] Then the multiset M *contains* d *errors*, if $|M_{\text{irreg}}|$ is d. An aggregate signature scheme Σ with list verification is *tolerant against* d *errors*, if for any such multiset M containing at most d errors, for any signature τ that was aggregated from the signatures in M (in arbitrary order) and the corresponding claim sequence C, which may additionally contain any number of claim placeholders $\perp$, we have

$$R \subseteq \Sigma.\mathsf{Verify}(C, \tau),$$

where R is the multiset of all the claims (i.e. the first component of the pairs) in M_{reg}. In other words, Verify outputs at least all claims of regular signatures.[3]

A *d-fault-tolerant aggregate signature scheme* is an aggregate signature scheme with list verification that is tolerant against d errors. A *fault-tolerant aggregate signature scheme* is a scheme that is d-fault-tolerant for some $d > 0$.

[2] While there may be schemes with valid signatures which are not regularly generated, like in the usual correctness properties, our guarantees do only concern regular signatures.

[3] Intuitively, one would expect $R = \Sigma.\mathsf{Verify}(C, \tau)$. However, this is not achievable in general, as the aggregation of multiple irregular signatures may contain a new valid claim c_i corresponding to an irregular signature σ_i. This does not contradict security, as crafting such irregular signatures may be hard if one does not know σ_i.

Correctness. Observe that 0-fault-tolerance means that if M contains only regularly created signatures, then Verify must output all claims in M (or C, respectively). This is analogous to the common definition of correctness for aggregate signature schemes. We therefore call an aggregate signature scheme with list verification *correct*, if it is tolerant against 0 errors.

Errors During Aggregation. Our definitions above assume that aggregation is always done correctly. This is a necessary assumption, since it is impossible to give guarantees for arbitrary errors that happen during aggregation. Consider for example a faulty aggregation algorithm that ignores its input and just outputs a random string. It is an interesting open question to find a fault-tolerant signature scheme that can tolerate certain types of aggregation errors, too.

Compression Ratio. Denote by $\mathsf{size}(\sigma)$ the size of a signature σ. Let C be a claim sequence of length n, and σ^* an aggregate signature of maximum size[4] which is regular for C. We say that an aggregate signature scheme has *compression ratio* $\rho(n)$ iff

$$\frac{n}{\mathsf{size}(\sigma^*)} \in \Theta(\rho(n)).$$

Note that if $\mathsf{size}(\sigma^*)$ is upper bounded by a constant, then the compression ratio is $\rho(n) = n$, which is optimal for common aggregate signature schemes. As argued in the introduction this is not possible for fault-tolerant aggregate signatures, cf. Appendix A.

Security Experiment. The security experiment for aggregate signatures with list verification, which is a direct adaption of the standard security experiment of [Bon+03], consists of three phases:

- *Setup Phase.* The challenger generates a pair of keys $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KeyGen}(1^\kappa)$ and gives the public key pk to the adversary.
- *Query Phase.* The adversary $\mathcal{A}$ may (adaptively) issue signature queries m_i to the challenger, who responds with $\tau_i := \mathsf{Sign}(\mathsf{sk}, m_i)$.
- *Forgery Phase.* Finally, $\mathcal{A}$ outputs a claim sequence C^* and a signature τ^*.

The adversary *wins* the experiment iff there is a message m^* such that $c^* = (\mathsf{pk}, m^*) \in \mathsf{Verify}(C^*, \tau^*)$, and m^* has never been submitted to the signature oracle.

Definition 4. *An aggregate signature scheme with list verification is* (t, q, ε)-secure *if there is no adversary* $\mathcal{A}$ *running in time at most* t*, making at most* q *queries to the signature oracle and winning in the above experiment with probability at least* ε*.*

[4] The size of an aggregated signature might depend on the aggregation order.

4 Generic Construction of Fault-Tolerant Aggregate Signatures

In this section, we present our generic construction of fault-tolerant aggregate signature schemes. It is based on an arbitrary aggregate signature scheme Σ, which is used as a black box, and a cover-free family. Our scheme inherits its security from Σ, and can tolerate d faults if it uses a d-cover-free family.

Our Construction. In the following we describe a generic construction of our fault-tolerant aggregate signature scheme. For this let Σ be an ordinary aggregate signature scheme. Moreover, let $\mathcal{M}$ be the incidence matrix of a d-cover-free family $\mathcal{F} = (\mathcal{S}, \mathcal{B})$, as defined in Sect. 2.2. For the sake of presentation, we first show our bounded construction. In this version of our construction, the maximum number of signatures that can be aggregated is $\mathrm{cols}(\mathcal{M})$. We discuss in Sect. 4.1 how to remove this restriction.

In our scheme, signatures for just one claim are simply signatures of the underlying scheme Σ, whereas aggregate signatures are short vectors of signatures of Σ. We identify each element of the universe $\mathcal{S}$ with a position in this vector, and each subset $B \in \mathcal{B}$ with an individual signature of the underlying scheme Σ.

Here, we require also that the underlying scheme Σ supports claim sequences and claim placeholders as an input to Agg and Verify, contrary to just multisets, as in the definition of Sect. 2.1. Moreover, we assume that Σ supports the empty signature λ as an input to Agg and Verify. However, these are not essential restrictions, as for instance any normal aggregate scheme may be easily adapted to a scheme of the modified syntax, by ignoring any order and claim placeholders, i.e. applying $\mathsf{elem}(\cdot)$ on the claim sequences before they are passed to the Agg and Verify algorithm.

- $\mathsf{KeyGen}(1^\kappa)$ creates a key pair $(\mathsf{pk}, \mathsf{sk})$ by using the KeyGen algorithm of Σ.
- $\mathsf{Sign}(\mathsf{sk}, m)$ takes as input a secret key sk and a message m and outputs the signature as given by $\Sigma.\mathsf{Sign}(\mathsf{sk}, m)$.
- $\mathsf{Agg}(C_1, C_2, \tau_1, \tau_2)$ takes as input two exclusively mergeable claim sequences C_1 and C_2 and corresponding signatures τ_1 and τ_2. It proceeds as follows:
  1. If one or both of the claim sequences C_k ($k \in \{1, 2\}$) contains only one (proper) claim c, i.e. τ_k is an individual signature, then σ_k is initialized as τ_k, the corresponding signature given to Agg. Then τ_k is expanded to a vector, by setting

$$\tau_k[i] := \begin{cases} \sigma_k, & \text{if } \mathcal{M}[i, j] = 1, \\ \lambda, & \text{otherwise,} \end{cases} \qquad \text{for } i = 1, \ldots, m,$$

     where j is the index of c in the claim sequence.
  2. Then the signatures τ_1, τ_2, which are both vectors now, are aggregated component-wise, i.e.

$$\tau[i] = \Sigma.\mathsf{Agg}(C_1[\mathcal{M}_i], C_2[\mathcal{M}_i], \tau_1[i], \tau_2[i]).$$

Finally, Agg outputs τ.

– $\mathsf{Verify}(C, \tau)$ takes as input a claim sequence C and an aggregate signature τ for C. For each component $\tau[i]$ of τ it computes $b_i :=\Sigma.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i])$ and outputs the multiset of valid claims

$$C_{\text{valid}} := \mathsf{elem}\left(\bigsqcup_{i\in[k], b_i=1} C[\mathcal{M}_i]\right). \tag{1}$$

We now prove the security of our scheme.

Theorem 1. *If Σ is a (t, q, ε)-secure aggregate signature scheme, then the scheme defined above is a (t', q, ε)-secure aggregate signature scheme with list verification, where t' is approximately the same as t.*

Proof. Let Σ' be the scheme described above. The following argument is rather direct. Assume that $\mathcal{A}$ is an adversary breaking the (t', q, ε)-security of Σ'. We construct an attacker $\mathcal{B}$ breaking the (t, q, ε)-security of Σ.

$\mathcal{B}$ simulates $\mathcal{A}$ as follows. In the setup phase $\mathcal{B}$ starts executing $\mathcal{A}$ and passes its own input pk on to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a signature query for a message m, $\mathcal{B}$ obtains the signature σ by forwarding m to the challenger. $\mathcal{B}$ then passes σ to $\mathcal{A}$ and continues the simulation. When $\mathcal{A}$ outputs a claim sequence C^* and a signature τ^*, $\mathcal{B}$ checks if there is a claim $c^* = (\mathsf{pk}, m^*)$ in C^*, such that m^* was never queried by $\mathcal{A}$.

If this is not the case, then $\mathcal{B}$ outputs $\perp$ and terminates. Otherwise, by definition of $\Sigma'.\mathsf{Verify}$, there must be an index i such that

$$\Sigma.\mathsf{Verify}(C^*[\mathcal{M}_i], \tau^*[i]) = 1 \tag{2}$$

and $m^* \in \mathsf{elem}(C^*[\mathcal{M}_i])$. $\mathcal{B}$ outputs $C^*[\mathcal{M}_i]$ and $\tau^*[i]$. This is a valid signature, because of (2).

Note that $\mathcal{B}$'s queries are exactly the same as $\mathcal{A}$'s. Therefore, if $\mathcal{A}$ did not query m^*, then neither did $\mathcal{B}$. Thus, $\mathcal{B}$ wins exactly iff $\mathcal{A}$ wins, and therefore $\mathcal{B}$ also has success probability ε. We also see that $\mathcal{B}$ makes at most q queries. Finally, it is easy to verify that the running time of $\mathcal{B}$ is approximately the same as the running time of $\mathcal{A}$. □

We now turn to proving the fault-tolerance of our scheme.

Theorem 2. *Let Σ be the aggregate signature scheme with list verification defined above. If Σ is based on a d-CFF, then it is tolerant against d errors, and in particular, it is correct.*

Proof. Let $M = \{(c_1, \tau_1), \ldots, (c_m, \tau_m)\}$ be a multiset of claim and signature pairs, which is partitioned into two multisets M_{irreg} and M_{reg}, containing the pairs for which τ_i is irregular for $C_i = (c_i)$ or regular for C_i, respectively. Let M contain at most d errors, i.e., $|M_{\text{irreg}}| \leq d$. Moreover, let τ be a signature that was aggregated from the signatures in M (in arbitrary order) and C the

corresponding claim sequence. To simplify the proof, we assume without loss of generality that $C = (c_1, \ldots, c_n)$, i.e., the order in the claim sequence is the same as in the indexing of the signatures in M and it does not include any claim placeholders $\perp$. Finally, let $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ be a d-cover-free family used by the scheme above, where $\mathcal{S} = \{s_1, \ldots, s_m\}$ and $\mathcal{B} = \{B_1, \ldots, B_n\}$.

We need to show that $R \subseteq \Sigma.\mathsf{Verify}(C, \tau) =: V$, where R is the multiset of all the claims in M_{reg}. Recall that $\text{rows}(\mathcal{M})$ and $\text{cols}(\mathcal{M})$ denote the number of rows and columns of $\mathcal{M}$, respectively. Let $b_i := \Sigma'.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i])$ for all $i \in [\text{rows}(\mathcal{M})]$.

Assume for a contradiction that there is a claim c^* that is contained strictly more often in R than in V. Then there exists an index j^* such that $C[j^*] = c^*$ and $b_i = 0$ for all $i \in [\text{rows}(\mathcal{M})]$ with $\mathcal{M}[i, j^*] = 1$.

In the following, let $I := \{i \in [\text{rows}(\mathcal{M})] \colon \mathcal{M}[i, j^*] = 1\}$ be the set of these indices I, and observe that these are the indices of all rows where the signature for c^* is aggregated into $\tau[i]$.

We now try to obtain a contradiction by showing that the set B_{j^*}, which corresponds to the column j^* of $\mathcal{M}$, is covered by the sets B_k, corresponding to the columns of the claims with irregular signatures.

For each $i \in I$, since $b_i = 0$ and using the correctness of Σ', there must be some $k \in [n]$ such that $(c_k, \sigma_k) \in M_{\text{irreg}}$ and $\mathcal{M}[i, k] = 1$. Since M contains at most d errors, there are at most d such indices k in total. Let K denote the set of these indices. Note that $j^* \notin K$, since $(c^*, \sigma^*) \in M_{\text{reg}}$, according to our assumption.

We now have established that for each $i \in I$, there exists a k with $(c_k, \sigma_k) \in M_{\text{irreg}}$ and $\mathcal{M}[i, k] = 1$, and $|K| \leq d$. Recall that by definition of the incidence matrix $\mathcal{M}$, we have for all $i \in [\text{rows}(\mathcal{M})]$ and $j \in [\text{cols}(\mathcal{M})]$:

$$\mathcal{M}[i, j] = 1 \quad \iff \quad s_i \in B_j.$$

Restating the fact from the above paragraph using this equivalence yields that for all i with $s_i \in B_{j^*}$, there exists a k with $s_i \in B_k$, where there are at most d distinct indices $k \in K$ in total. But this means that $B_{j^*} \subseteq \bigcup_{k \in K} B_k$, where the union is over at most d different subsets B_k of $\mathcal{S}$. This is a direct contradiction to the d-cover-freeness of $\mathcal{F}$, so our assumption must be false, and we must therefore have $R \subseteq V$. □

Compression Ratio. Let C be a claim sequence of length $n \in \mathbb{N}$, and τ be an aggregate signature regular for C. We assume in the following that the length of all signatures of the underlying scheme Σ' is bounded by a constant s and is at least 1. Then the compression ratio of our scheme is $\rho(n) = \frac{n}{\text{rows}(\mathcal{M})}$, since

$$\frac{n}{\mathsf{size}(\tau)} \leq \frac{n}{\text{rows}(\mathcal{M}) \cdot s} \in \mathcal{O}(\rho(n)) \quad \text{and} \quad \frac{n}{\mathsf{size}(\tau)} \geq \frac{n}{\text{rows}(\mathcal{M})} \in \Omega(\rho(n)). \quad (3)$$

Clearly, the compression ratio $\rho(n)$ of our scheme is less than 1 if $n < \text{rows}(\mathcal{M})$, and the resulting aggregate signature is larger than the sum of the individual signature sizes when only few signatures have been aggregated so far. Our scheme can

be easily adapted to fix this behavior, by simply storing all individual signatures instead of immediately aggregating them, until $n = \text{rows}(\mathcal{M})$. When the $n+1$-st signature is added, the individual signatures are aggregated using the aggregation algorithm defined above. When further signatures are added, the size of the aggregate signature remains bounded by $\text{rows}(\mathcal{M}) \cdot s$.

4.1 Achieving Unbounded Aggregation

In order to achieve unbounded aggregation, we do not need just one cover-free family, but a sequence of cover-free families increasing in size, such that we can jump to the next larger one, as soon as we exceed the capacity for the number of aggregatable signatures. This sequence needs to exhibit a monotonicity property, in order to work with our scheme, which we define next.

Definition 5. *We consider a family $(\mathcal{M}^{(l)})_l$ of incidence matrices of corresponding d-cover-free families $(\mathcal{F}_l)_l := (\mathcal{S}_l, \mathcal{B}_l)_l$, where* $\text{rows}(l)$ *denotes the number of rows and* $\text{cols}(l)$ *denotes the number of columns of $\mathcal{M}^{(l)}$. $(\mathcal{M}^{(l)})_l$ is a* monotone family *of incidence matrices of $(\mathcal{F}_l)_l$, if $\mathcal{S}_l \subseteq \mathcal{S}_{l+1}$, $\mathcal{B}_l \subseteq \mathcal{B}_{l+1}$, $l \geq 1$, s.t. $\mathcal{S}_{l+1} = \{s_1, \ldots, s_{\text{rows}(l)}, s_{\text{rows}(l)+1}, \ldots, s_{\text{rows}(l+1)}\}$ and $\mathcal{B}_{l+1} = \{B_1, \ldots, B_{\text{cols}(l)}, B_{cols(l)+1}, \ldots, B_{\text{cols}(l+1)}\}$, where $\mathcal{S}_l = \{s_1, \ldots, s_{\text{rows}(l)}\}$ and $\mathcal{B}_l = \{B_1, \ldots, B_{\text{cols}(l)}\}$.*

Note that Definition 5 implies that

$$\mathcal{M}^{(l+1)} = \begin{pmatrix} \mathcal{M}^{(l)} & \mathbf{A} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}$$

where for $i = 1, \ldots, \text{rows}(l), j = \text{cols}(l)+1, \ldots, \text{cols}(l+1)$

$$\mathbf{A}[i,j] = \begin{cases} 1, & \text{if } s_i \in B_j, \\ 0, & \text{otherwise} \end{cases}$$

and for $i = \text{rows}(l), \ldots, \text{rows}(l+1), j = \text{cols}(l)+1, \ldots, \text{cols}(l+1)$

$$\mathbf{B}[i,j] = \begin{cases} 1, & \text{if } s_i \in B_j, \\ 0, & \text{otherwise.} \end{cases}$$

So, each $\mathcal{M}^{(l)}$ contains all previous $\mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(l-1)}$.

Now, we are able to achieve unbounded aggregation, i.e. our construction is able to aggregate an arbitrary number of signatures, by replacing the fixed incidence matrix $\mathcal{M}$ of a d-CFF in our construction with a monotone family of incidence matrices $(\mathcal{M}^{(l)})_l$. For this, a run of our aggregation algorithm Agg on inputs C_1, C_2, τ_1, τ_2 first has to determine the smallest l, such that $\text{cols}(l) \geq \max(|C_1|, |C_2|)$ and then proceeds with the corresponding incidence matrix $\mathcal{M}^{(l)}$. Analogously, our verification algorithm Verify on inputs C, τ first determines the smallest l such that $\text{cols}(l) \geq |C|$.

Compression Ratio. The compression ratio of our unbounded scheme is $\rho(n) = n/\text{rows}(l)$, where l is the minimum index such that $\text{cols}(l) \geq n$.

4.2 Additional Features of Our Construction

Selective Verification. Let τ be a regular signature with corresponding claim sequence $C = (c_1, \ldots, c_n)$. Assume we would want to know whether a signature for a specific claim c^* was aggregated into τ, but we want to avoid verifying all the claims in C to save verification time, especially if C is large. It is a unique feature of our fault-tolerant aggregate signature scheme that there is an additional algorithm $\mathsf{SelectiveVerify}(C, \tau, c^*)$ that outputs the number of occurrences of c^* in C that have a valid signature in τ, i.e., the number of occurrences of c^* in $\mathsf{Verify}(C, \tau)$, while being faster than actually calling $\mathsf{Verify}(C, \tau)$.

Let Σ be the aggregate signature scheme with list verification defined above and Σ' be the underlying aggregate signature scheme. Then SelectiveVerify works as follows. First, it determines the set J of indices j where c^* occurs in C, i.e. $c_j = c^*$. Then it determines the set $I := \{i \in \text{rows}(\mathcal{M})\colon \mathcal{M}[i, j] = 1 \text{ for a } j \in J\}$, i.e. the set of indices of all rows where an individual signature for c^* should have been aggregated. Then, it initializes $M := ()$ and iterates over all $i \in I$, checking if $b_i := \Sigma'.\mathsf{Verify}(C[\mathcal{M}_i], \tau[i]) = 1$. If this is the case for an i, it sets

$$M := M \sqcup C[\mathcal{M}_i].$$

As soon as M contains $|J|$ occurrences of c^*, SelectiveVerify skips all remaining $i \in I$. After the loop is done, SelectiveVerify outputs the number of occurrences of c^* in M.

Since $\Sigma.\mathsf{Verify}$ returns all claims that are contained in a subsequence $C[\mathcal{M}_i]$ with $b_i = 1$, the output of SelectiveVerify is exactly the number of occurrences of c^* in $\Sigma.\mathsf{Verify}$. SelectiveVerify therefore inherits the fault-tolerance and security properties already proven for $\Sigma.\mathsf{Verify}$.

In the best case, SelectiveVerify requires only one call to the underlying verification algorithm $\Sigma'.\mathsf{Verify}$. In the worst case, it still only requires $|I| \leq \sum_{j \in J} |B_j|$ calls to $\Sigma'.\mathsf{Verify}$, where B_j is the set from the cover-free family corresponding to column j.

Going a little further, it is even possible to create a "subsignature" for c^* that allows everyone to check that c^* has a valid signature without requiring the complete claim sequence C and the complete signature τ: It is sufficient to give $C' := \bigsqcup_{i \in I} C[\mathcal{M}_i]$ and the signatures $\tau[i]$ for $i \in I$ to the verifier.

5 A Concrete Instantiation of Our Scheme

In this section, we consider a concrete construction of a d-CFF which can be used to instantiate our generic d-fault-tolerant aggregate signature scheme. There are several d-CFF constructions in the literature, for instance, constructions based on concatenated codes [LVY01, DMR00b, DMR00a], polynomials,

algebraic-geometric Goppa codes as well as randomized constructions [KRS99]. The following theorem gives a lower bound for the number of rows of the incidence matrix in terms of parameter d and the number of columns. Proofs can be found in [DR82, Fur96, Rus94].

Theorem 3. *For a d-CFF $\mathcal{F} = (\mathcal{S}, \mathcal{B})$, where $|\mathcal{S}| = m$, $|\mathcal{B}| = n$, it holds*

$$m \geq c \frac{d^2}{\log d} \log n$$

for some constant $c \in (0, 1)$.

In the following construction we use for concreteness only a single incidence matrix, but the next lemma by [LVW06] shows a generic construction to get a monotone family of incidence matrices.

Lemma 1. *If $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ and $\mathcal{F}' = (\mathcal{S}', \mathcal{B}')$ are d-CFFs, then there exist a d-CFF $\mathcal{F}^* = (\mathcal{S}^*, \mathcal{B}^*)$ with $|\mathcal{S}^*| = |\mathcal{S}| + |\mathcal{S}'|$ and $|\mathcal{B}^*| = |\mathcal{B}| + |\mathcal{B}'|$.*

Proof. Suppose $\mathcal{M}$ and $\mathcal{M}'$ are the incidence matrices of d-CFFs $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ and $\mathcal{F}' = (\mathcal{S}', \mathcal{B}')$, respectively. Then

$$\mathcal{M}^* = \begin{pmatrix} \mathcal{M} & \mathbf{0} \\ \mathbf{0} & \mathcal{M}' \end{pmatrix}$$

is an incidence matrix for a d-CFF $\mathcal{F}^* = (\mathcal{S}^*, \mathcal{B}^*)$ with $|\mathcal{S}^*| = |\mathcal{S}| + |\mathcal{S}'|$ and $|\mathcal{B}^*| = |\mathcal{B}| + |\mathcal{B}'|$. □

For our approach we could use a deterministic construction of a d-CFF based on polynomials like [KRS99] did in the following way and for which we propose a generalization to the multivariate case in Appendix B.

For our d-CFF $\mathcal{F} = (\mathcal{S}, \mathcal{B})$ let $\mathbb{F}_q = \{x_1, \ldots, x_q\}$ be a finite field and

$$\mathcal{S} := \mathbb{F}_q^2 = \{(x_i, x_j) \colon i, j = 1, \ldots, q\}, \text{ with } |\mathcal{S}| = q^2.$$

For ease of presentation, we assume that q is a prime (as opposed to a prime power), so we may write $\mathbb{F}_q = \{0, \ldots, q-1\}$. We consider the set of all univariate polynomials $f \in \mathbb{F}_q[X]$ of degree at most k, denoted by $\mathbb{F}_q[X]_{\leq k}$. So,

$$\mathbb{F}_q[X]_{\leq k} := \left\{ a_k X^k + \cdots + a_1 X + a_0 \colon a_i \in \mathbb{F}_q,\ i = 0, \ldots, k \right\}.$$

We have $|\mathbb{F}_q[X]_{\leq k}| = q^{k+1}$. Now, for every $f \in \mathbb{F}_q[X]_{\leq k}$, we consider the subsets

$$B_f = \{(x_1, f(x_1)), \ldots, (x_q, f(x_q))\} \subset \mathcal{S} \text{ of size } q,$$

consisting of all tuples $(x, y) \in \mathcal{S}$ which lie on the graph of $f \in \mathbb{F}_q[X]_{\leq k}$, i.e. for which $f(x) = y$. From this we obtain

$$\mathcal{B} := \{B_f \colon f \in \mathbb{F}_q[X]_{\leq k}\}, \text{ which is of size } q^{k+1}.$$

For any distinct $B_f, B_{f_1}, \ldots, B_{f_d} \in \mathcal{B}$ it holds that

$$|B_f \cap B_{f_i}| \leq k,$$

since the degree of each polynomial $g_i := f - f_i$ is at most k and hence they have at most k zeros. Thus, we have

$$\left| B_f \setminus \bigcup_{i=1}^{d} B_{f_i} \right| \geq q - d \cdot k$$

To achieve a d-CFF with this construction, $q \geq d \cdot k + 1$ must be fulfilled.

Now, we consider the incidence matrix $\mathcal{M}$ of a d-CFF, which consists of $|\mathcal{S}|$ rows and $|\mathcal{B}|$ columns. Each row corresponds to an element of $\mathcal{S}$ and each column to an element of $\mathcal{B}$. In the construction above each row corresponds to a tuple $(x, y) \in \mathbb{F}_q^2$, where the order is $(0,0), (0,1), \ldots, (q-1, q-1)$. In the following, let (x_i, y_i) denote the corresponding tuple for row i, $i = 0, \ldots, q^2 - 1$. We start counting from 0 for simplicity, hence,

$$\begin{array}{lcl}
(x_0, y_0) = (0,0), & \ldots, & (x_{q-1}, y_{q-1}) = (0, q-1), \\
(x_q, y_q) = (1,0), & \ldots, & (x_{2q-1}, y_{2q-1}) = (1, q-1), \\
 & \ddots & \\
(x_{q^2-q}, y_{q^2-q}) = (q-1, 0), & \ldots, & (x_{q^2-1}, y_{q^2-1}) = (q-1, q-1).
\end{array}$$

Each column of the incidence matrix $\mathcal{M}$ corresponds to a polynomial of degree at most k, where we decide to start with constant polynomials and end with polynomials of degree k, i.e. $f_0 := 0$, $f_1 := 1$, $f_2 := 2, \ldots, f_q := X$, $f_{q+1} := X + 1$, $f_{q+2} := X + 2, \ldots, f_{2q} := 2X$, $f_{2q+1} := 2X + 1, \ldots, f_{q^{k+1}-1} := (q-1)X^k + (q-1)X^{k-1} + \cdots + (q-1)X + q - 1$.

By f_j we will denote the corresponding polynomial for column j, for $j = 0, \ldots, q^{k+1} - 1$, again starting from 0. Now, the incidence matrix is built as

$$\mathcal{M}[i,j] = \begin{cases} 1, & \text{if } f_j(x_i) = y_i, \\ 0, & \text{otherwise.} \end{cases}$$

Example. For $q = 5$, $d = 2$ and $k = 2$ we have a 2-CFF with

$$\mathcal{S} = \{(0,0), (0,1), \ldots, (4,3), (4,4)\}, \ |\mathcal{S}| = 25.$$

We have

$$\mathcal{B} = \{B_{f_0}, \ldots, B_{f_{124}}\},$$

since $|\mathbb{F}_5[X]_{\leq 2}| = 5^3 = 125$, where

$$B_{f_j} = \{(0, f_j(0)), (1, f_j(1)), \ldots, (4, f_j(4))\}, \quad |B_{f_j}| = 5, \quad j = 0, \ldots, 5^3 - 1$$

and $f_0 := 0$, $f_1 := 1, \ldots, f_{124} := 4X^2 + 4X + 4$. Thus, we obtain our incidence matrix $\mathcal{M}$:

$$\begin{array}{c} \\ (0,0) \\ (0,1) \\ \vdots \\ (4,4) \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & \ldots & X & \ldots & 4X^2+4X+4 \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 1 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & \ldots & 1 \end{pmatrix} \end{array} = \mathcal{M}$$

Remark. With this univariate polynomial-based construction of a d-CFF it is very easy to generate our incidence matrix or only some parts of it, which we need for our verification algorithm or if we want to check some information separately.

If, for example, one is interested to verify the validity of only one single claim–signature pair (c_j, σ_j) in an aggregate signature, it is not necessary to generate the whole matrix but only the rows where the related column j has 1-entries. So, you have to know which polynomial corresponds to column j.

For this, we can use the fact, that each positive number $n = 0, \ldots, q^{k+1} - 1$ can be written as $a_k \cdot q^k + a_{k-1} \cdot q^{k-1} + \cdots + a_0$, where $a_k, \ldots, a_0 \in \{0, \ldots, q-1\}$. So, each n corresponds to a $(k+1)$-tuple denoted by $(a_k^{(n)}, \ldots, a_0^{(n)})$. For the sake of convenience, we start to count the rows and columns of our matrix by 0, as before. Thus, for column $j = 0, \ldots, q^{k+1} - 1$ we assign the polynomial $f_j = a_k^{(j)} X^k + \cdots + a_0^{(j)}$.

Analogously, for each row $i = 0, \ldots, q^2 - 1$, we assign the tuple $(b_1^{(i)}, b_0^{(i)}) \in \mathbb{F}_q^2$, where $i = b_1^{(i)} \cdot q + b_0^{(i)}$. Let $I'_j \subset \{0, \ldots q^2 - 1\}$ be the subset of all rows i' where $f_j(b_1^{(i')}) = b_0^{(i')}$. So, it suffices to generate only the rows $i' \in I'_j$ to verify the validity of σ_j. To get the 1-entries of these rows, you have to check for each $i' \in I'_j$ which polynomials $f \in \mathbb{F}_q[X]_{\leq k}$ fulfill $f(b_1^{(i')}) = b_0^{(i')}$. For all arbitrary, but fixed values $a_k, \ldots a_1 \in \{0, \ldots, q-1\}$ compute an appropriate a_0. This results in q^k polynomials, accordingly columns, per row. If the coefficients of the appropriate polynomials are known then we can use them to compute the number of the corresponding columns with 1-entries.

Compression Ratio of Our Bounded Scheme. If our bounded scheme is instantiated with this CFF, and we assume that the length of signatures of the underlying scheme Σ' is bounded by a constant s, then, as shown in (3), the compression ratio is

$$\rho(n) = \frac{n}{\text{rows}(\mathcal{M})} = \frac{n}{|\mathcal{S}|} = \frac{n}{q^2}\,.$$

For $n = |\mathcal{B}|$, we therefore have

$$\rho(n) = \frac{|\mathcal{B}|}{|\mathcal{S}|} = \frac{q^{k+1}}{q^2}\,.$$

Since $q \geq dk + 1$, we have that $|\mathcal{B}|$ grows exponentially in k, whereas $|\mathcal{S}|$ grows only quadratically in k. Hence, $|\mathcal{B}|$ is exponential in $|\mathcal{S}|$, or, stated differently, $|\mathcal{S}|$ is logarithmic in $|\mathcal{B}|$.

Compression Ratio of Our Unbounded Scheme. When our unbounded scheme is instantiated with the monotone family of CFFs obtained by fixing an incidence matrix $\mathcal{M}$ and repeatedly using Lemma 1 on $\mathcal{M}$, then the asymptotic compression ratio is $\rho(n) = 1$, since

$$\frac{n}{\text{rows}(l)} \leq \frac{\text{cols}(l)}{\text{rows}(l)} = \frac{\text{cols}(\mathcal{M})}{\text{rows}(\mathcal{M})} \quad \text{for all } l,$$

which is constant. Therefore, the size of an aggregate signature is linear in the length of the claim sequence.

However, if we assume that all signatures of the underlying scheme Σ' have a size bounded by s, then the concrete size of an aggregate signature is at most

$$\begin{aligned} \text{rows}(l)s &\leq l\,\text{rows}(\mathcal{M})s \\ &\leq (n/\text{cols}(\mathcal{M}) + 1)\,\text{rows}(\mathcal{M})s \\ &= \left(\frac{\text{rows}(\mathcal{M})}{\text{cols}(\mathcal{M})}n + \text{rows}(\mathcal{M})\right)s\,, \end{aligned}$$

since $\text{rows}(l) = l\,\text{rows}(\mathcal{M})$ for the construction of the monotone family of CFFs, and $l = \lceil n/\text{cols}(\mathcal{M})\rceil \leq n/\text{cols}(\mathcal{M}) + 1$.

Therefore we see that the length of the aggregate signature is linear in n, but the factor $\text{rows}(\mathcal{M})/\text{cols}(\mathcal{M})$ can be made arbitrarily small by choosing a proper CFF, such as the one described above.

It is an interesting open problem to construct an unbounded fault-tolerant scheme with better compression ratio, for example by finding a better monotone family of CFFs. A generalization of the above construction to multivariate polynomials, which might be advantageous in some scenarios, is given in Appendix B.

Example Instantiations. Table 1 shows parameters of several cover-free families based on the construction described in this section. For each of the rows given there, there is an instance of our fault-tolerant signature scheme that can compress signatures for up to n claims to a vector of m aggregates, while tolerating

Table 1. Example parameters for cover-free families.

q	k	d	$m = \lvert\mathcal{S}\rvert$	$n = \lvert\mathcal{B}\rvert$
5	2	2	25	125
11	2	5	121	1331
17	2	8	289	4913
17	4	4	289	$\approx 1.42 \cdot 10^6$
29	2	14	841	24389
53	2	26	2809	148877
101	2	50	10201	$\approx 1.03 \cdot 10^6$
251	3	83	63001	$\approx 3.97 \cdot 10^9$
1021	2	510	1042441	$\approx 1.06 \cdot 10^9$

up to d errors. (The numbers q and k are needed for the instantiation of the CFF, but do not immediately reflect a property of our fault-tolerant aggregate signature scheme.) Of course, our scheme can be instantiated with different parameters and completely different constructions of CFFs as well.

Acknowledgements. We wish to thank our colleague and friend Julia Hesse for raising the initial research question that led to this work. We would also like to thank the anonymous reviewers for their helpful comments.

A Discussion on Signature Size

A typical requirement for aggregate signatures is that the length of an aggregate signature is the same as that of any of the individual signatures [HSW13]. Also, the number of signatures that can be aggregated into a single signature should be unbounded.

We show that these goals are mutually exclusive for an "ideal" fault-tolerant aggregate signature schemes if one wishes to maintain a constant $d \geq 1$.

Proposition 1. *Let $n, d \in \mathbb{N}$, and Σ be a d-fault-tolerant signature scheme. Assume that $\Sigma.\mathsf{Verify}(C, \tau) = R$ for all claim sequences C and corresponding signatures τ constructed from an arbitrary multiset $M = \{(c_1, \tau_1), \ldots, (c_n, \tau_n)\}$ of n claim–signature pairs and containing at most d errors, and where R is the multiset of all claims c_i accompanied by a regular signature τ_i in M. Then we have $|\tau| \geq \Omega(\log_2 n)$ as a function of n, where d is considered constant, and $|\tau|$ is the length of the signature τ in bits.*

Proof. Call an output O of $\Sigma.\mathsf{Verify}$ *in accordance with* C, if O is a sub-multiset of $\mathsf{elem}(C)$ and $|O| \geq |\mathsf{elem}(C)| - d$.

Now, let $n, d, \Sigma, C, \tau, M, R$ be as in the theorem statement. Clearly, since we assumed that Verify always outputs R, $\Sigma.\mathsf{Verify}$'s output must be in accordance with C. For a fixed number of errors $i \in \{0, \ldots, d\}$, there are $\binom{n}{i}$ distinct outputs in accordance with C. Thus, for up to d errors, there are up to

$$s(n) := \sum_{i=0}^{d} \binom{n}{i} \geq \binom{n}{d} \geq \frac{(n-d)^d}{d!}$$

distinct outputs in accordance with C.

$\Sigma.\mathsf{Verify}$ must use τ to determine the correct output R among the set of outputs in accordance with C. If the signature size $|\tau|$ is at most $l \in \mathbb{N}$ bits, then $\Sigma.\mathsf{Verify}$ can distinguish at most 2^l cases based on τ. Thus, we must have $2^l \geq s(n)$, or, equivalently,

$$l \geq \log_2 s(n) \geq \log_2 \frac{(n-d)^d}{d!} = d \log_2(n-d) - \log_2(d!) \in \Omega(\log_2 n) \ .$$

This concludes the proof. □

Note that the assumption that $\Sigma.\mathsf{Verify}(C,\tau) = R$ is somewhat artificial: We assume an *ideal* d-fault-tolerant signature scheme, where $\Sigma.\mathsf{Verify}$ always "magically" outputs the correct multiset R, when called with a claim sequence containing n claims.

On the one hand, $\Sigma.\mathsf{Verify}(C,\tau) \supseteq R$ is required by the d-fault-tolerance of Σ. Intuitively, one would expect the other $\Sigma.\mathsf{Verify}(C,\tau) \subseteq R$ direction to follow from the security of Σ. However, this does not appear to follow in general, due to two reasons:

The first reason is that security is only required against adversaries that have running time polynomial in κ, i.e. adversaries that can create at most a polynomial number of claims.

The second reason is that if for two fixed C, τ there is a claim $c = (\mathsf{pk}, m)$ in $\Sigma.\mathsf{Verify}(C,\tau)$ that is not in R, then this does only violate the security definition if the challenge public key randomly drawn by the security experiment happens to be equal to pk by chance.

B Cover-Free Families Using Multivariate Polynomials

For our polynomial based construction, we can also use multivariate polynomials $f \in \mathbb{F}_q[X_1,\ldots,X_t]$, $t \in \mathbb{N}$, of degree at most k. Each multivariate polynomial f with degree $\leq k$ consists of monomials in terms of $a_{i_1,\ldots,i_t} X_1^{i_1} \cdots X_t^{i_t}$, where $a_{i_1,\ldots,i_t} \in \mathbb{F}_q$ and $i_1 + \cdots + i_t \leq k$. We denote by $\mathbb{F}_q[X_1,\ldots,X_t]_{\leq k}$ the set of all multivariate polynomials $f \in \mathbb{F}_q[X_1,\ldots,X_t]$ of degree at most k, i.e.

$$\mathbb{F}_q[X_1,\ldots,X_t]_{\leq k} := \left\{ \sum_{i_1+\cdots+i_n \leq k} a_{i_1,\ldots,i_t} X_1^{i_1} \cdots X_t^{i_t} \colon a_{i_1,\ldots,i_t} \in \mathbb{F}_q \right\}.$$

For the maximal number of monomials of degree exactly k, we obtain $\binom{t+k-1}{k}$. Hence, for degree at most k, we have

$$\sum_{i=0}^{k} \binom{t+i-1}{i} = \binom{t+k}{k}, \text{ and hence, } |\mathbb{F}_q[X_1,\ldots,X_t]_{\leq k}| = q^{\binom{t+k}{k}}.$$

We can now define

$$B_f := \left\{ (\boldsymbol{x}, f(\boldsymbol{x})) \colon \boldsymbol{x} \in \mathbb{F}_q^t \right\}, \text{ with } |B_f| = q^t,$$

and

$$\mathcal{B} := \left\{ B_f \colon f \in \mathbb{F}_q[X_1,\ldots,X_t]_{\leq k} \right\}, \text{ with } |\mathcal{B}| = q^{\binom{t+k}{k}}.$$

Now, we set

$$\mathcal{S} := \mathbb{F}_q^{t+1}, \text{ which is of size } q^{t+1}.$$

The number of zeros is at most $k \cdot q^{t-1}$ and thus, for different $B_f, B_{f_1}, \ldots, B_{f_d} \in \mathcal{B}$ it holds

$$\left| B_f \setminus \bigcup_{i=1}^{d} B_{f_i} \right| \geq q^t - d \cdot k \cdot q^{t-1}.$$

To achieve a d-CFF with this construction, $q^t \geq d \cdot k \cdot q^{t-1} + 1$ must be fulfilled.

Compression Ratio of Our Bounded Scheme. If our bounded scheme is instantiated with this multivariate CFF, and we assume for simplicity, that the size of signatures of the underlying scheme Σ' is bounded by a constant, then as shown in (3), the compression ratio is

$$\rho(n) = \frac{n}{\text{rows}(\mathcal{M})} = \frac{n}{|\mathcal{S}|} = \frac{n}{q^{t+1}} \quad .$$

For $n = |\mathcal{B}|$, we therefore have

$$\rho(n) = \frac{|\mathcal{B}|}{|\mathcal{S}|} = \frac{q^{\binom{t+k}{k}}}{q^{t+1}} \quad .$$

Compression Ratio of Our Unbounded Scheme. By using Lemma 1 on $\mathcal{M}$ we can also obtain a monotone CFF based on multivariate polynomials and use it to instantiate our unbounded scheme. The discussion about the compression ratio of the unbounded scheme in Sect. 5 also applies to this instantiation.

References

[AGH10] Ahn, J.H., Green, M., Hohenberger, S.: Synchronized aggregate signatures: new definitions, constructions and applications. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) CCS 2010, pp. 473–484. ACM Press, October 2010

[BGR14] Brogle, K., Goldberg, S., Reyzin, L.: Sequential aggregate signatures with lazy verification from trapdoor permutations. Inf. Comput. **239**, 356–376 (2014). doi:10.1016/j.ic.2014.07.001

[BJ10] Bagherzandi, A., Jarecki, S.: Identity-based aggregate and multi-signature schemes based on RSA. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 480–498. Springer, Heidelberg (2010)

[BN07] Bellare, M., Neven, G.: Identity-based multi-signatures from RSA. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 145–162. Springer, Heidelberg (2006)

[Bol+07] Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) CCS 2007, pp. 276–285. ACM Press, October 2007

[Bon+03] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003, vol. 2656. LNCS, pp. 416–432. Springer, Heidelberg (2003)

[Cra+07] Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)

[DMR00a] Dyachkov, A.G., Macula, A.J., Rykov, V.V.: New applications and results of superimposed code theory arising from the potentialities of molecular biology. In: Althöfer, I., Cai, N., Dueck, G., Khachatrian, L., Pinsker, M.S., Sárközy, A., Wegener, I., Zhang, Z. (eds.) Numbers, Information and Complexity, pp. 265–282. Springer, Heidelberg (2000)

[DMR00b] Dyachkov, A.G., Macula, A.J., Rykov, V.V.: New constructions of superimposed codes. IEEE Trans. Inf. Theory **46**(1), 284–290 (2000). doi:10.1109/18.817530

[Dod+02] Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)

[DR82] Dyachkov, A.G., Rykov, V.V.: Bounds on the length of disjunctive codes. Problemy Peredachi Informatsii **18**(3), 7–13 (1982)

[Fur96] Füredi, Z.: On r-cover-free families. J. Comb. Theory, Ser. A **73**(1), 172–173 (1996). doi:10.1006/jcta.1996.0012

[Ger+12] Gerbush, M., Lewko, A., O'Neill, A., Waters, B.: Dual form signatures: an approach for proving security from static assumptions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 25–42. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_4

[GR06] Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)

[Her06] Herranz, J.: Deterministic identity-based signatures for partial aggregation. Comput. J. **49**(3), 322–330 (2006). doi:10.1093/comjnl/bxh153

[HJK11] Hofheinz, D., Jager, T., Kiltz, E.: Short signatures from weaker assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011)

[HK04] Heng, S.-H., Kurosawa, K.: k-resilient identity-based encryption in the standard model. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 67–80. Springer, Heidelberg (2004)

[HKW15] Hohenberger, S., Koppula, V., Waters, B.: Universal signature aggregators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 3–34. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_1

[HSW13] Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_27

[KRS99] Kumar, R., Rajagopalan, S., Sahai, A.: Coding constructions for blacklisting problems without computational assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)

[KS64] Kautz, W.H., Singleton, R.C.: Nonrandom binary superimposed codes. IEEE Trans. Inf. Theory **10**(4), 363–377 (1964). doi:10.1109/TIT.1964.1053689

[LLY15] Lee, K., Lee, D.H., Yung, M.: Sequential aggregate signatures with short public keys without random oracles. Theor. Comput. Sci. **579**, 100–125 (2015). doi:10.1016/j.tcs.2015.02.01923

[Lu+06] Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)

[LVW06] Li, P., Van Rees, G., Wei, R.: Constructions of 2-cover-free families and related separating hash families. J. Comb. Des. **14**(6), 423–440 (2006)

[LVY01] Lebedev, V., Vilenkin, P., Yekhanin, S.: Cover-free families and superimposed codes: constructions, bounds, and applications to cryptography and group testing. In: IEEE International Symposium on Information Theory (2001)

[Lys+04] Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)

[MT09] Ma, D., Tsudik, G.: A new approach to secure logging. TOS **5**(1) (2009). doi:10.1145/1502777.1502779

[Nev08] Neven, G.: Efficient sequential aggregate signed data. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Heidelberg (2008)

[Rus94] Ruszinkó, M.: On the upper bound of the size of the r-cover-free families. J. Comb. Theory, Ser. A **66**(2), 302–310 (1994). doi:10.1016/0097-3165(94)90067-1

[Sch11] Schröder, D.: How to aggregate the CL signature scheme. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 298–314. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23822-2_17

[STW97] Stinson, D.R., Trung, T.V., Wei, R.: Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. J. Stat. Plann. Infer. **86**, 595–617 (1997)

[SW99] Safavi-Naini, R., Wang, H.: Multireceiver authentication codes: models, bounds, constructions, and extensions. Inf. Comput. **151**(1–2), 148–172 (1999). doi:10.1006/inco.1998.2769

[TS06] Tonien, D., Safavi-Naini, R.: An efficient single-key pirates tracing scheme using cover-free families. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 82–97. Springer, Heidelberg (2006)

[ZS11] Zaverucha, G.M., Stinson, D.R.: Short one-time signatures. Adv. Math. Comm. **5**(3), 473–488 (2011). doi:10.3934/amc.2011.5.473

Delegatable Functional Signatures

Michael Backes[1,2], Sebastian Meiser[1(✉)], and Dominique Schröder[1]

[1] CISPA, Saarland University, Saarbrücken, Germany
{backes,meiser}@cs.uni-saarland.de
[2] MPI-SWS, Kaiserslautern, Germany

Abstract. We introduce *delegatable functional signatures* (DFS) which support the delegation of signing capabilities to another party, called the *evaluator*, with respect to a functionality $\mathcal{F}$. In a DFS, the signer of a message can choose an evaluator, specify how the evaluator can modify the signature without voiding its validity, allow additional input, and decide how the evaluator can further delegate its capabilities. Technically, DFS unify several seemingly different signature primitives, including functional signatures and policy-based signatures (PKC'14), sanitizable signatures, identity based signatures, and blind signatures. We characterize the instantiability of DFS with respect to the corresponding security notions of unforgeability and privacy. On the positive side we show that privacy-free DFS can be constructed from one-way functions. Furthermore, we show that unforgeable and private DFS can be constructed from doubly-enhanced trapdoor permutations. On the negative side we show that the previous result is optimal regarding its underlying assumptions presenting an impossibility result for unforgeable private DFS from one-way permutations.

1 Introduction

Digital signature schemes resemble the idea of a hand written signature in the sense that a signer signs messages with his private key sk_{sig} and anybody can check the validity of the signature using the corresponding public key pk_{sig}. The elementary security property is unforgeability under chosen message attacks which says that an adversary cannot compute a signature on a fresh message, even if he has observed q signatures on q messages of his choice [31]. This security definition models the idea of *non*-malleability for digital signatures: The adversary should not be able to modify any signature such that it verifies for a different message.

For many emerging applications, such as the delegation of computation on authenticated data, the basic notion is insufficient and a controlled form of malleability would be desirable. Consider as an example a company S that wishes to outsource the computation on authenticated data to untrusted parties (resp. to parties that may further delegate the computation to sub-contractors), called the evaluators, without handing out any secret key. For each data, S chooses a set of allowed functions $\mathcal{F}$ that can be applied. The evaluators are organized hierarchically, where each evaluator receives an intermediate result and can compute

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 357–386, 2016.
DOI: 10.1007/978-3-662-49384-7_14

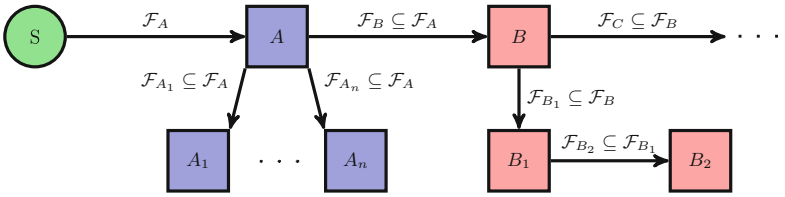


Fig. 1. Example hierarchical application of DFS.

any function $f \in \mathcal{F}$ chosen by S, or delegate subsets $\mathcal{F}' \subseteq \mathcal{F}$ to other evaluators. We allow S to restrict the number of delegations, as well as the order in which functions can be applied. The chain computation should be publicly verifiable which means that everybody can verify that:

- The computation was based on the original data of S.
- Only the functions chosen by S were applied to the data (in the right order, if specified).
- Any delegation of computation by an evaluator A to another evaluator (a third party B or any sub-contractor A_i) has been authorized by S and A.

We refer to Fig. 1 for an example structure of delegation. We consider the following security notions: *Unforgeability* says that malicious evaluators can only apply the functions(s) they were allowed to apply, e.g.: B_1 can only apply $\mathcal{F}_{B_1} \subseteq \mathcal{F}_B \subseteq \mathcal{F}_A$ and further delegate sets of functions $\mathcal{F}_{B_2} \subseteq \mathcal{F}_{B_1}$. *Privacy* says that given the result of a computation, it is not possible to gain information about the computed functions or their input (or the parties that did the computation): To an observer, the signature σ_{B_2} computed by B_2 for a message m_{B_2} is indistinguisable from a signature σ for m_{B_2} computed by S. Privacy is a useful and desirable property in many applications as it hides the business structure of the (sub-)contractor and still allows to verify the correctness of the computation. Traditional signature schemes as well as their malleable variants are not suitable in this setting. In this paper we close this gap by introducing the concept of *delegatable functional signatures*.

1.1 Our Contribution

Our main contributions are as follows. First, we introduce *delegatable functional signatures* (DFS). This primitive supports highly controlled, fine-grained delegation of signing capabilities to designated third parties and is general enough to cover several malleable signature schemes. Second, we present strong security notions for unforgeability and privacy that also take into account insider adversaries. Third, we provide a complete characterization regarding the achievability of our security notions based on general complexity assumptions. In the following we discuss each contribution comprehensively.

Delegatable Functional Signatures. Delegatable functional signatures support the delegation of signing capabilities to another party, called the *evaluator*,

with respect to a functionality $\mathcal{F}$. The evaluator may compute valid signatures on messages m' and delegate capabilities f' to another evaluator with key k whenever $(f', m') \leftarrow \mathcal{F}(f, \alpha, k, m)$ for a value α of the evaluators choice. Thus, the functionality describes how an evaluator can perform the following two tasks.

Malleability. The designated evaluator can derive a signature on m' from a signature on m, if $(f', m') \leftarrow \mathcal{F}(f, \alpha, k, m)$, where the evaluator picks α and k himself.

Delegatability. The designated evaluator can delegate signing capabilities f' on his signature on m', to other parties, if $(f', m') \leftarrow \mathcal{F}(f, \alpha, k, m)$, where k is the key of another *evaluator* (or his own key, if he wants to apply several functions successively) and where the evaluator picks α himself.

Example 1 (Malleability). Suppose that a sender S wants to sign a document, but allow another entity A to fill in information in a few fields. S chooses f to describe the places where information can be added, as well as which information can be added (e.g., 16 characters) without harming the validity of the signature. A chooses the fields and the information it wishes to fill in by choosing the corresponding value for α, and he derives a signature on m', where $(\cdot, m') \leftarrow \mathcal{F}(f, \alpha, k, m)$.

Example 2 (Delegatability). Suppose that a sender S wants to restrict how A can delegate further capabilities. Her choice of f additionally describes that after filling in information, certain parts of the document can be censored without harming the validity of the signature, but no further information can be added.[1] After filling in information, A may delegate the censoring to B, but impose restrictions on which part of the document may be censored by choosing the corresponding value for α. Then, A and delegates the corresponding capabilities f' to B, where $(f', \cdot) \leftarrow \mathcal{F}(f, \alpha, k_B, m)$.

Our definition also covers signing capabilities for fresh messages. If a sender S wants to give A the capability to sign certain messages in his name, he can simply generate a signature σ_{fresh} for a new (empty) message and use f to specify which capabilities A has, i.e., which signatures he can derive from σ_{fresh}.

Security Model for DFS. A central contribution of this paper are the formal definitions of unforgeability and privacy. On an abstract level, these notions resemble the well known intuition: Unforgeability means that no signatures can be forged, except on messages within a certain class. Privacy means that derived signatures are indistinguishable from fresh signatures. However, finding meaningful and achievable definitions for DFS is rather challenging, because the signatures are malleable by nature and we are also considering the multi-party setting:

[1] If a PKI exists, S can additionally add descriptions of the evaluators that are allowed to do this second round of processing.

Unforgeability: In a DFS scheme the signer specifies for every signature the degree of malleability and how this malleability can be delegated. Unforgeability is then captured by a transitive closure that contains all messages that can trivially be derived.

Privacy: Our notion of privacy follows the idea that all information about signatures should be hidden (except for the message). This is captured in an indistinguishability game where the adversary can hand in a signature of his own. Either this signature is treated exactly as the adversary specifies it (modified by evaluators of his choice, possibly under keys of the adversary possesses) or a new signature for the same (resulting) message is created.

For both unforgeability and privacy we present three different security notions for DFS schemes: The weakest one, unforgeability/privacy against outsider attacks, holds only for adversaries that do not have access to the private key of an evaluator. The second one, unforgeability/privacy against insider attacks, assumes that an evaluator is malicious and possesses a honestly generated evaluator key. The third one, unforgeability/privacy against strong insider attacks assumes a malicious evaluator that might generate its own keys.

Unifying Signature Primitives. Delegatable functional signatures are very versatile and imply several seemingly different signature primitives. These include functional signatures, which were recently introduced by Boyle, Goldwasser, and Ivan [15], policy-based signatures, which were recently introduced by Bellare and Fuchsbauer [11], blind signatures, identity based signatures, sanitizable signatures and redactable signatures.

Instantiability of DFS. We give a complete characterization of the instantiability of DFS from general complexity based assumptions presenting both positive and negative results.

Possibility of DFS. On the positive side we show that DFS can be constructed from one-way functions in a black-box way if one gives up privacy.

Theorem 1 (Possibility, informal). *Unforgeable delegatable functional signatures exist if one-way functions exist.*

Furthermore, we show that unforgeable and private DFS schemes can be constructed from (doubly enhanced) trapdoor permutations in a black-box way. Our scheme shows that our strong definitions for unforgeability and privacy are achievable for arbitrary, efficiently computable, choices of $\mathcal{F}$.

Theorem 3 (Possibility, informal). *Private unforgeable delegatable functional signatures exist if doubly enhanced trapdoor permutations exist.*

Impossibility of DFS. We show that the previous result is optimal w.r.t. the underlying assumptions. We show that unforgeable and private delegatable functional signatures cannot be constructed from one-way functions. The basic idea is to construct a blind signature scheme out of any functional signature scheme

in a black-box way. Recently, Katz, Schröder, and Yerukhimovich have shown that blind signature schemes cannot be build from one-way permutations using black-box techniques only [34]. A construction of DFS based on OWFs would yield a black-box construction of blind signature schemes based on OWFs. However, this would directly contradict the result of [34].

Theorem 2 (Impossibility, informal). *Private unforgeable delegatable functional signatures secure against insider adversaries cannot be constructed from one-way functions in a black-box way.*

1.2 Related Work

(Delegatable) Anonymous Credentials. In anonymous credential systems users can prove the possession of a credential without revealing their identity. We view this very successful line of research as orthogonal to our work: Credentials can be applied on top of a signature scheme in order to prove properties that are specified in an external logic. In fact, one could combine delegatable functional signatures with credentials in order to partially leak the delegation chain, while allowing to issue or modify credentials in an anonymous but controlled way. Anonymous credential systems have been investigated extensively, e.g., [10,16, 17,20–23,28]. The main difference between delegatable anonymous credential schemes, such as [1,9], and our approach is that delegation is done by extending the proof chain (and thus leaking information about the chain). Restricting the properties of the issuer in a credential system has been considered in [8]. However, they only focus on access control proofs and their proof chain is necessarily visible, whereas our primitive allows for privacy-preserving schemes.

Malleable Signature Schemes. A limited degree of malleability for digital signatures has been considered in many different ways (we refer to [24] for a nice overview). We group malleable signature schemes into three categories: publicly modifiable signatures, sanitizable signatures and proxy signatures. Publicly modifiable signatures do not consider a special secret key for modifying signatures, which means that everyone with access to the correct public key and one or more valid message-signature pairs can derive new valid message-signature pairs. There are schemes that allow for redacting signatures [18,33,36,37] that allow for deriving valid signatures on parts (or subsets) of the message m. There are schemes that allow for deriving subset and union relations on signed sets [33], linearly homomorphic signature schemes [5,14,29] and schemes that allow for evaluating polynomial functions [13,25]. Libert et al. combine linearly homomorphic signatures with structure preserving signatures [35]. However, known publicly modifiable signature schemes only consider static functions or predicates (one function or predicate for every scheme) and leave the signer little room for bounding a class of functions to a specific message. As the signatures can be modified by everyone with access to public information, they do not allow for a concept of controlled delegation.

Sanitizable signature schemes [3,19] extend the concept of malleable signatures by a new secret key sk_{San} for the evaluator. Only a party in possession of this key can modify signatures. In general, this primitive allows the signer to specify which blocks of the message can be changed, without restricting the possible content. However, they do not consider delegation and they do not allow for computing arbitrary functions on signed data.

Anonymous Proxy Signatures [30] consider delegation of signing rights in a specific context. For example, the delegator may choose a subset of signing rights for the tasks of quoting. Their notion of privacy makes sure that all delegators remain anonymous. The main difference to our work is that they only allow delegation on the basis of the keys and that they do not support restricting further delegation, whereas we support restricting delegation capabilities depending on each message.

Constructing delegatable anonymous credentials out of malleable signatures was investigated by Chase et al. [27]. The main contribution is an efficient scheme based on malleable zero-knowledge proofs [26] and the question regarding the minimal assumptions was left open.

1.3 Closely Related Work

The general framework by Ahn et al. [2] is versatile and, like delegatable functional signatures, unifies a variety of signature notions. A variety of instantiations can be captured in their framework using their predicate P to describe a complex functionality for deriving signatures. In fact, it seems possible to describe delegatable functional signatures in their framework by encoding the functionality in a complex predicate and by encoding the keys of the evaluators as specifically structured signatures. However, so far there exist no construction for their framework that is capable of dealing with such predicates (their constructions support single element sets $\mathcal{M}$, but to encode our scheme, at least sets of size two are required). Attrapadung et al. [4] discuss and refine this framework. Both works do not explore the minimal computational assumptions.

The works of Boyle et al. [15] (introducing *functional digital signatures*) and of Bellare and Fuchsbauer [11] (introducing policy-based signatures) are closely related to our notion of DFS.

In a functional signature scheme, the signer hands out keys sk_f for functions f to allow the recipient to sign all messages in the range of f. Similar to our contributions, they define notions of unforgeability and privacy (called function privacy) and present several constructions for functional digital signatures. One of their constructions also shows that functional signatures can be build from one-way functions provided that one is willing to give up privacy. They furthermore show how to construct one-round delegation schemes out of a functional digital signature scheme.

While our work is closely related to both works, it differs in several aspects: First, we not only consider the controlled malleability of the signature, but also support the delegation of signing capabilities. Second, while we also show for

our notions that unforgeable-only DFS schemes can be build from one-way functions, we additionally show that private DFS schemes can *not* be constructed from from one-way permutations (see Sect. 4). We believe that our impossibility result should also hold for functional signatures [15], as well as for policy-based signatures [11], because our impossibility result does not rely on the delegation property of our scheme. Furthermore, DFS signatures allow for authenticated chain computations. In the extended version [6] we compare delegatable functional signature schemes to functional digital signature schemes and policy-based signature schemes and show how to construct them out of a delegatable functional signature scheme. Whether the converse is possible is unknown.

2 Delegatable Functional Signatures

Delegatable functional signatures support the delegation of signing capabilities to another party, called the *evaluator*, with respect to a functionality $\mathcal{F}$. The evaluator may compute valid signatures on messages m' and delegate capabilities f' to another evaluator with key k whenever $(f', m') \leftarrow \mathcal{F}(f, \alpha, k, m)$ for a value α of the evaluators choice.

Our definition of DFS limits the delegation capabilities of the evaluator. In particular, the signer specifies how an evaluator may delegate his signing rights.

2.1 Formal Description of a DFS Scheme

A delegatable functional signature (DFS) scheme over a message space $\mathcal{M}$, a key space $\mathcal{K}$, and parameter spaces $\mathcal{P}_f$ and $\mathcal{P}_\alpha$ is a signature scheme that additionally supports a controlled form of malleability and delegation. A DFS is described by a functionality $\mathcal{F} : \mathbb{N} \times \mathcal{P}_f \times \mathcal{P}_\alpha \times \mathcal{K} \times \mathcal{M} \to (\mathcal{P}_f \times \mathcal{M}) \cup \{\perp\}$ that specifies how messages can be changed and how capabilities can be delegated. Once the signer received a message-signature pair, it can compute signatures on messages of its choice (that are legitimate w.r.t. $\mathcal{F}$) and can partially delegate his signing capabilities to another evaluator. We model this property by introducing an algorithm $\mathsf{Eval}_\mathcal{F}$ for evaluating functions on signatures. This algorithm takes as input the parameter α that defines the evaluator's own input to the function f, the message m, and a key pk'_{ev}. The algorithm $\mathsf{Eval}_\mathcal{F}$ outputs a signature σ' on m', where $(f', m') \leftarrow \mathcal{F}(\lambda, f, \alpha, pk'_{ev}, m)$. This new signature σ' can be changed by an evaluator that owns a (possibly different) key sk'_{ev} and this evaluator can transform it further with the new capability f'.

Definition 1 *(Delegatable functional signatures).* *A* delegatable functional signature scheme DFSS *is a tuple of efficient algorithms* $\mathsf{DFSS} = (\mathit{Setup}, \mathit{KGen}_{sig}, \mathit{KGen}_{ev}, \mathit{Sig}, \mathit{Eval}_\mathcal{F}, \mathit{Vf})$ *defined as follows:*

$(\boldsymbol{pp}, \boldsymbol{msk}) \leftarrow \mathbf{Setup}(\lambda)$**:** *The setup algorithm* Setup *outputs public parameters* pp *and a master secret key* msk.

$(\boldsymbol{sk}_{sig}, \boldsymbol{pk}_{sig}) \leftarrow \mathbf{KGen}_{sig}(\boldsymbol{pp}, \boldsymbol{msk})$**:** *The signature key generation algorithm outputs a secret signing key* sk_{sig} *and a public signing key* pk_{sig}.

$(\boldsymbol{sk}_{ev}, \boldsymbol{pk}_{ev}) \leftarrow \mathbf{KGen}_{ev}(\boldsymbol{pp}, \boldsymbol{msk})$**:** *The evaluation key generation algorithm* KGen_{ev} *outputs a secret evaluator key* sk_{ev} *and a public evaluator key* pk_{ev}.

$\sigma \leftarrow \mathbf{Sig}(\boldsymbol{pp}, \boldsymbol{sk}_{sig}, \boldsymbol{pk}_{ev}, f, m)$**:** *The signing algorithm* Sig *outputs a signature* σ *on* m*, on which functions from the class* f *can be applied (or an error symbol* $\perp$*).*

$\hat{\sigma} \leftarrow \mathbf{Eval}_{\mathcal{F}}(\boldsymbol{pp}, \boldsymbol{sk}_{ev}, \boldsymbol{pk}_{sig}, \alpha, m, \boldsymbol{pk}'_{ev}, \sigma)$**:** *The evaluation algorithm outputs a derived signature* $\hat{\sigma}$ *for* m' *on the capability* f'*, that can be modified using the evaluator key* sk'_{ev} *associated with* pk'_{ev}*, where* $(f', m') \leftarrow \mathcal{F}(\lambda, f, \alpha, pk'_{ev}, m)$ *(or an error symbol* $\perp$*).*

$b \leftarrow \mathbf{Vf}(\boldsymbol{pp}, \boldsymbol{pk}_{sig}, \boldsymbol{pk}_{ev}, m, \sigma)$**:** *The verification algorithm* Vf *outputs a bit* $b \in \{0, 1\}$.

A DFS is *correct* if the verification algorithm outputs 1 for all honestly generated signatures and for all valid transformations of honestly generated signatures. We refer to the extended edition [6] for a formal definition of our correctness.

3 Security Notions for DFS

In this section we define unforgeability and privacy for delegatable functional signatures. In both cases we distinguish between outsider and insider attacks: In an *outsider* attack, the adversary only knows both public keys, whereas an adversary launching an *insider* attack knows the private key of the evaluator. Informally we say that a delegatable functional signature scheme provides privacy if it is computationally hard to distinguish whether a signature was created by the signer or whether it was modified by the evaluator. In the following subsections we discuss the intuition behind each definition in more detail and provide formal definitions.

For the following security definitions we follow the concept of Bellare and Rogaway in defining the security notions as a game $G(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ [12]. Each game G behaves as follows: First, it invokes an algorithm $\mathsf{Initialize}$ with the security parameter and sends its output to the algorithm $\mathcal{A}$. Then it simulates $\mathcal{A}$ with oracle access to all specified algorithms $\mathsf{Query[x]}$ that are defined for G. It also allows $\mathcal{A}$ to call the algorithm $\mathsf{Finalize}$ once and ends as soon as $\mathsf{Finalize}$ is called. The output of $\mathsf{Finalize}$ is a boolean value and is also the output of G. Note that G is allowed to maintain state. We say that $\mathcal{A}$ "wins" the game if $G(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda) = 1$.

3.1 Unforgeability

Intuitively, a delegatable functional signature scheme is unforgeable, if no adversary $\mathcal{A}$ is able to compute a fresh message-signature pair that is not trivially deducible from its knowledge. In the case of regular signature schemes this means that the attacker needs to compute a signature on a fresh message. The situation here is more complex, because our signatures are malleable and because several parties are involved (and they may even use malicious keys). We present three different unforgeability notions:

Unforgeability Against Outsider Attacks. We model the outsider as an active adversary that knows the public keys (pk_{sig}, pk_{ev}) and has oracle access to both the Sig and the $\mathsf{Eval}_{\mathcal{F}}$ algorithm. Our definition of unforgeability against outsider attacks resembles the traditional definition of unforgeability for signature schemes [32], where the adversary knows the public-key and has access to a signing oracle.

Unforgeability Against (Weak/Strong) Insider Attacks. Our second definition considers the case where the evaluator is malicious. We define two different notions depending on the capabilities of the adversary. That is, our first definition that we call unforgeability against weak insider attacks (or just insider attacks), gives the attacker access to an honestly generated private key sk_{ev}. The second notion allows the adversary to choose its own private key(s) maliciously. Note, that the attacker might choose these keys adaptively. We refer to this notion as unforgeability against *strong* insider attacks.

We model our notions by giving the adversary access to three different KGen oracles. An adversary that can only access $\mathsf{Query[KGenP]}$ to retrieve public keys is considered an *outsider*; an adversary that can access the oracle $\mathsf{Query[KGenS]}$ to retrieve one or more secret evaluator keys is considered an *insider*; an adversary that additionally can access the oracle $\mathsf{Query[RegKey]}$ to (adaptively) register its own (possibly malicious) evaluator keys is considered a *strong insider* (*S-Insider*). All adversaries have access to the honestly generated public signer key pk_{sig}. We keep track of the following sets: $\mathcal{K}_{\mathcal{C}}$ stores all key pairs, $K_{\mathcal{A}}$ stores all public keys for which the adversaries knows the private key, and Q stores $\mathcal{A}$'s queries to both $\mathsf{Query[Sign]}$ and $\mathsf{Query[Eval]}$. To handle the information that an adversary can trivially deduce from its queries, we define the transitive closure for functionalities.

Definition 2 *(Transitive closure of functionality $\mathcal{F}$). Given a functionality $\mathcal{F}$, we define the n-transitive closure $\mathcal{F}^n$ of $\mathcal{F}$ on parameters $(\lambda, (f, m))$ recursively as follows:*

- *For $n = 0$, $\mathcal{F}^0(\lambda, (f, m)) := \{(f, m)\}$.*
- *For $n > 0$, $\mathcal{F}^n(\lambda, (f, m)) := \{(f, m)\} \bigcup\limits_{\alpha, pk'_{ev}} \mathcal{F}^{n-1}(\lambda, \mathcal{F}(\lambda, f, \alpha, pk'_{ev}, m))$*

We define the transitive closure *$\mathcal{F}^*$ of $\mathcal{F}$ on parameters $(\lambda, (f, m))$ as* $\mathcal{F}^*(\lambda, (f, m)) := \bigcup\limits_{i=0}^{\infty} \mathcal{F}^i(\lambda, (f, m))$.

Note that the transitive closure $\mathcal{F}^*$ on $(\lambda, (f, m))$ might not be efficiently computable (and thus a challenger for Unf might not be efficient).

Although it is not necessary to compute the closure explicitly in our case, one could require a DFSS to provide an efficient algorithm $\mathsf{Check}{-}\mathcal{F}$ such that $\mathsf{Check}{-}\mathcal{F}(\lambda, f, m, m^*) = 1$ iff $m \in \mathcal{F}^*(\lambda, (f, m))$.

Definition 3 *(Unforgeability Against $X \in \{Outsider, Insider, S-Insider\}$ Attacks). Let* $\mathsf{DFSS} = (\mathsf{Setup}, \mathsf{KGen}_{sig}, \mathsf{KGen}_{ev}, \mathsf{Sig}, \mathsf{Eval}_{\mathcal{F}}, \mathsf{Vf})$ *be a delegatable*

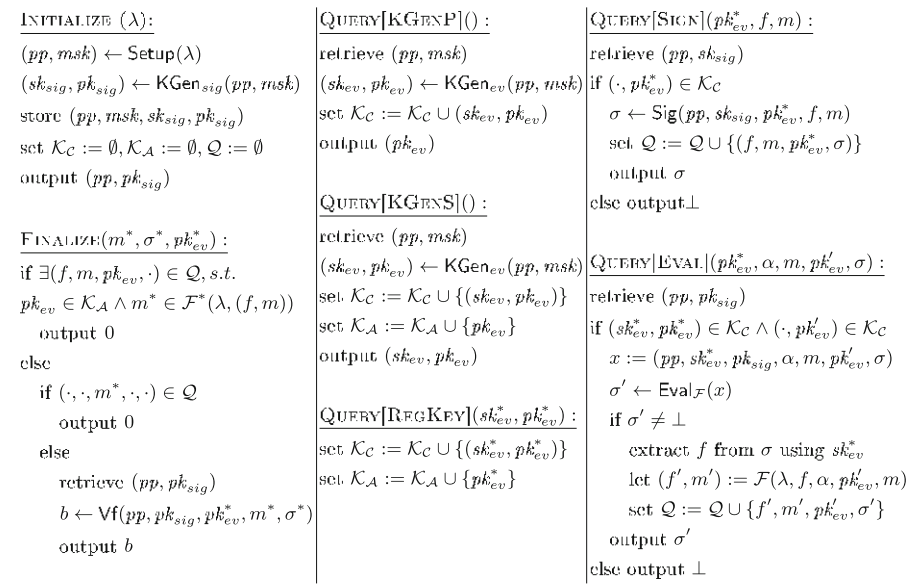


Fig. 2. Unforgeability for delegatable functional signature schemes.

functional signature scheme. The definition uses the game $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ *defined in Fig. 2. We say that* DFSS *is* existential unforgeable against X-attacks (EU-X-A) *for the functionality* $\mathcal{F}$ *if for all PPT adversaries* $\mathcal{A}_X$

$$\mathbf{Adv}^{\text{EU-X-A}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_X} = Pr\left[\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}_X, \lambda) = 1\right]$$

is negligible in λ*, where* $\mathcal{A}_{Outsider}$ *can neither invoke the oracles Query[KGenS] nor Query[RegKey]; the attacker* $\mathcal{A}_{Insider}$ *can not make use of Query[RegKey] and the adversary* $\mathcal{A}_{S\text{-}Insider}$ *is not restricted in its queries.*

Remark: We assume implicitly that f can be extracted from σ using sk_{ev} from any valid query to $\mathsf{Eval}_{\mathcal{F}}$. We believe that this is a reasonable assumption, because the evaluator that transforms a signature should learn the value f, as it describes the capabilities of the evaluator. In fact, our construction (Sect. 5) satisfies this property.

Remark on Measuring the Success of $\mathcal{A}$***:*** The success of the adversary is determined by the challenger and measured in the Finalize algorithm. Within the oracles Query[Sign] and Query[Eval], the challenger only allows to delegate to known keys $k \in \mathcal{K}_{\mathcal{C}}$. Note that his does not restrict the adversary, but allows the challenger to distinguish between weak insider and strong insider. All messages m signed either by Query[Sign] or Query[Eval] are added to $\mathcal{Q}$, together with the respective function f and the public key of the evaluator to whom the message was delegated.

For both outsiders ($\mathcal{K}_{\mathcal{A}} = \emptyset$) and insiders ($\mathcal{K}_{\mathcal{A}} \neq \emptyset$), we require that the forgery message m^* is a fresh message, i.e., it has not been signed by the

challenger, which is formally expressed by $(\cdot, m^*, \cdot, \cdot) \neq \mathcal{Q}$. Moreover, for insider adversaries, the forgery must not be trivially deducible from previously issued signatures $(f, m, {pk_{ev}}^*, \sigma)$ for keys ${pk_{ev}}^* \in \mathcal{K}_{\mathcal{A}}$. Observe that a different public key pk_{ev} might have been used when signing a message as compared to when verifying the resulting signature.

We leave it up to the signature scheme to decide whether a signature can verify under different evaluator keys. As a matter of fact: There can be schemes where Vf does not need to receive pk_{ev} at all.

3.2 Relations Between the Unforgeability Notions

The three notions of unforgeability describe a hierarchy of adversaries. It is intuitive, that security against outsider attacks does not imply security against insider attacks, as the key sk_{ev} of the evaluator can indeed leak enough information to construct the signature key sk_{sig} out of it.

However, although an *insider* adversary is stronger than an *outsider* adversary, making use of the additional oracle can weaken an adversary. Consider a scheme that leaks the secret signing key sk_{sig} with every signature, and that has only one valid public evaluator key pk_{ev}, that allows an insider with the respective secret key sk_{ev} to change messages inside signatures to arbitrary values. An insider that received sk_{ev} can not create a forgery, since every message he creates after receiving at least one signature is not considered a forgery: he could have computed them trivially using $\mathsf{Eval}_{\mathcal{F}}$. Without invoking $\mathsf{Query}[\mathsf{KGenS}]$, the adversary can request a signature and subsequently forge signatures for arbitrary messages, using the key sk_{sig} he received with the signature.

An *S-Insider* is again stronger than an *insider* or an *outsider*. A scheme can become insecure if a certain key pair (sk_{ev}, pk_{ev}) is used that is highly unlikely to be an output of KGen_{ev} (e.g., one of them is 0^{λ}).

Proposition 1 (EU-X-A-Implications). *Let* DFSS *be a functional signature scheme.*

(i) *For all PPT adversaries* $\mathcal{A}_{Outsider}$ *there exists a PPT adversary* $\mathcal{A}_{Insider}$ *s.t.* $\mathbf{Adv}^{\mathsf{EU\text{-}IA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Insider}} \geq \mathbf{Adv}^{\mathsf{EU\text{-}OA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Outsider}}$

(ii) *For all PPT adversaries* $\mathcal{A}_{Insider}$ *there exists a PPT adversary* $\mathcal{A}_{S\text{-}Insider}$ *s.t.* $\mathbf{Adv}^{\mathsf{EU\text{-}SIA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{S\text{-}Insider}} \geq \mathbf{Adv}^{\mathsf{EU\text{-}IA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Insider}}$

Proof. The proposition follows trivially. For (i), the adversary $\mathcal{A}_{\text{Insider}}$ runs a black-box simulation of $\mathcal{A}_{\text{Outsider}}$ and makes no use of the additional oracle. For (ii) the proposition follows analogously.

3.3 Privacy

Our privacy notion for DFS says that it should be hard to distinguish the following two signatures:

– a signature on a message m' that has been derived from a signature on a challenge message m by one or more applications of $\mathsf{Eval}_{\mathcal{F}}$.

– a fresh signature on m', where $(\cdot, m') \leftarrow \mathcal{F}(\ldots)$ was computed via one or more applications of $\mathcal{F}$ to m.

This indistinguishability should hold even against an adversary with oracle access to KGen_{ev}, Sig and $\mathsf{Eval}_{\mathcal{F}}$ that can choose which transformations are to be applied to which challenge message m and under which evaluator keys (even if they are known to the adversary), as long as the resulting signature is not delegated to the adversary.

Analogously to our definitions of unforgeability, we distinguish between three different types of adversaries, depending on their strength: outsiders, insiders and strong insiders. We model this by giving the adversary access to three different KGen oracles that are defined analogously to Definition 3 in Sect. 3.1. In the following definition, the set $\mathcal{K}_{\mathcal{C}}$ stores all key pairs, $K_{\mathcal{A}}$ contains all public keys for which the adversaries knows the private key, and $\mathcal{K}_X$ stores the keys used in the challenge oracle $\mathsf{Query}[\mathsf{Sign}\text{-}\mathcal{F}]$.

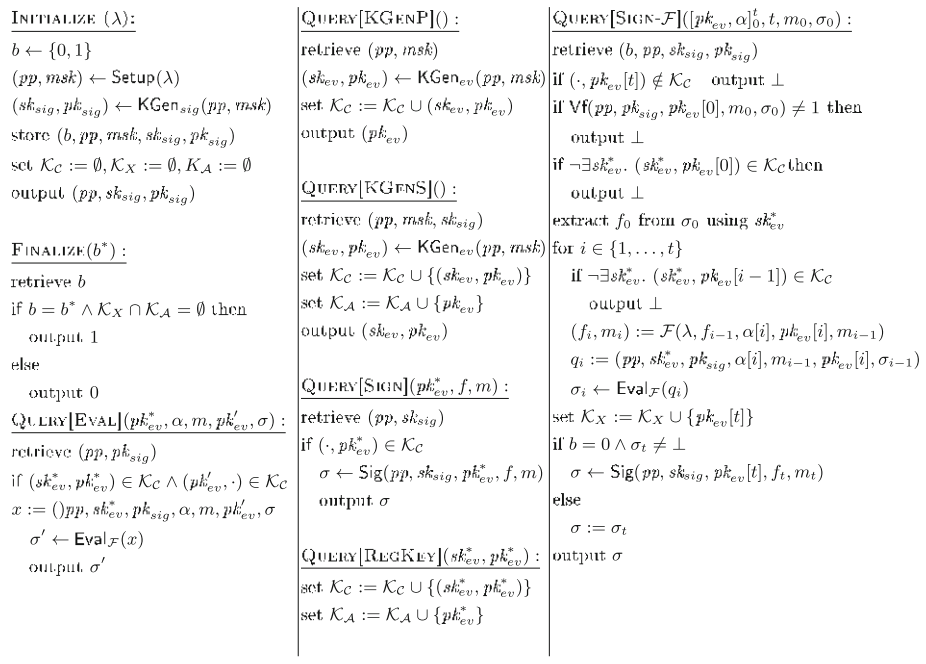


Fig. 3. Privacy under chosen functionality attacks CFA for delegatable functional signature schemes.

Definition 4. *(Privacy under chosen function attacks (CFA)) against* $X \in \{$*Outsider, Insider, S-Insider*$\}$. *Let* $\mathsf{DFSS} = (\mathsf{Setup}, \mathsf{KGen}_{sig}, \mathsf{KGen}_{ev}, \mathsf{Sig}, \mathsf{Eval}_{\mathcal{F}}, \mathsf{Vf})$ *be a delegatable functional signature scheme. The definition uses the game* $\mathsf{CFA}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ *defined in Fig. 3. We say that* DFSS *is* privacy-preserving

under chosen function attacks (X-CFA) *for the functionality* $\mathcal{F}$ *if for all PPT adversaries* $\mathcal{A}_X$

$$\mathbf{Adv}^{\text{PP-X-CFA}}_{\text{DFSS},\mathcal{F},\mathcal{A}_X} = \left| Pr\left[\text{CFA}(\text{DFSS}, \mathcal{F}, \mathcal{A}, \lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible in λ*, where* $\mathcal{A}_{Outsider}$ *can neither invoke the oracles* Query[KGenS] *nor* Query[RegKey]*; the attacker* $\mathcal{A}_{Insider}$ *can not make use of* Query[RegKey] *and the adversary* $\mathcal{A}_{S\text{-}Insider}$ *is not restricted in its queries.*

Remark on Measuring the Success of $\mathcal{A}$: The adversary may choose an arbitrary challenge message m_0, together with a signature σ_0 that allows the capability f_0 (technically the adversary can invoke PROC QUERY[SIGN] for computing σ_0), and a list of t public keys of evaluators together with evaluator inputs α. The chosen keys must be known to the challenger to distinguishing between outsiders, insiders and strong insiders. The challenger repeatedly applies $\text{Eval}_{\mathcal{F}}$ to σ_0, using the specified parameters α_i keys $pk_{ev}[i]$. Additionally, $\mathcal{C}$ computes the derived valued m_i and f_i for the resulting signature via direct application of $\mathcal{F}$. If all transformations succeed,[2] $\mathcal{C}$ yields a signature σ_t, on a message m_t delegated to $pk_{ev}[t]$ with capability f_t. Depending on the bit b, $\mathcal{C}$ either sends σ_t or a fresh signature on m_t delegated to $pk_{ev}[t]$ with capability f_t to $\mathcal{A}$ and adds the key $pk_{ev}[t]$ to the set $\mathcal{K}_X$ that contains all keys to which the signatures in challenges were (finally) delegated. If at the end of the game $\mathcal{K}_{\mathcal{A}} \cap \mathcal{K}_X \neq \emptyset$, the challenger outputs 0. This way we allow a scheme to leak some information to the evaluator to which a signature is delegated. For security against insider attacks only "local" information is allowed. After one delegation, this information has to vanish, since an insider adversary $\mathcal{A}$ can delegate the signature σ_t to a key $pk^*_{ev} \in \mathcal{K}_{\mathcal{A}}$ by using the Query[Eval] oracle.

3.4 Relations Between the Privacy Notions

For privacy, we have the same hierarchy as for unforgeability: A scheme that is secure against outsiders may be insecure against insiders, as the key sk_{ev} of an evaluator can help to distinguish between delegated and fresh signatures. Again, calling Query[KGenS] might weaken the adversary. Consider a scheme that does not preserve privacy against outsiders and that only has one valid evaluator key. An insider that calls both Query[KGenS] and Query[Sign-$\mathcal{F}$] is discarded, because it knows the only valid evaluator key (and thus $\mathcal{K}_X \cap \mathcal{K}_{\mathcal{A}} \neq \emptyset$).

Analogously to the hierarchy for unforgeability, an S-Insider is stronger an insider or an outsider. A scheme can leak information about delegation if a certain key pair (sk_{ev}, pk_{ev}) is used that is highly unlikely to be an output of KGen_{ev} (e.g., one of them is 0^λ).

[2] If one of the transformations failed Query[Sign-$\mathcal{F}$] outputs $\perp$ independently from the value of b, as we only want to give guarantees for valid signatures and not extend the notion of correctness.

Proposition 2 (PP-X-CFA-**Implications**). *Let* DFSS *be a functional signature scheme.*

(i) *For all PPT adversaries* $\mathcal{A}_{Outsider}$ *there exists a PPT adversary* $\mathcal{A}_{Insider}$ *s.t.* $\mathbf{Adv}^{\mathsf{PP\text{-}I\text{-}CFA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Insider}} \geq \mathbf{Adv}^{\mathsf{PP\text{-}O\text{-}CFA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Outsider}}$

(ii) *For all PPT adversaries* $\mathcal{A}_{Insider}$ *there exists a PPT adversary* $\mathcal{A}_{S\text{-}Insider}$ *s.t.* $\mathbf{Adv}^{\mathsf{PP\text{-}SI\text{-}CFA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{S\text{-}Insider}} \geq \mathbf{Adv}^{\mathsf{PP\text{-}I\text{-}CFA}}_{\mathsf{DFSS},\mathcal{F},\mathcal{A}_{Insider}}$

Proof. The proposition follows analogously to the proof for Proposition 1.

4 Possibility and Impossibility of DFS from OWFs

In this section we investigate the instantiability of DFS. In particular, we are interested in understanding which security property is "harder" to achieve. If we counter-intuitively are not interested in unforgeability, naturally we can construct a delegatable functional signature scheme DFSS unconditionally. A signature on m simply consists of the string "this is a signature for m". Obviously, this construction satisfies privacy against strong insiders in the sense of Definition 4 but not even unforgeability against outsiders.

Similarly, if we are not interested in privacy, DFS schemes can easily be constructed similarly to the construction in [15]. We assume a signature scheme S that is based on any one-way function. Now, the idea is that the signer simply signs a tuple consisting of the message together with the capability f and the public verification key of an evaluator. When evaluating, an evaluator adds his own signature on the previous signature together with α and the key of the following evaluator to the original signature. The verification procedure only accepts a signature if the signed trace of evaluations and delegations is legitimate w.r.t. the functionality $\mathcal{F}$. This scheme trivially satisfies unforgeability against strong insiders (cf. Definition 3) but none of our privacy notions. Thus, we obtain the following simple result:

Theorem 1. *If one-way functions exist, then there exists an unforgeable delegatable functional signature scheme.*

4.1 Impossibility of DFS from OWPs

In this section we prove an impossibility result showing that (D)FS cannot be constructed from OWP in a black-box way. The basic idea of our impossibility is to build a blind signature scheme in a black-box way. Since it is known that blind signature cannot be constructed from OWP only using black-box techniques [34], this implies that (D)FS cannot be constructed from OWF as well.

Blind Signatures and Their Security. A blind signature scheme is an interactive protocol between a signer $\mathcal{S}$, holding a secret key sk_{BS} and a user $\mathcal{U}$ who wishes to obtain a signature on a message m such that the user cannot create any additional signatures and such that $\mathcal{S}$ remains oblivious about this message. We refer to the extended edition [6] for formal definitions of commitment schemes and blind signatures.

Building Blind Signatures from (D)FS. The basic idea of our construction is as follows. The user chooses a message m, commits to the message and sends the commitment c on m to the signer. The signer signs the commitment, using a delegatable functional signature scheme and sends the signature σ_c back to the user. The user then calls $\mathsf{Eval}_{\mathcal{F}}$ with the open information o_m to derive a signature on m.

Given a commitment scheme $\mathsf{C} = (\mathsf{Commit}, \mathsf{Open})$ and a delegatable functional signature scheme $\mathsf{DFSS} = (\mathsf{Setup}, \mathsf{KGen}_{sig}, \mathsf{KGen}_{ev}, \mathsf{Sig}, \mathsf{Eval}_{\mathcal{F}}, \mathsf{Vf})$ for functionality $\mathcal{F}_C$, where $\mathcal{F}_C(\lambda, 1, \alpha, pk_{ev}, m) := (0, \mathsf{Open}(\alpha, m))$, we construct a blind signature scheme $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ as follows.[3]

$(\boldsymbol{sk}_{\mathrm{BS}}, \boldsymbol{pk}_{\mathrm{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^{\lambda})$. The key generation algorithm $\mathsf{KG}_{\mathsf{BS}}(1^{\lambda})$ performs the following steps:

$(msk, pp) \leftarrow \mathsf{Setup}(\lambda)$
$(sk_{sig}, pk_{sig}) \leftarrow \mathsf{KGen}_{sig}(pp, msk)$
$(sk_{ev}, pk_{ev}) \leftarrow \mathsf{KGen}_{ev}(pp, msk)$
$(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow (sk_{sig}, (pp, pk_{sig}, pk_{ev}, sk_{ev}))$

Signing. The protocol for $\mathcal{U}$ to obtain a signature on message m is depicted in Fig. 4 and consists of the following steps:

$\mathcal{U} \rightarrow \mathcal{S}$ The user sends a commitment c to the signer, where $(c, o_m) := \mathsf{Commit}(\lambda, m)$.

$\mathcal{S} \rightarrow \mathcal{U}$ The signer signs c together with the capability f and the public evaluator key of the user, obtaining $\sigma_c \leftarrow \mathsf{Sig}(sk_{sig}, pk_{ev}, 1, c)$. It sends σ_c to $\mathcal{U}$. The user calls $\mathsf{Eval}_{\mathcal{F}}$ with the open information o_m to derive a signature on m, as $\sigma_m \leftarrow \mathsf{Eval}_{\mathcal{F}}(sk_{ev}, pk_{sig}, o_m, pk'_{ev}, \sigma_c)$ and outputs (m, σ').

$b \leftarrow \mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, \sigma, m)$. The verification algorithm $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, \sigma, m)$ immediately returns $\mathsf{Vf}(pp, pk_{sig}, pk_{ev}, m, \sigma)$.

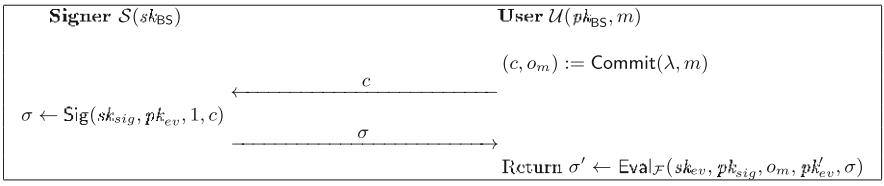


Fig. 4. Issue protocol of the two move blind signature scheme.

Theorem 2. *If* $\mathsf{DFSS} = (\mathit{Setup}, \mathit{KGen}_{sig}, \mathit{KGen}_{ev}, \mathit{Sig}, \mathit{Eval}_{\mathcal{F}}, \mathit{Vf})$ *is an unforgeable and private delegatable signature scheme (both against insider attacks) and* $\mathsf{C} = (\mathsf{Commit}, \mathsf{Open})$ *is a commitment scheme which is both computationally binding and hiding, then the interactive signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *as defined above is unforgeable and blind.*

[3] $\mathcal{F}_C$ outputs $\perp$ whenever $f \neq 1$.

Intuitively, unforgeability holds because the user can only obtain a signature on m if he calls $\mathsf{Eval}_{\mathcal{F}}$ on an authenticated commitment. This follows from the binding of the commitment scheme and from the unforgeability of the DFS. Blindness follows directly from the hiding property of the commitment scheme and from the privacy of our DFS. Note that the impossibility result of [34] rules out blind signature schemes that are secure against semi-honest adversaries.

We prove this theorem with the following two propositions.

Proposition 3. *If* $\mathsf{DFSS} = (\mathit{Setup}, \mathit{KGen}_{sig}, \mathit{KGen}_{ev}, \mathit{Sig}, \mathit{Eval}_{\mathcal{F}}, \mathit{Vf})$ *is an unforgeable delegatable signature scheme and* $\mathsf{C} = (\mathsf{Commit}, \mathsf{Open})$ *is a commitment scheme which is binding, then the interactive signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *as defined above is unforgeable.*

Proof. Assume there is an efficient algorithm $\mathcal{A}$ that forges a signature for BS. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ against the unforgeability of DFSS. First, $\mathcal{B}$ receives (pp, pk_{sig}) from the $\mathsf{Initialize}$ algorithm of the Unfl challenger. Then it calls $\mathsf{Query[KGenS]}()$ to receive an evaluator key pair (sk_{ev}, pk_{ev}), sets $pk_{\mathsf{BS}} := (pp, pk_{sig}, pk_{ev}, sk_{ev})$ and simulates $\mathcal{A}(pk_{\mathsf{BS}})$. Whenever $\mathcal{A}$ interacts with its signature oracle with a (blinded) message c_i, then $\mathcal{B}$ calls $\mathsf{Query[Sign]}(pk_{ev}, f, c_i)$ and returns the resulting signature σ_i to $\mathcal{A}$.

Eventually, $\mathcal{A}$ stops, outputting $k+1$ message-signature pairs (m_i^*, σ_i^*) after k successful interactive signing procedures, $\mathcal{B}$ chooses one of them at random and outputs it as a (possible) forgery.

For the analysis observe that the functionality $\mathcal{F}_C$ only applies the Open algorithm to the signed message and only if $f = 1$ has been set. So for every signature σ_i that $\mathcal{A}$ received, only one application of $\mathsf{Eval}_{\mathcal{F}}$ is allowed and only the Open algorithm can be applied. Since C is a binding commitment scheme, every commitment can only be opened to a unique message, except for a negligible error probability. However, this means that one of the signatures σ_i^* must be a forgery for DFSS as it is either a signature on a new message, or an evaluation of a function that has not been allowed (and thus is not in the transitive hull $\mathcal{F}^*$ of any message that has been signed via $\mathsf{Query[Sign]}$).

If $\mathcal{A}$ constructs a forgery, $\mathcal{B}$ chooses the right message-signature pair (m_i^*, σ_i^*) with probability at least $\frac{1}{k}$, so the probability that $\mathcal{B}$ constructs a forgery is at least $\frac{1}{k}$ times the probability that $\mathcal{A}$ constructs a forgery.

Remark. Note that f is necessary to ensure that $\mathsf{Eval}_{\mathcal{F}}$ is only called once, otherwise there is a simple attack against the scheme: The adversary $\mathcal{A}$ picks a message m and computes $(c_1, o_1) \leftarrow \mathsf{Commit}(m)$. Then $\mathcal{A}$ computes $(c_2, o_2) \leftarrow \mathsf{Commit}(c_1)$ and sends c_2 to the signer. Upon receiving a signature σ_{c_2}, the algorithm $\mathcal{A}$ uses $\mathsf{Eval}_{\mathcal{F}}$ to get a signature on $c_1 = \mathsf{Open}(c_2, o_2)$. Now $\mathcal{A}$ uses $\mathsf{Eval}_{\mathcal{F}}$ again to derive a signature on $m = \mathsf{Open}(c_1, o_1)$ and it outputs both signatures. Since $\mathcal{A}$ outputs two valid message-signature pairs (with two distinct messages) after one successful interaction it breaks the unforgeability of BS.

Proposition 4. *If* $\mathsf{DFSS} = (\mathit{Setup}, \mathit{KGen}_{sig}, \mathit{KGen}_{ev}, \mathit{Sig}, \mathit{Eval}_{\mathcal{F}}, \mathit{Vf})$ *is a private delegatable signature scheme and* $\mathsf{C} = (\mathsf{Commit}, \mathsf{Open})$ *is a commitment scheme*

which is hiding, then the interactive signature scheme $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *as defined above is blind.*

Proof. We show this proposition via a game-based proof. We start with the original game $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}(\lambda)$ for blindness and modify it until we reach a game in which the adversary can not observe any information that might help him in guessing the bit b.

In the following proof, as well as in subsequent proofs, we assign a number to each line where the first digit marks the game and the remaining digits the line in this game (e.g., 234 marks the 34th line of game 2). All lines that are not explicitly stated are as they were defined in the last game that defined them. We refer to Fig. 5 for a description of the games.

GAME $\mathcal{G}_0$
$-\,-\mathsf{KG}_{\mathsf{BS}}\,-\,-$
001 $(msk, pp) \leftarrow \mathsf{Setup}(\lambda)$
002 $(sk_{sig}, pk_{sig}) \leftarrow \mathsf{KGen}_{sig}(pp, msk)$
003 $(sk_{ev}, pk_{ev}) \leftarrow \mathsf{KGen}_{ev}(pp, msk)$
004 $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow (sk_{sig}, (pp, pk_{sig}, pk_{ev}, sk_{ev}))$
$-\,-\mathsf{find}-\,-$
005 $(m_0, m_1, state) \leftarrow \mathcal{A}(\mathsf{find}, sk_{\mathsf{BS}}, pk_{\mathsf{BS}})$
006 $b \leftarrow \{0,1\}$
$-\,-\mathsf{issue}-\,-$
007 $(state, \sigma_{c_b}, \sigma_{c_{1-b}}) \leftarrow \mathcal{A}(\mathsf{issue}, state)^{\langle \cdot, U_b \rangle^1, \langle \cdot, U_{1-b} \rangle^1}$
$-$ where U_b computes $-$
008 $(c_b, o_b) \leftarrow \mathsf{Commit}(m_b)$
$-$ and U_{1-b} computes $-$
009 $(c_{1-b}, o_{1-b}) \leftarrow \mathsf{Commit}(m_{1-b})$
$-$ unblind $-$
010 $\sigma_{m_0} \leftarrow \mathsf{Eval}_{\mathcal{F}}(pp, sk_{ev}, pk_{sig}, o_0, c_0, pk_{ev}, \sigma_{c_0})$
011 $\sigma_{m_1} \leftarrow \mathsf{Eval}_{\mathcal{F}}(pp, sk_{ev}, pk_{sig}, o_1, c_1, pk_{ev}, \sigma_{c_1})$
012 if $\sigma_{m_0} = \perp \vee \sigma_{m_1} = \perp$ then
013 $\quad (\sigma_{m_0}, \sigma_{m_1}) := (\perp, \perp)$
$-\,-\mathsf{guess}-\,-$
014 $b^* \leftarrow \mathcal{A}(\mathsf{quess}, state, \sigma_{m_0}, \sigma_{m_1})$

GAME $\mathcal{G}_1$
110 $\sigma_{m_0} \leftarrow \mathsf{Sig}(pp, sk_{sig}, pk_{ev}, 0, m_0, pk_{ev})$
111 $\sigma_{m_1} \leftarrow \mathsf{Sig}(pp, sk_{sig}, pk_{ev}, 0, m_1, pk_{ev})$

GAME $\mathcal{G}_2$
208 $(c_x, o_x) \leftarrow \mathsf{Commit}(0^n)$
209 $(c_y, o_y) \leftarrow \mathsf{Commit}(0^n)$

Fig. 5. The games for our proof for Proposition 4.

Game $\mathcal{G}_0 \Rightarrow$ Game $\mathcal{G}_1$: Since DFSS is private, we can create new signatures instead of calling $\mathsf{Eval}_{\mathcal{F}}$ on the signature of $\mathcal{A}$.

Claim. GAME $\mathcal{G}_0$ and GAME $\mathcal{G}_1$ are computationally indistinguishable.

Proof. Assume there is an efficient, malicious signer $\mathcal{A}$, which is able to distinguish both games. We show how to use $\mathcal{A}$ to build a distinguisher $\mathcal{B}$ that breaks the privacy property of DFSS. The algorithm $\mathcal{B}$ simulates GAME $\mathcal{G}_0$, but instead of calling $\mathsf{Eval}_{\mathcal{F}}$ in lines 10 and 11, it respectively queries $\mathsf{Query}[\mathsf{Sign} - \mathcal{F}]((pk_{ev}, \perp), (pk_{ev}, o_0), 1, c_0, \sigma_{c_0})$ and $\mathsf{Query}[\mathsf{Sign} - \mathcal{F}]((pk_{ev}, \perp), (pk_{ev}, o_1), 1, c_1, \sigma_{c_1})$. If $\mathcal{A}$ distinguishes GAME $\mathcal{G}_0$ and GAME $\mathcal{G}_1$ with probability noticeably larger than $\frac{1}{2}$, then $\mathcal{B}$ breaks the privacy property of DFSS by guessing the bit b of the challenger for CFA with probability noticeably larger than $\frac{1}{2}$.

Game $\mathcal{G}_1 \Rightarrow$ Game $\mathcal{G}_2$: In Game $\mathcal{G}_1$ the signature of the adversary is not used anymore and thus the commitment is not opened anymore. Consequently we can replace the commitments by commitments to zero.

Claim. Game $\mathcal{G}_1$ and Game $\mathcal{G}_2$ are computationally indistinguishable.

Proof. This follows from the hiding property of the commitment scheme. If there is an efficient, malicious signer $\mathcal{A}$, which is able to distinguish the two games, then we can use it to break the hiding property of C.

In Game $\mathcal{G}_2$ the bit b is never used. The commitments and all signatures are completely independent of b. Thus, the probability that $\mathcal{A}$ guesses $b^* = b$ in Game $\mathcal{G}_2$ is exactly $\frac{1}{2}$. Since the games are (pairwise) computationally indistinguishable, the proposition holds.

Combining Theorem 2 and the impossibility result of [34] (which is based on the work of Barak and Mahmoody [7]), we obtain the following result.

Corollary 1. *(Delegatable) functional signature schemes that are unforgeable and private against insider adversaries cannot be build from one-way permutations in a black-box way.*

Remark. Since this construction did not use the delegation property of the delegatable functional signature scheme, it should be possible to construct blind signatures from functional signatures, as defined by [15]. A DFS that is unforgeable and private against outsider adversaries is not ruled out by this corollary. Exploring whether the impossibility also holds in this case would be interesting.

5 Bounded DFS from Trapdoor Permutations

In this section we construct a bounded delegatable functional signature scheme DFSS as defined in Sect. 2.1, where we put an *a-priori* bound on the number of evaluations. Our construction is based on (regular) unforgeable signature schemes, a public-key encryption scheme, and a non-interactive zero-knowledge proof system. It is well known that these primitives can be constructed from (doubly enhanced) trapdoor permutations. Formal definition of the underlying primitives can be found in the extended edition of this paper [6].

5.1 Our Scheme

Our construction follows the encrypt and proof strategy and is completely general with respect to efficiently computable functionalities $\mathcal{F}$ with the exception that $\mathcal{F}$ may allow only for up to n applications of $\mathsf{Eval}_{\mathcal{F}}$. We let the signer choose how many applications he allows by defining f as a tuple $(f', k) \in \mathcal{P}_f \times \{0, \ldots, n\}$. We achieve the strong notion of privacy under chosen function attack (CFA) according to Definition 4 by applying the following idea : If the signer choses a

number of k possible applications of $\mathsf{Eval}_{\mathcal{F}}$, we still create $n+1$ encryptions, but place the encryption a signature on m at the $k+1^{th}$ position (and only encryptions of zero-strings at the other positions). The evaluators fill up the encryptions from the k^{th} position to the first one. Although each evaluator receives information from his predecessor in the chain of delegations (the first evaluator will know, that the signature originates from the signer), even the second evaluator in the chain will be unable to find out more than its predecessor and the number of applications of $\mathsf{Eval}_{\mathcal{F}}$ that are still allowed. Figure 6 shows the construction in more detail.

$\underline{\mathsf{Setup}(1^\lambda):}$
$CRS \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda)$
$(msk_{\mathcal{S}}, pp_{\mathcal{S}}) \leftarrow \mathsf{Setup}_{\mathcal{S}}(1^\lambda)$
$(msk_{\mathcal{E}}, pp_{\mathcal{E}}) \leftarrow \mathsf{Setup}_{\mathcal{E}}(1^\lambda)$
$(\widetilde{dk}, \widetilde{ek}) \leftarrow \mathsf{KGen}_{\mathcal{E}}(pp_{\mathcal{E}}, msk_{\mathcal{E}})$
$pp := (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
$msk := (msk_{\mathcal{S}}, msk_{\mathcal{E}})$
output (pp, msk)

$\underline{\mathsf{KGen}_{sig}(pp, msk):}$
parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
parse $msk = (msk_{\mathcal{S}}, msk_{\mathcal{E}})$
$(ssk_{\mathcal{S}}, vk_{\mathcal{S}}) \leftarrow \mathsf{KGen}_{\mathcal{S}}(pp_{\mathcal{S}}, msk_{\mathcal{S}})$
$pk_{sig} := vk_{\mathcal{S}}$
$sk_{sig} := (ssk_{\mathcal{S}}, pk_{sig})$
output (sk_{sig}, pk_{sig})

$\underline{\mathsf{KGen}_{ev}(pp, msk):}$
parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
parse $msk = (msk_{\mathcal{S}}, msk_{\mathcal{E}})$
$(ssk_{\mathcal{F}}, vk_{\mathcal{F}}) \leftarrow \mathsf{KGen}_{\mathcal{S}}(pp_{\mathcal{S}}, msk_{\mathcal{S}})$
$(dk, ek) \leftarrow \mathsf{KGen}_{\mathcal{E}}(pp_{\mathcal{E}}, msk_{\mathcal{E}})$
$sk_{ev} := (ssk_{\mathcal{F}}, dk)$
$pk_{ev} := (vk_{\mathcal{F}}, ek)$
output (sk_{ev}, pk_{ev})

$\underline{\mathsf{Sig}(pp, sk_{sig}, pk_{ev}, (f,k), m):}$
parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
parse $pk_{ev} = (vk_{\mathcal{F}}, ek)$
parse $sk_{sig} = (ssk_{\mathcal{S}}, pk_{sig}))$
$h_k := (f, m, pk_{ev}, k)$
$\sigma_k \leftarrow \mathsf{Sig}_{\mathcal{S}}(pp_{\mathcal{S}}, ssk_{\mathcal{S}}, h_k; r_{\mathcal{S}})$
$s_k \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\sigma_k, h_k); r_{s_k})$
For $i \in \{0, \ldots, n\} \setminus \{k\}$
  $\sigma_i := 0^{|\sigma_k|}$
  $h_i := (0^{\ell_f(\lambda)}, 0^{\ell_m(\lambda)}, 0^{|pk_{ev}|}, 0)$
  $s_i \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\sigma_i, h_i); r_{s_i})$
$d \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (f, k, \sigma_k); r_d)$
$S := (s_0, \ldots, s_n)$
$x = (pp, pk_{sig}, pk_{ev}, S, d, m)$
$\omega = (f, n, r_d)$
with $\omega_n = (r_{\mathcal{S}}, k)$ and
$x_n = (pp - \mathcal{S}, pp_{\mathcal{E}}, pk_{sig}, pk_{ev}, S, m, CRS, f, \sigma)$
$\Pi \leftarrow P_{\mathsf{NIZK}}(CRS, x, \omega)$
$\sigma := (S, d, \Pi)$
output σ

$\underline{\mathsf{Vf}(pp, pk_{sig}, pk_{ev}, m, \sigma):}$
parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
parse $pk_{sig} = (vk_{\mathcal{S}}, \widetilde{ek})$
parse $pk_{ev} = (vk_{\mathcal{F}}, ek)$
parse $\sigma = (S, d, \Pi)$
$x := (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk_{ev}, S, d, m, CRS)$
$b \leftarrow \mathsf{Vf}_{\mathsf{NIZK}}(CRS, x, \Pi)$
output b

$\underline{\mathsf{Eval}_{\mathcal{F}}(pp, sk_{ev}, pk_{sig}, \alpha, m, pk'_{ev}, \sigma):}$
parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$
parse $sk_{ev} = (ssk_{\mathcal{F}}, dk)$
parse $pk'_{ev} = (vk'_{\mathcal{F}}, ek')$
parse $\sigma = (S, d, \Pi)$
$(f, i, \sigma_i) \leftarrow \mathsf{Dec}_{\mathcal{E}}(pp_{\mathcal{E}}, dk, d)$
$x = (pp, pk_{sig}, pk_{ev}, S, d, m)$
if $pk_{ev} = (vk_{\mathcal{F}}, ek)$ belongs to sk_{ev}
$\wedge\, \mathsf{Vf}_{\mathsf{NIZK}; Z_i}(CRS, x, \Pi) = 1$
  $(\hat{f}, \hat{m}) := \mathcal{F}(\lambda, f, \alpha, pk'_{ev}, m)$
  $h_{i-1} := (\hat{f}, \hat{m}, pk'_{ev}, i-1)$
  $\hat{\sigma}_{i-1} \leftarrow \mathsf{Sig}_{\mathcal{S}}(pp_{\mathcal{S}}, ssk_{\mathcal{F}}, h_{i-1}; r_{\mathcal{S}})$
  $s_{i-1} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\hat{\sigma}_{i-1}, h_{i-1}); r_s)$
  $d \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek', (\hat{f}, i-1, \hat{\sigma}_{i-1}); r_d)$
  $\hat{x} = (pp, pk_{sig}, pk'_{ev}, S, d, \hat{m})$
  $\omega = (\hat{f}, i-1, r_d)$
  with $\omega_{i-1} = (r_{\mathcal{S}}, \Pi, pk_{ev}, m, f, \alpha)$,
  $x_{i-1} = (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk'_{ev}, S, \hat{m}, CRS, \hat{f}, \hat{\sigma}_{i-1})$
  $\hat{\Pi} \leftarrow P_{\mathsf{NIZK}}(CRS, x, \omega)$
  $\hat{\sigma} := (S, d, \Pi)$
  output $\hat{\sigma}$
else
  output $\perp$

Fig. 6. Construction of a DFSS.

Given a signature scheme $S = (\mathsf{Setup}_{\mathcal{S}}, \mathsf{KGen}_{\mathcal{S}}, \mathsf{Sig}_{\mathcal{S}}, \mathsf{Vf}_{\mathcal{S}})$ with a simple key generation algorithm and with signatures of equal length, an encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathcal{E}}, \mathsf{KGen}_{\mathcal{E}}, \mathsf{Enc}_{\mathcal{E}}, \mathsf{Dec}_{\mathcal{E}})$ and a zero-knowledge scheme

$\mathsf{NIZK} = (\mathsf{KGen}_{\mathsf{NIZK}}, P_{\mathsf{NIZK}}, \mathsf{Vf}_{\mathsf{NIZK}})$ for languages in NP we construct a delegatable functional signature scheme DFSS as follows: We define a recursive class of languages L_i, where $L_n : \ x_n = (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk_{ev}, S, m, CRS, f, \sigma) \in L_n$ means that there exists a witness $\omega_n = (r, k)$ such that $pk_{sig} = (vk_{\mathcal{S}}, \widetilde{ek}) \ \wedge \ s_k = \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\sigma, (f, m, pk_{ev}, k)); r) \ \wedge \ \mathsf{Vf}_{\mathcal{S}}(pp_{\mathcal{S}}, vk_{\mathcal{S}}, (f, m, pk_{ev}, k), \sigma) = 1$ and where L_i for $0 \leq \mathrm{i} < \mathrm{n} : \ \mathrm{x_i} = (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{\mathrm{sig}}, pk_{\mathrm{ev}}, \mathrm{S}, \mathrm{m}, CRS, \mathrm{f}, \sigma) \in \mathrm{L_i}$ if there exists a witness $\omega_i = (r, \Pi, pk'_{ev}, m', f', \alpha)$ s.t. $pk_{sig} = (vk_{\mathcal{S}}, \widetilde{ek})$ and

$$
\begin{aligned}
&\wedge \quad s_i = \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\sigma, (f, m, pk_{ev}, k)); r) \ \wedge \ \mathsf{Vf}_{\mathcal{S}}(pp_{\mathcal{S}}, vk', (f, m, pk_{ev}, k), \sigma) = 1 \\
&\wedge \quad x' := (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk'_{ev}, S', m', CRS, f') \ \wedge \ S = S' \, \{s'_i := s_i\} \\
&\wedge \quad pk'_{ev} = (vk', \cdot) \wedge (f, m) = \mathcal{F}(\lambda, f', \alpha, pk'_{ev}, m') \\
&\wedge \quad \left(\mathsf{Vf}_{\mathsf{NIZK}_{i+1}}(CRS, x') = 1 \vee \mathsf{Vf}_{\mathsf{NIZK}_n}(CRS, x') = 1\right).
\end{aligned}
$$

The signer proves that $x = (pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}), pk_{sig}, \ pk_{ev} = (vk_{\mathcal{F}}, ek), S, d, m) \in L$, where L contains tuples for which there exists a witness $\omega = (f, i, r_d)$ such that $\mathsf{Vf}_{\mathsf{NIZK}_i}(CRS, (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk_{ev}, S, m, CRS, f, \sigma)) = 1 \wedge d \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (f, i, \sigma); r_d)$.

5.2 Security

Concerning security, we show the following theorem.

Theorem 3. *If $\mathcal{E}$ is a public key encryption scheme that is secure against chosen ciphertext attacks (CCA-2), $\mathcal{S}$ a length preserving unforgeable signature scheme with a simple key generation, and NIZK is a sound non-interactive proof scheme that is zero knowledge, the construction presented in this section is* unforgeable against outsider and (strong) insider attacks *and* secure against chosen function attacks (CFA) against outsiders and (strong) insiders

Proof. The theorem follows directly from Lemmas 1 and 2.

Lemma 1. *If $\mathcal{E}$ is a public key encryption scheme, $\mathcal{S}$ a length preserving unforgeable signature scheme with a simple key generation (i.e., not requiring a master secret key), and NIZK is a sound non-interactive proof scheme, then the construction* DFSS *presented in Sect. 5 is* unforgeable against outsider and (strong) insider attacks *according to Definition 3.*

Given an adversary $\mathcal{A}$ that breaks the unforgeability of our construction we construct an efficient adversary $\mathcal{B}$ that breaks the underlying signature scheme.

Proof. By Proposition 1 it suffices to show unforgeability against an S-Insider adversary. Assume towards contradiction that DFSS is not unforgeable against strong insider attacks. Then there exists an efficient adversary $\mathcal{A} := \mathcal{A}_{\text{S-Insider}}$ that makes at most $p(\lambda)$ many steps for a polynomial p and that wins the game $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}_{\text{S-Insider}}, \lambda)$, formalized in Definition 3, with non-negligible probability. Since $\mathcal{A}$ makes at most $p(\lambda)$ many steps, $\mathcal{A}$ invokes the oracle $\mathsf{Query}[\mathsf{KgenP}]$ at most $p(\lambda)$ many times. We show how to build an adversary $\mathcal{B}$ that runs $\mathcal{A}$ in a

black-box way in order to break the unforgeability of $\mathcal{S}$ with non-negligible probability. In the following we denote the values and the oracles that the challenger $\mathcal{C}$ from the game $\mathsf{Unf}(S, \mathcal{B}, \lambda)$ provides to $\mathcal{B}$ with the index $\mathcal{C}$.

The algorithm $\mathcal{B}$, upon receiving as input a tuple $(pp_{\mathcal{C}}, vk_{\mathcal{C}})$ from $\mathsf{Initialize}_{\mathcal{C}}$, simulates a challenger for the game $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$. First, the algorithm $\mathcal{B}$ generates the public parameters and the master public/private key-pair, computing $(pp_{\mathcal{E}}, msk_{\mathcal{E}}) \leftarrow \mathsf{Setup}_{\mathcal{E}}(1^{\lambda})$, $CRS \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^{\lambda})$and setting $pp := (CRS, pp_{\mathcal{C}}, pp_{\mathcal{E}})$, $msk := (\epsilon, msk_{\mathcal{E}})$. Subsequently, $\mathcal{B}$ computes $(\widetilde{dk}, \widetilde{ek}) \leftarrow \mathsf{KGen}_{\mathcal{E}}(pp_{\mathcal{E}}, msk_{\mathcal{E}})$, $(sk_{\mathcal{S}}, vk_{\mathcal{S}}) \leftarrow \mathsf{KGen}_{\mathcal{S}}(pp_{\mathcal{C}}, \varepsilon)$ and sets $pk_{sig} := vk_{\mathcal{S}}$.

The algorithm $\mathcal{B}$ embeds its own challenge key $vk_{\mathcal{C}}$ in a randomly chosen position $z \in \{0, \ldots, p(\lambda)\}$; if $z = 0$, then $\mathcal{B}$ replaces $vk_{\mathcal{S}}$ by $vk_{\mathcal{C}}$. Finally, $\mathcal{B}$ runs a black-box simulation of $\mathcal{A}$ on input (pp, pk_{sig}), where $pk_{sig} = vk_{\mathcal{S}}$ or $pk_{sig} = vk_{\mathcal{C}}$, depending on z and $\mathcal{B}$ simulates the four oracles $\mathsf{Query[Sign]}, \mathsf{Query[Trans]}, \mathsf{Query[KGenP]}$ and $\mathsf{Query[Finalize]}$. The inputs of these oracles are provided by $\mathcal{A}$ upon calling them and are thus sent to $\mathcal{B}$. The algorithm $\mathcal{B}$ handles the oracle queries from $\mathcal{A}$ as follows:

Query[KGenP]()**:** The algorithm $\mathcal{B}$ answers the i^{th} invocation of $\mathsf{Query[KgenP]}$ as follows. First, $\mathcal{B}$ generates a key pair for encryption and decryption $(dk, ek) \leftarrow \mathsf{KGen}_{\mathcal{E}}(pp_{\mathcal{E}}, msk_{\mathcal{E}})$. Then it behaves differently depending on i: If $i = z$, then $\mathcal{B}$ sends $vk_{\mathcal{C}}$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates a new key-pair $(sk_{ev}, pk_{ev}) \leftarrow \mathsf{KGen}_{\mathcal{S}}(pp_{\mathcal{C}}, \varepsilon)$, stores this pair, and sends pk_{ev} to $\mathcal{A}$.

Query[Sign]$(\boldsymbol{pk}^*_{ev}, f, g, m)$**:** If $z \neq 0$, the algorithm $\mathcal{B}$ computes all necessary values locally exactly as a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ would. For computing the values locally, $\mathcal{B}$ needs to know pp (publicly known), $sk_{sig} = (ssk_{\mathcal{S}}, vk_{\mathcal{S}})$ (generated by $\mathcal{B}$ since $z \neq 0$) and the values pk^*_{ev}, f and m (provided to $\mathcal{B}$ by $\mathcal{A}$).
If $z = 0$, this local computation is not possible since $\mathcal{B}$ replaced $vk_{\mathcal{S}}$ with $vk_{\mathcal{C}}$. Thus, the algorithm $\mathcal{B}$ sets $h_k := (f, m, pk_{ev}, k)$ and invokes $\mathsf{Query[Sig]}_{\mathcal{C}}(h_k)$. It sets σ_k to the output of the challenger and otherwise proceeds as above.

Query[Eval]$(\boldsymbol{pk}^*_{ev}, \alpha, m, \boldsymbol{pk}'_{ev}, \sigma)$**:** Parse $pk^*_{ev} = (vk, ek)$. $\mathcal{B}$ behaves differently depending on the value of vk.
If the key pk^*_{ev} is a key for which $\mathcal{B}$ knows a secret key (in particular it does not contain the challenge key $vk_{\mathcal{C}}$), $\mathcal{B}$ computes all necessary values locally exactly as a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ would. For computing the values locally, $\mathcal{B}$ needs to know pp (publicly known), a value for sk^*_{ev} corresponding to pk^*_{ev} (discussed below), pk_{sig} (known to $\mathcal{B}$) and the values for α, m, pk'_{ev} and σ (provided by $\mathcal{A}$). There are four cases for sk^*_{ev}. If pk^*_{ev} was output by $\mathsf{Query[KGenP]}$ (and since $vk \neq vk_{\mathcal{C}}$, this was not the z^{th} invocation of $\mathsf{Query[KGenP]}$), $\mathcal{B}$ has generated the value $sk^*_{ev} = (ssk_{\mathcal{F}}, dk)$ itself. The same applies if pk^*_{ev} was output by $\mathsf{Query[KGenS]}$. If pk^*_{ev} was registered by $\mathcal{A}$ via $\mathsf{Query[RegKey]}$, $\mathcal{B}$ uses the corresponding (registered) key sk^*_{ev}. If none of the three cases applies, then the key pk^*_{ev} is unknown and $\mathcal{B}$ returns $\perp$ instead.

If the key pk^*_{ev} is the key in which $\mathcal{B}$ has embedded its own challenge key ($vk = vk_{\mathcal{C}}$), a corresponding value $ssk_{\mathcal{F}}$ (the first part of the secret key sk^*_{ev} corresponding to pk^*_{ev}) is not known to $\mathcal{B}$. This key is necessary to sign the value $h = (\hat{f}, \hat{m}, pk'_{ev}, k-1)$. Thus, instead of computing a signature with some key $ssk_{\mathcal{F}}$, $\mathcal{B}$ calls its own oracle $\mathsf{Query[Sig]}_{\mathcal{C}}(h)$ and otherwise proceeds as above.

Finalize $(m^*, \sigma^*, \boldsymbol{pk}^*_{ev})$**:** Eventually, $\mathcal{A}$ invokes Finalize on a tuple $(m^*, \sigma^*, pk^*_{ev})$, then $\mathcal{B}$ parses $\sigma^* = (S, d, \pi)$ with $S = (s_0, \ldots, s_{n+1})$. Now, the algorithm $\mathcal{B}$ checks the validity of the signature computing $\mathsf{Vf}(pp, pk_{sig}, pk^*_{ev}, m^*, \sigma^*)$. If the verification algorithm outputs 0, then $\mathcal{B}$ stops. Otherwise $\mathcal{B}$ decrypts all signatures $(\sigma_i, h_i) := \mathsf{Dec}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{dk}, s_i)$. $\mathcal{B}$ tries to find a pair (σ_x, h_x) that verifies under the key $vk_{\mathcal{C}}$ and that has not been sent to $\mathsf{Query[Sign]}_{\mathcal{C}}$ by $\mathcal{B}$, then $\mathcal{B}$ sends (h_x, σ_x) to its own $\mathsf{Finalize}_{\mathcal{C}}$ oracle. Otherwise it halts.

Claim. The algorithm $\mathcal{B}$ perfectly simulates a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$.

Proof (for Claim 5.2). We investigate the simulation of all oracles and local computations.

Simulation of Initialize: Observe that by construction and by the fact that $\mathcal{S}$ the values pp and msk are identically distributed to values for pp and msk generated by a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$. Thus, the keys generated out of them are also identically distributed. If $z \neq 0$ then $\mathcal{B}$ uses only pp and msk to compute the keys (sk_{sig}, pk_{sig}) and thus they are identically distributed as keys (sk_{sig}, pk_{sig}) generated by $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$.
If $z = 0$, then $\mathcal{B}$ replaces the verification $vk_{\mathcal{S}}$ of the signer with the verification key $vk_{\mathcal{C}}$ of the challenger. However, since $\mathcal{S}$ does not require a master secret key, the key $vk_{\mathcal{C}}$ is identically distributed as the key $vk_{\mathcal{S}}$. Moreover, $\mathcal{B}$ does not use the corresponding signing key $ssk_{\mathcal{S}}$ in any way and queries its own signing oracle instead.

Simulation of Query[KGenP]: On any but the z^{th} invocation, $\mathcal{B}$ perfectly simulates a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ and computes a new key pair based on pp and msk. As pp and msk are identically distributed as for a challenger, the resulting keys are also identically distributed.
On the z^{th} invocation, however, $\mathcal{B}$ replaces the verification key $vk_{\mathcal{F}}$ with the verification key $vk_{\mathcal{C}}$ of the challenger. However, since $\mathcal{S}$ does not require a master secret key, the key $vk_{\mathcal{C}}$ is identically distributed as the key $vk_{\mathcal{S}}$. Moreover, $\mathcal{B}$ does not use the corresponding signing key $ssk_{\mathcal{F}}$ in any way and queries its own signing oracle instead.

Simulation of Query[KGenS]: $\mathcal{B}$ uses the values pp and msk that are identically distributed to the corresponding values of a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$. On them it performs a perfect simulation of Query[KGenS]. Thus, the resulting keys have the same distribution as the keys output by Query[KGenS] of the challenger.

Simulation of Query[RegKey]: This oracle does not return an answer.

Simulation of Query[Sign] and Query[Eval: $\mathcal{B}$ perfectly simulates these oracles as long as it does not have to create a signature with the key corresponding to $vk_{\mathcal{C}}$. However, in these cases $\mathcal{B}$ calls its own signature oracle. Since the keys are identically distributed, this still is a perfect simulation.

Since all messages that $\mathcal{B}$ sends to $\mathcal{A}$ are identically distributed to the messages that $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ sends to $\mathcal{A}$, the algorithm $\mathcal{B}$ perfectly simulates a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$.

Claim. Whenever $\mathcal{A}$ produces a forgery, then with probability at least $\frac{1}{p(\lambda)+1}$ $\mathcal{B}$ also produces a forgery.

Proof (Proof of Claim 2.5). First we show the following statement: Whenever $\mathcal{A}$ produces a forgery $(m^*, \sigma^*, pk^*_{ev})$, then σ^* is of the form $\sigma^* = (S, d, \pi)$. Moreover, $S = (s_0, \ldots, s_{n+1})$ contains the encryption s_x of a signature σ_x such that:

- σ_x verifies for a message m_x under a key vk^*
- vk^* either equals pk_{sig} or that has been sent to $\mathcal{A}$ as an answer to an oracle query Query[KGenP]
- m_x a message that has not been sent to Query[Sign] or achieved as result of Query[Eval].

Assume that $\mathcal{A}$ invokes Finalize with $(m^*, \sigma^*, pk^*_{ev})$ such that $(m^*, \sigma^*, pk^*_{ev})$ constitutes a forgery for DFSS. Technically: If our algorithm $\mathcal{B}$ would simulate the Finalize algorithm (as in Fig. 7), it would output 1.[4]

PROC FINALIZE$(m^*, \sigma^*, pk^*_{ev})$:
if $\exists (f, g, m, pk_{ev}, \cdot) \in \mathcal{Q}, s.t. pk_{ev} \in \mathcal{K}_{\mathcal{A}} \wedge m^* \in \mathcal{F}^*(\lambda, (f, m))$
or $(\cdot, \cdot, m^*, \cdot, \cdot) \in \mathcal{Q}$
  output 0
else
  retrieve (pp, pk_{sig})
  parse $pp = (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek}); \sigma^* = (S, d, \Pi)$
  parse $pk_{sig} = (vk_S, \widetilde{ek}); pk^*_{ev} = (vk_{\mathcal{F}}, ek)$
  $x := (pp_{\mathcal{S}}, pp_{\mathcal{E}}, pk_{sig}, pk^*_{ev}, S, d, m^*, CRS)$
  $b \leftarrow \mathsf{Vf}_{\mathsf{NIZK}}(CRS, x, \Pi)$
  output b

Fig. 7. A simulated version of Finalize for our construction DFSS.

If Finalize would output 1, $(\cdot, m^*, \cdot, \cdot) \notin \mathcal{Q}$. This especially means that σ^* can not be output of Query[Sign] or Query[Eval]. Moreover, there was no query to

[4] Note that simulating Finalize is not necessarily possible in polynomial time, which is of no concern, since $\mathcal{B}$ does not simulate Finalize.

$\mathsf{Query[Sign]}(pk'_{ev}, f, m)$ for an adversary key pk'_{ev} such that m^* is in the transitive hull $\mathcal{F}^*(\lambda, (f, m))$. Also, there was no query to $\mathsf{Query[Eval]}(pk_{ev}, \alpha, m, pk'_{ev}, \sigma')$ for an adversary key pk'_{ev} such that f was extracted from σ' and such that m^* is in the transitive hull $\mathcal{F}^*(\lambda, (f', m'))$ for $(f', m') := \mathcal{F}(\lambda, f, \alpha, pk'_{ev}, m)$.

If the NIZK Π verifies then there is a signature that verifies under pk_{sig} and that marks the start of the delegation chain. Let σ_k be this signature for a value $h_k = (f, m, pk_{ev}, k)$. The NIZK makes sure that m^* is in the transitive hull $\mathcal{F}^*(\lambda, (f, m))$ and that all transformations are legitimized by the previous ones (depending on the intermediate α's).

We distinguish the following cases:

- $i = 0$: There was no call to $\mathsf{Query[Sig]}$ with parameters $(pk_{ev}, (f, k), m)$. Thus, $\mathcal{B}$ never sent h_k to $\mathsf{Query[Sig]}_{\mathcal{C}}$. and thus, S contains a signature $\sigma_x = \sigma_k$ that verifies with pk_{sig} for the message h_k.
- $0 < i < k$: There was a call to $\mathsf{Query[Sig]}$ with parameters $(pk_{ev}, (f, k), m)$. And for all $0 < j \leq i$ there was a call to $\mathsf{Query[Eval]}$ with parameters $(pk_{evj}, \alpha_j, m_j, pk'_{evj}, \sigma'_j)$, such that $h_{k-j} = (f_j, m_j, pk'_{evj}, k - j)$ with $(f_j, m_j) = \mathcal{F}(\lambda, f_{j-1}, \alpha_j, pk'_{evj}, m_{j-1})$, but there was no call to $\mathsf{Query[Eval]}$ with parameters $(pk_{evi}, \alpha_i, m_i, pk'_{evi}, \sigma'_i)$, such that $h_{k-i} = (f_i, m_i, pk'_{evi}, k - i)$ with $(f_i, m_i) = \mathcal{F}(\lambda, f_{i-1}, \alpha_i, pk'_{evi}, m_{i-1})$, where $f_0 = f$ and $m_0 = m$. Thus, $\mathcal{B}$ never sent h_i to $\mathsf{Query[Sig]}_{\mathcal{C}}$ and thus, σ_i and h_i fulfill our claim.
- $i = k$: There was a call to $\mathsf{Query[Sig]}$ with parameters $(pk_{ev}, (f, k), m)$. And for all $0 < j \leq k$ there was a call to $\mathsf{Query[Eval]}$ with parameters $(pk_{evj}, \alpha_j, \beta_j, m_j, pk'_{evj}, \sigma'_j)$, such that $h_{k-j} = (f_j, m_j, pk'_{evj}, k - j)$ with $(f_j, m_j) = \mathcal{F}(\lambda, f_{j-1}, \alpha_j, pk'_{evj}, m_{j-1}$. The NIZK makes sure that at most k transformations of the original message exist. Thus, all transformations have been done via calls to $\mathsf{Query[Eval]}$, which means that $(m^*, \sigma^*, pk^*_{ev})$ is not a forgery.

Thus, each forgery of $\mathcal{A}$ constitutes a forgery of a signature σ_x that verifies with a key vk^* that either equals pk_{sig} or a key that has been given to $\mathcal{A}$ as answer to an oracle query $\mathsf{Query[KGenP]}$. Note that if, by chance, $vk^* = vk_{\mathcal{C}}$, then σ_x is a valid forgery for the message h_x. By Claim, Sect. 5.2, $\mathcal{B}$ performs a perfect simulation of a challenger for $\mathsf{Unf}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}, \lambda)$ (from $\mathcal{A}$'s point of view), independent of the value z that $\mathcal{B}$ has chosen in the beginning. As $vk_{\mathcal{C}}$ is randomly placed in the set of possible honest keys ($p(\lambda)$ many), $\mathcal{B}$ produces a forgery for $vk_{\mathcal{C}}$ with probability at least $\frac{1}{p(\lambda)+1}$.

For the analysis of the success of $\mathcal{B}$ let us assume that $\mathcal{A}$ produces a forgery with a non-negligible probability. However, by Claim 5.2, whenever $\mathcal{A}$ produces a forgery, there is a chance of $\frac{1}{p(\lambda)+1}$ that $\mathcal{B}$ will produce a forgery. Since $\mathcal{A}$ is assumed to succeed with a non-negligible probability, $\mathcal{B}$ will also succeed with a non-negligible probability, losing a polynomial factor of $p(\lambda) + 1$. Since $\mathcal{B}$ is an efficient algorithm, this concludes the proof.

Lemma 2. *If $\mathcal{E}$ is a public key encryption scheme that is secure against chosen ciphertext attacks (CCA-2), and the interactive proof scheme NIZK is zero*

knowledge, then the construction DFSS *presented in Sect. 5 is* secure against chosen function attacks (CFA) *as in Definition 4.*

For showing this lemma we will first give a game-based proof for an adversary that only uses the oracle Query[Sign-$\mathcal{F}$] once. We proceed using a hybrid argument that shows that the existence of a successful adversary that makes polynomially many calls to Query[Sign-$\mathcal{F}$] implies the existence of a successful adversary that only makes one call.

Proof. Let $\mathsf{DFSS} = (\mathsf{Setup}, \mathsf{KGen}_{sig}, \mathsf{KGen}_{ev}, \mathsf{Sig}, \mathsf{Eval}_{\mathcal{F}}, \mathsf{Vf})$ be our construction for functionalities $\mathcal{F}$ and $\mathcal{G}$. Assume towards contradiction that DFSS is not secure against chosen function attacks against a strong insider. Then there exists an efficient adversary $\mathcal{A}_{\text{S-Insider}}$ that wins the game $\mathsf{CFA}(\mathsf{DFSS}, \mathcal{F}, \mathcal{A}_{\text{S-Insider}}, \lambda)$

GAME $\mathcal{G}_0$

−−Initialize −−

001 $b := 0$

−−Setup −−

002 $CRS \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda)$

003 $(msk_{\mathcal{S}}, pp_{\mathcal{S}}) \leftarrow \mathsf{Setup}_{\mathcal{S}}(1^\lambda)$

004 $(msk_{\mathcal{E}}, pp_{\mathcal{E}}) \leftarrow \mathsf{Setup}_{\mathcal{E}}(1^\lambda)$

005 $(\widetilde{dk}, \widetilde{ek}) \leftarrow \mathsf{KGen}_{\mathcal{E}}(pp_{\mathcal{E}}, msk_{\mathcal{E}})$

006 $pp := (CRS, pp_{\mathcal{S}}, pp_{\mathcal{E}}, \widetilde{ek})$

007 $msk := (msk_{\mathcal{S}}, msk_{\mathcal{E}})$

−−KGen_{sig} −−

008 $(ssk_{\mathcal{S}}, vk_{\mathcal{S}}) \leftarrow \mathsf{KGen}_{\mathcal{S}}(pp_{\mathcal{S}}, msk_{\mathcal{S}})$

009 $pk_{sig} := vk_{\mathcal{S}}$

010 $sk_{sig} := (ssk_{\mathcal{S}}, pk_{sig})$

− output of (pp, sk_{sig}, pk_{sig}) to $\mathcal{A}$−

011 $c \leftarrow \mathcal{A}_1^{O_{pp, msk, sk_{sig}}}(pp, pk_{sig})$

−−Query[Sign-$\mathcal{F}$] −−

012 parse $c = ([pk_{ev}, \alpha]_0^t, t, m_0, \sigma_0)$

013 if $(\cdot, pk_{ev}[t]) \notin K_{\mathcal{C}}$ $out := \bot$

014 if $\mathsf{Vf}(pp, pk_{sig}, pk_{ev}[0], m_0, \sigma_0) \neq 1$
then output $\bot$

015 if $\neg\exists sk^*_{ev}.\ (sk^*_{ev}, pk_{ev}[0]) \in \mathcal{K}_{\mathcal{C}}$
then output $\bot$

016 extract (f_0, k) from σ_0 using sk^*_{ev}

017 for $i \in \{1, \ldots, t\}$

018 if $\neg\exists sk^*_{ev} = (ssk_{\mathcal{F}}, dk).\ (sk^*_{ev}, pk_{ev}[i-1]) \in \mathcal{K}_{\mathcal{F}}$

019 $out := \bot$

020 $(f_i, m_i) := \mathcal{F}(\lambda, f_{i-1}, \alpha, pk_{ev}[i], m_{i-1})$

021 $q_i := (pp, sk^*_{ev}, pk_{sig}, \alpha[i], m_{i-1}, pk_{ev}[i], \sigma_{i-1})$

022 parse $pk_{ev}[i] = (vk_{\mathcal{F}}, ek)$

023 parse $\sigma_{i-1} = (S, d, \Pi)$

024 $(f_{i-1}, j, \varsigma_j) \leftarrow \mathsf{Dec}_{\mathcal{E}}(pp_{\mathcal{E}}, dk, d)$

025 $x = (pp, pk_{sig}, pk_{ev}[i-1], S, d, m_{i-1})$

026 if $\mathsf{Vf}_{\mathsf{NIZK};Z_j}(CRS, x, \Pi) = 1$

027 $h_{j-1} := (f_i, m_i, pk_{ev}[i], j-1)$

028 $\hat{\varsigma}_{j-1} \leftarrow \mathsf{Sig}_{\mathcal{S}}(pp_{\mathcal{S}}, ssk_{\mathcal{F}}, h_{j-1}; r_{\mathcal{S}})$

029 $s_{j-1} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\hat{\varsigma}_{j-1}, h_{j-1}); r_s)$

030 $\hat{d} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (f_i, j-1, \varsigma_{j-1}); r_d)$

031 $\hat{x} = (pp, pk_{sig}, pk_{ev}[i], S, \hat{d}, m_i)$

032 $\omega = (f_i, j-1, r_d)$

033 with $\omega_{j-1} = (\varsigma_{j-1}, r_{\mathcal{S}}, \Pi, pk_{ev}[i-1], m_{i-1}, f_{i-1}, \alpha[i],)$

034 $\hat{\Pi} \leftarrow P_{\mathsf{NIZK}}(CRS, x, \omega)$

035 $\sigma_i := (S, \hat{d}, \hat{\Pi})$,

036 else

037 $\sigma_i := \bot$

038 if $\sigma_t \neq \bot$

039 $h_{k-t} := (f_t, m_t, pk_{ev}[t], k-t)$

040 $\varsigma_{k-t} \leftarrow \mathsf{Sig}_{\mathcal{S}}(pp_{\mathcal{S}}, ssk_{\mathcal{S}}, h_{k-t}; r_{\mathcal{S}})$

041 $s_{k-t} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\varsigma_{k-t}, h_{k-t}); r_{s_{k-t}})$

042 For $j \in \{0, \ldots, n\} \setminus \{k-t\}$

043 $\varsigma_j := 0^{|\varsigma_{k-t}|}$

044 $h_j := (0^{\ell_p(\lambda)}, 0^{\ell_m(\lambda)}, 0^{|pk_{ev}[t]|}, 0)$

045 $s_j \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\varsigma_j, h_j); r_{s_i})$

046 $d \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (f_t, k-t, \varsigma_{k-t}; r_d))$

047 $S := (s_0, \ldots, s_n)$

048 $x := (pp, pk_{sig}, pk_{ev}[t], S, d, m_t)$

049 $\omega := (f_t, n, r_d)$

050 with $\omega_{k-t} := (\varsigma_{k-t}, r_{\mathcal{S}}, k-t)$

051 $\Pi \leftarrow \mathcal{P}(CRS, x, \omega)$

052 $\sigma := (S, d, \Pi)$

053 else

054 $\sigma := \sigma_t$

055 if $out \neq \bot$ then $out := \sigma$

056 $b^* \leftarrow \mathcal{A}_2(out)$

Fig. 8. Definition of GAME $\mathcal{G}_0$ for Sect. 5.2.

from Definition 4 with non negligible advantage. For simplicity we will write $\mathcal{A}$ for $\mathcal{A}_{\text{S-Insider}}$ in this proof.

Claim. If $\mathcal{A}$ invokes the challenge oracle Query[Sign-$\mathcal{F}$] at most once, then the advantage of $\mathcal{A}$ is negligible.

Proof (Proof of Claim 5.2). The challenger uses the uniformly distributed value b only when Query[Sign-$\mathcal{F}$] is called. Thus, if $\mathcal{A}$ does not call Query[Sign-$\mathcal{F}$], the advantage of $\mathcal{A}$ is 0.

For the case that $\mathcal{A}$ calls Query[Sign-$\mathcal{F}$] exactly once, we show the claim via a series of indistinguishable games that start with a game where $b = 0$ and end with a game $b = 1$. Our proof shows that all intermediate games are indistinguishable.

Let Game $\mathcal{G}_0$ be the original game from Definition 4 where $b = 0$, as defined in Fig. 8. As by our claim $\mathcal{A}$ calls Query[Sign-$\mathcal{F}$] only once we will simplify the notation of the game by making the call to Query[Sign-$\mathcal{F}$] explicit. Moreover we make the invokation of Initialize explicit as we will modify it in the following games. The oracles that $\mathcal{A}$ can access (aside from Query[Sign-$\mathcal{F}$]) are as they are formalized in Definition 4. As before, we annotate each line with the game (first digit) and the line within the game (remaining digits).

Game $\mathcal{G}_1$

102 $(CRS, state) \leftarrow \mathcal{S}_0(1^\lambda)$

134 $\hat{\Pi} \leftarrow \mathcal{S}_1(state, x)$

151 $\Pi \leftarrow \mathcal{S}_1(state, x)$

Game $\mathcal{G}_2$

229 $s_{j-1} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (0^{|\varsigma_{j-1}|}, (0^{\ell_p(\lambda)}, 0^{\ell_m(\lambda)}, 0^{|pk_{ev}[t]|}, 0)); r_s)$

230 $\hat{d} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (0^{|f_t|}, 0, 0^{|\varsigma_{j-1}|}); r_d)$

241 $s_{k-t} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (0^{|\varsigma_{k-t}|}, (0^{\ell_p(\lambda)}, 0^{\ell_m(\lambda)}, 0^{|pk_{ev}[t]|}, 0); r_{s_{k-t}})$

246 $d \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (0^{|f_t|}, 0, 0^{|\varsigma_{k-t}|}); r_d)$

Game $\mathcal{G}_3$

301 $b := 1$

338 if *false*

Game $\mathcal{G}_4$

429 $s_{j-1} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, \widetilde{ek}, (\varsigma_{j-1}, h_{j-1}); r_s)$

430 $\hat{d} \leftarrow \mathsf{Enc}_{\mathcal{E}}(pp_{\mathcal{E}}, ek, (f_t, j-1, \varsigma_{j-1}); r_d)$

Game $\mathcal{G}_5$

501 $CRS \leftarrow \mathsf{KGen}_{\mathsf{NIZK}}(1^\lambda)$

534 $\Pi \leftarrow \mathcal{P}(CRS, x, \omega)$

Game $\mathcal{G}_0$ $\Rightarrow$ Game $\mathcal{G}_1$: Since NIZK is zero knowledge, there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$. In Game $\mathcal{G}_1$, Initialize calls this simulator $\mathcal{S}_0$ to compute the common reference string CRS, instead of the algorithm $\mathsf{Setup}_{\mathsf{NIZK}}$. The simulator is allowed to keep state from $\mathcal{S}_0$ to $\mathcal{S}_1$. Moreover, in Query[Sign-$\mathcal{F}$] we call $\mathcal{S}_1$ to simulate the proof Π instead of computing it by calling the prover $\mathcal{P}$.

Claim. Game $\mathcal{G}_0$ and Game $\mathcal{G}_1$ are computationally indistinguishable.

Proof. The indistinguishability follows from the fact that NIZK is zero knowledge. If a PPT distinguisher could distinguish between Game $\mathcal{G}_0$ and Game $\mathcal{G}_1$, we could construct an efficient distinguisher for NIZK.

Game $\mathcal{G}_1$ $\Rightarrow$ Game $\mathcal{G}_2$: The game Game $\mathcal{G}_2$ is identical to Game $\mathcal{G}_1$ except for the fact that now S and d contain only descriptions of zero-strings: we

put encryptions of zero strings in all s_j for $j \in \{0, \ldots, n\}$ instead of leaving an encryption of a signature ς_{k-t} together with its message h_{k-t} at position $k-t$ and in an encryption of a zero string in d instead of an encryption of ς_{k-t} together with f_t and $k-t$.
To compensate for the loss of information in d, we store the tuple $(f_t, k-t, \varsigma_{k-t})$ together with the (supposed) ciphertext d. Whenever Query[Eval] is called and within the call one of the ciphertexts d is placed, we look up the values $(f_t, k-t, \varsigma_{k-t})$ instead of decrypting d. The same applies to the decryption in line 22 of our game.

Claim. Game $\mathcal{G}_1$ and Game $\mathcal{G}_2$ are computationally indistinguishable.

Proof. If the games could be distinguished by a PPT distinguisher, then we could construct an efficient distinguisher that breaks the CCA-2 security of $\mathcal{E}$. We distinguish two cases:

- The simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ behaves differently. Although the simulatability of the NIZK only is defined for valid statements $x \in L_R$, a simulator that can distinguish with a non-negligible probability between a "normal" S or d (as in Game $\mathcal{G}_1$) and an S or d that consists only of encryptions of zero-strings (as in Game $\mathcal{G}_2$) can also be used to break the CCA-2 security of $\mathcal{E}$.
- The adversary distinguishes the games. If the adversary is able to distinguish Game $\mathcal{G}_1$ and Game $\mathcal{G}_2$ with a non-negligible probability, it can be used to break the CCA-2 security of $\mathcal{E}$.

Thus, Game $\mathcal{G}_1$ and Game $\mathcal{G}_2$ are computationally indistinguishable.

Game $\mathcal{G}_2$ $\Rightarrow$ Game $\mathcal{G}_3$: In Game $\mathcal{G}_3$, the bit b is set to 1 instead of 0. However, b is never used explicitly in the game. Moreover we always use the signature generated by $\mathsf{Eval}_{\mathcal{F}}$ (from line 33) instead of the fresh signature (from line 50).

Claim. Game $\mathcal{G}_2$ and Game $\mathcal{G}_3$ are computationally indistinguishable.

Proof. In both cases S and d are encryptions of zero strings (under the same keys) and in both cases Π is a proof generated by $\mathcal{S}_1$ for the same statement $x = (pp, pk_{sig}, pk_{ev}[t], S, d, m_t)$. Since S_1 does not receive a witness, the proofs are based on the same arguments.

Game $\mathcal{G}_3$ $\Rightarrow$ Game $\mathcal{G}_4$: The game Game $\mathcal{G}_4$ is identical to Game $\mathcal{G}_3$ except for the fact that S and d are "normal" encryptions again (not encryptions of zero strings).

Claim. Game $\mathcal{G}_3$ and Game $\mathcal{G}_4$ are computationally indistinguishable.

Proof. The same argument as for Game $\mathcal{G}_1$ and Game $\mathcal{G}_2$ applies here. If the games could be distinguished, we could construct an efficient distinguisher for the encryption scheme.

Note that we do not need to revert the encryptions in lines 39 and 44 as they are within the "*if false*"-block.

Game $\mathcal{G}_4 \Rightarrow$ Game $\mathcal{G}_5$: In Game $\mathcal{G}_5$ we replace the simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ with the original $\mathsf{Setup}_{\mathsf{NIZK}}$ and $\mathcal{P}$ algorithms again.

Claim. Game $\mathcal{G}_4$ and Game $\mathcal{G}_5$ are computationally indistinguishable.

Proof. As for Claim 5.2, the indistinguishability again follows from the fact that NIZK is zero knowledge. If a PPT distinguisher could distinguish between Game $\mathcal{G}_4$ and Game $\mathcal{G}_5$, we could construct an efficient distinguisher for NIZK.

As we have shown, the games Game $\mathcal{G}_0$ and Game $\mathcal{G}_5$ are computationally indistinguishable. However, Game $\mathcal{G}_0$ perfectly models the case, where an adversary plays against a challenger for CFA when $b = 0$, whereas Game $\mathcal{G}_5$ perfectly models the case, where an adversary plays against a challenger for CFA when $b = 1$. Since the games are (pairwise) computationally distinguishable, the cases are also computationally indistinguishable and thus the advantage of $\mathcal{A}$ is negligible. This concludes the proof for Claim 5.2.

Via hybrid argument we reduce the case in which the adversary might make polynomially many calls to Query[Sign-$\mathcal{F}$] to the case of Claim 5.2 where the adversary makes at most one call to Query[Sign-$\mathcal{F}$]. We can simulate the calls to Query[Sign-$\mathcal{F}$] both for $b = 0$ and for $b = 1$ using the oracle access to Sig and to $\mathsf{Eval}_{\mathcal{F}}$.

Acknowledgments. We would like the Marc Fischlin, Özgür Dagdelen, and Sebastian Gajek for the helpful discussions and their encouragement to submit our work. We also thank the reviewers for the valuable comments. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – www.cispa-security.org) and the project PROMISE. Moreover, it was supported by the Initiative for Excellence of the German federal and state governments through funding for the Saarbrücken Graduate School of Computer Science and the DFG MMCI Cluster of Excellence. Part of this work was also supported by the German research foundation (DFG) through funding for the collaborative research center 1223. Dominique Schröder was also supported by an Intel Early Career Faculty Honor Program Award.

References

1. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Heidelberg (2011)
2. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)

4. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: new privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
5. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
6. Backes, M., Meiser, S., Schröder, D.: Delegatable functional signatures. Cryptology ePrint Archive, Report 408 (2013). http://eprint.iacr.org/
7. Barak, B., Mahmoody-Ghidary, M.: Lower bounds on signatures from symmetric primitives. In: 48th FOCS, pp. 680–688. IEEE Computer Society Press, October 2007
8. Bauer, L., Jia, L., Sharma, D.: Constraining credential usage in logic-based access control. In: 23rd IEEE Computer Security Foundations Symposium, CSF 2010, pp. 154–168. IEEE Computer Society (2010)
9. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
10. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
11. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014)
12. 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)
13. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
14. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
15. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
16. Brands, S.: Restrictive blinding of secret-key certificates. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 231–247. Springer, Heidelberg (1995)
17. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. The MIT Press, Cambridge (2000)
18. Brzuska, C., et al.: Redactable signatures for tree-structured data: definitions and constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
19. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
20. Camenisch, J., Groß, T.: Efficient attributes for anonymous credentials. In: CCS 2008, pp. 345–356. ACM Press, October 2008
21. Camenisch, J.L., 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)

22. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
23. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
24. Catalano, D.: Homomorphic signatures and message authentication codes. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 514–519. Springer, Heidelberg (2014)
25. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (2014)
26. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Succinct malleable NIZKs and an application to compact shuffles. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 100–119. Springer, Heidelberg (2013)
27. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: new definitions and delegatable anonymous credentials. In: 2014 IEEE 27th Computer Security Foundations Symposium (CSF), pp. 199–213. IEEE (2014)
28. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 501–520. Springer, Heidelberg (2009)
29. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
30. Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 201–217. Springer, Heidelberg (2008)
31. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
32. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
33. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
34. Katz, J., Schröder, D., Yerukhimovich, A.: Impossibility of blind signatures from one-way permutations. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 615–629. Springer, Heidelberg (2011)
35. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure preserving signatures and their applications. Des. Codes Crypt. **77**(2–3), 441–477 (2015)
36. Miyazaki, K., Hanaoka, G.: Invisibly sanitizable digital signature scheme. IEICE Trans. Fundament. Electron. Commun. Comput. Sci. **91**(1), 392–402 (2008)
37. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)

Mitigating Multi-target Attacks in Hash-Based Signatures

Andreas Hülsing[1(✉)], Joost Rijneveld[2(✉)], and Fang Song[3(✉)]

[1] Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
andreas@huelsing.net
[2] Digital Security Group, Radboud University, P.O. Box 9010,
6500 GL Nijmegen, The Netherlands
joost@joostrijneveld.nl
[3] Department of Combinatorics and Optimization,
Institute for Quantum Computing, University of Waterloo, Waterloo, Canada
fang.song@uwaterloo.ca

Abstract. This work introduces XMSS-T, a new stateful hash-based signature scheme with tight security. Previous hash-based signatures are facing a loss of security, linear in performance parameters such as the total tree height. Our new scheme can achieve the same security level but using hash functions with a smaller output length, which immediately leads to a smaller signature size. The same techniques also apply directly to the recent stateless hash-based signature scheme SPHINCS (Eurocrypt 2015), and the signature size is reduced as well.

Being a little more specific and technical, the tight security stems from new multi-target notions of hash-function properties which we define and analyze. We show precise complexity for breaking these security properties under both classical and quantum generic attacks, thus establishing a reliable estimate for the quantum security of XMSS-T. Especially, we prove quantum query complexity tailored for cryptographic applications, which overcome some limitations of standard techniques in quantum query complexity such as they usually only consider worst-case complexity. Our proof techniques may be useful elsewhere.

We also implement XMSS-T and compare its performance to that of XMSS (PQCrypto 2011), the most recent stateful hash-based signature scheme before our work.

Keywords: Post-quantum cryptography · Hash-based signatures · Hash function security · Multi-target attacks · Quantum query complexity

Full version of this paper including missing proofs is available at: http://ia.cr/2015/1256. This work was supported by European Commission through the ICT program under contract ICT-645622 (PQCRYPTO). Part of this work was done while the first author was visiting IQC. F.S. acknowledges support by Cryptoworks21, Canada's NSERC and ORF.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 387–416, 2016.
DOI: 10.1007/978-3-662-49384-7_15

1 Introduction

Hash-based signatures are considered the most promising post-quantum alternative to existing schemes RSA and ECDSA which are vulnerable to quantum attacks. This is especially because the security of cryptographic hash functions has been well understood under intensive scrutinization. In addition, there are exact reductionist proofs relating the hardness of breaking the schemes to the hardness of breaking security properties of the hash functions used in the schemes. This allows precise estimation on the security of specific parameter sets.

Traditionally, the security of hash-based signature schemes was related to collision-resistance of the used hash function. In recent years several works focused on basing security on milder assumptions [5,11,12,15,20,22], such as second-preimage resistance and one-wayness. There are two fundamental reasons driving this trend. On the one hand, the attacks against the collision-resistance of SHA1 and MD5 motivated researchers to develop collision-resilient signature schemes [19,26]. On the other hand, collision resistance is subject to birthday attacks while (second-)preimage resistance is not. Hence, to reach a security level of λ bits, a hash function with $n = 2\lambda$ bit digests is needed if collision resistance is required whereas for (second-)preimage resistance only $n = \lambda$ bit digests are needed. Halving the output size of the used hash function immediately halves the signature and key sizes of hash-based signatures.

Multi-target Attacks. The above statement is only half the truth because it bears on the implicit assumption that a hash function is used only once. Clearly, for many cryptographic constructions this is not the case. Consider for example preimage resistance (aka. one-wayness). For many cryptographic constructions, an adversary will be able to learn a magnitude of function values and security breach may occur once he finds a preimage for just one of them. More specifically, suppose that a hash function with n bit outputs is used d times in a cryptographic construction. If it suffices to invert the hash function on any one out of the d outputs to break the security of the scheme, then the attack complexity is downgraded to $\mathcal{O}(2^n/d)$ instead of $\mathcal{O}(2^n)$. Intuitively this is because every input value that an adversary tries has probability $d/2^n$ of being a solution instead of $1/2^n$, if we treat the hash function as a random function. For theoretical (asymptotic) security this worries nobody as d is normally at most polynomial in n. However, when choosing parameters in practice this can easily cause serious consequences.

This issue is indeed very pertinent to hash-based signatures. Consider for example the hash-based signature scheme XMSS [12] and its multi-tree version XMSS^{MT} [22] (see Sect. 4) with parameters that allow to use a keypair for a virtually unlimited amount of signatures (e.g. a total tree height of $h = 60$). In this case, an attacker can learn about 2^{66} images under the same hash function and will succeed in forging a signature if he finds a single preimage for any one of the 2^{66} values. Consequently, to achieve for example security of 256 bits one cannot use a 256 bit hash function but has to use one with output length 322.

This does not only imply the use of a hash function with a bigger output length (and hence a slowdown), it also increases the signature size by roughly 25 %.

This Work. In this work we introduce a new hash-based signature scheme XMSS-T that is not vulnerable to multi-target attacks. Towards this end, we propose two new *multi-target* notions for preimage and second-preimage resistance. We then analyze the generic security of hash functions with regard to these new properties against classical and quantum adversaries, proving upper and lower bounds on the query complexity of generic attacks. More specifically, the first type of notions (single-function multi-target) models a notion that is implicitly used by recent collision-resilient hash-based signature schemes like XMSS, XMSS^{MT} and SPHINCS [5,12,22]. In these notions, an adversary $\mathcal{A}$ receives p target values and a random function from the hash function family. Then, $\mathcal{A}$ is asked to find a preimage (or second-preimage, respectively) for one of the target values under the given function. We prove that compared to standard (second-)preimage resistance, the query complexity of generic attacks drops by a factor p for classical and $\sqrt{p}$ for quantum adversaries. Then we introduce multi-function multi-target notions of preimage and second-preimage resistance. For these notions, $\mathcal{A}$ is given multiple pairs of function and target value, drawn independently at random. It is now $\mathcal{A}$'s goal to find a preimage (or second-preimage, respectively) for one of the target values under the associated function. We prove that in this case the query complexity of generic attacks is exactly the same as for the standard (single-function, single-target) notions.

Given that multi-function multi-target notions are as hard as the standard notions of preimage and second-preimage resistance we construct a new hash-based signature scheme with security based on these new notions. As the basic construction follows that of XMSS, we call the new scheme XMSS-T, indicating XMSS with tightened security. While XMSS looses in the bit security an amount linear in several parameters including the total tree height, XMSS-T looses only two bits, independent of any parameters. The differences between XMSS^{MT} and XMSS-T are a different hash tree and one-time signature scheme construction such that the security can be based on the multi-target multi-function properties. The basic change is that for every hash function call within a hash tree or a hash chain, a different hash function key and different bitmasks are used. Note that XMSS-T is stateful and it may be not suitable in some practical use cases. The good news is that we can make similar changes to the stateless hash-based signature scheme SPHINCS easily. Roughly speaking, it amounts to replacing the used hash trees and one-time signatures by the ones described in this work.

Finally, we present an implementation of XMSS-T and compare it to XMSS and XMSS^{MT}. We show that the applied changes only have marginal performance implications (a factor 3 loss in speed for all algorithms). Our code is available at https://joostrijneveld.nl/papers/multitarget_xmss.

Remarks on Proving Quantum Generic Security. At first sight the tasks of breaking the various security properties for hash functions seem similar to

some standard problems studied in quantum query complexity. However due to some limitations, existing results such as techniques for proving quantum query lower bounds [2,3] cannot be applied directly. For example, there are famous works showing upper and lower bounds on finding collisions in r-to-1 functions [1,9]. Nonetheless, random functions, whose properties our work studies, are very unlikely to be r-to-1. More generally, quantum query complexity usually considers *worst-case* complexity only, whereas in cryptographic settings we care about average-case complexity. Another issue is that, as observed by Zhandry [29], quantum query lower bounds often have implications about quantum algorithms with *high* success probability only. For cryptographic applications however, an attacker with small but noticeable chance of breaking a scheme is still relevant. Therefore, a complete lower bound would be bounding the success probability of any algorithm making a specified number of queries. It might be possible to find fixes by digging into existing works, the situation is yet unclear. We expect that techniques developed in this work can find useful in other cryptographic settings as well.

Organization. We introduce and discuss the new security notions for hash function families in Sect. 2, where detailed analysis for quantum generic security is presented in Section 3. In Sect. 4 we present XMSS-T and discuss its security in Sect. 5. Finally, we present our implementation results in Sect. 6.

Notation. We write $x \xleftarrow{\$} \mathcal{X}$ if x is randomly chosen from the set $\mathcal{X}$ using the uniform distribution. We further write log for $\log_2$. We denote the uniform distribution over bit strings of length n by $\mathcal{U}_n$. We write $m = \text{poly}(n)$ to denote that m is a function, polynomial in n. We call a function $\epsilon(n) : \mathbb{N} \to [0,1]$ negligible and write $\epsilon(n) = \text{negl}(n)$ if for any $c \in \mathbb{N}, c > 0$ there exists a $n_c \in \mathbb{N}$ s.th. $\epsilon(n) < n^{-c}$ for all $n > n_c$.

2 New Security Notions for Hash Function Families

In this section, we recall some known and define several new security notions for (hash) function families and discuss their security against both classical and quantum generic attacks. In the following we restrict ourselves to function families that operate on bit strings and have a fixed input size, as this is the case in our constructions. However, the definitions are the same for the more general case. In the following let $n \in \mathbb{N}$ be the security parameter, $m = \text{poly}(n), k = \text{poly}(n)$, and $\mathcal{H}_n = \{\text{H}_K : \{0,1\}^m \to \{0,1\}^n\}_{K \in \{0,1\}^k}$ be a family of functions. We say a function family $\mathcal{H}_n$ is efficient if there exists a probabilistic polynomial time (PPT) algorithm that evaluates $\text{H}_K(M)$ for any $M \in \{0,1\}^m$ and $K \in \{0,1\}^k$. We require all used functions to be efficient, unless we state otherwise. For hash-based signatures we are mainly interested in functions with $m, k \geq n$. However, we try to keep our results as general as possible and make it explicit whenever we are relying on $m, k \geq n$.

2.1 Defining the Security Notions

Preimage-Resistance (OW). Let's revisit the standard notion of preimage resistance (a.k.a. one-wayness). We define the success probability of an adversary $\mathcal{A}$ against the preimage resistance of a hash function family $\mathcal{H}_n$ as

$$\begin{aligned}\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{H}_n}(\mathcal{A}) = \mathsf{Pr}[\, K \xleftarrow{\$} \{0,1\}^k; M \xleftarrow{\$} \{0,1\}^m, Y \longleftarrow \mathrm{H}_K(M);\\ M' \xleftarrow{\$} \mathcal{A}(K,Y) : Y = \mathrm{H}_K(M')\,]\,. \end{aligned} \tag{1}$$

Single-Function, Multi-target Preimage Resistance (SM-OW). We now define the success probability of an adversary against SM-OW. This is the basic multi-target notion of preimage resistance implicitly used by previous collision resilient hash-based signature schemes like XMSS. We show in Sect. 3 that this notion is significantly easier to attack than standard preimage resistance. The definition takes another parameter p defining the number of targets.

$$\begin{aligned}\mathrm{Succ}^{\text{SM-OW}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}[\, K \xleftarrow{\$} \{0,1\}^k; M_i \xleftarrow{\$} \{0,1\}^m, Y_i \longleftarrow \mathrm{H}_K(M_i), 0 < i \le p;\\ M' \xleftarrow{\$} \mathcal{A}(K,(Y_1,\ldots,Y_p)) : \exists 0 < i \le p, Y_i = \mathrm{H}_K(M')\,]\,. \end{aligned} \tag{2}$$

Multi-Function, Multi-target Preimage Resistance (MM-OW). Next we define the success probability of an adversary $\mathcal{A}$ against MM-OW. This is the notion we are aiming for with XMSS-T as it is as hard to break as standard preimage resistance, as we will show below. Again the definition is parameterized by the number of targets:

$$\begin{aligned}\mathrm{Succ}^{\text{MM-OW}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}[\, K_i \xleftarrow{\$} \{0,1\}^k, M_i \xleftarrow{\$} \{0,1\}^m, Y_i \longleftarrow \mathrm{H}_{K_i}(M_i), 0 < i \le p;\\ (j,M') \xleftarrow{\$} \mathcal{A}((K_1,Y_1),\ldots,(K_p,Y_p)) : Y_j = \mathrm{H}_{K_j}(M')\,]\,. \end{aligned} \tag{3}$$

The difference between these two new definitions is that for SM-OW all targets are for the same function while for MM-OW each target has an associated random function from the family. We decided that $\mathcal{A}$ has to output the associated index i in case of MM-OW as otherwise any reduction would have to search for i and $\mathcal{A}$ knows i for any attack that does better than guessing.

Second-Preimage Resistance (SPR). After presenting the multi-target notions for one-wayness, we now turn to second-preimage resistance. We start revisiting the standard notion of second-preimage resistance. We define the success probability of an adversary $\mathcal{A}$ against the second-preimage resistance (SPR) of a hash function family $\mathcal{H}_n$ as

$$\begin{aligned}\mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{H}_n}(\mathcal{A}) = \mathsf{Pr}[\, K \xleftarrow{\$} \{0,1\}^k; M \xleftarrow{\$} \{0,1\}^m;\\ M' \xleftarrow{\$} \mathcal{A}(K,M) : M' \neq M \wedge \mathrm{H}_K(M) = \mathrm{H}_K(M')\,]\,. \end{aligned} \tag{4}$$

Note that in this definition the adversary is not promised to receive an M that actually has a second-preimage. Hence, especially for families $\mathcal{H}_n$ with $m = n$,

i.e. same size of domain and co-domain, the adversaries success probability is largely influenced by the probability that a random M actually has a second-preimage.

Single-Function, Multi-target Second-Preimage Resistance (SM-SPR). As for one-wayness, we define two multi-target notions: single-function multi-target second-preimage resistance (SM-SPR) and multi-function multi-target second-preimage resistance (MM-SPR). The first one (SM-SPR) is the notion implicitly used in XMSS. The latter is the notion we aim for with XMSS-T that is as hard to break as standard second-preimage resistance, as we will prove below. We start defining the success probability of an adversary against SM-SPR. The definition again takes another parameter p defining the number of targets:

$$\begin{aligned}\mathrm{Succ}^{\text{SM-SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}\,[K \xleftarrow{\$} \{0,1\}^k; M_i \xleftarrow{\$} \{0,1\}^m, 0 < i \leq p;\\ M' \xleftarrow{\$} \mathcal{A}(K,(M_1,\ldots,M_p)) :\\ \exists 0 < i \leq p : M' \neq M_i \wedge \mathrm{H}_K(M_i) = \mathrm{H}_K(M')]\,. \end{aligned} \tag{5}$$

Multi-function, Multi-target Second-Preimage Resistance (MM-SPR). Next we define the success probability of an adversary $\mathcal{A}$ against MM-SPR. Again the definition is parameterized by the number of targets:

$$\begin{aligned}\mathrm{Succ}^{\text{MM-SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}\,[K_i \xleftarrow{\$} \{0,1\}^k, M_i \xleftarrow{\$} \{0,1\}^m, 0 < i \leq p;\\ (j,M') \xleftarrow{\$} \mathcal{A}((K_1,M_1),\ldots,(K_p,M_p)) :\\ M' \neq M_j \wedge \mathrm{H}_{K_j}(M_j) = \mathrm{H}_{K_j}(M')]\,. \end{aligned} \tag{6}$$

Extended Target Collision Resistance (eTCR). In [19] Halevi and Krawczyk introduced extended target collision resistance (eTCR) as a hash function property that is close to target collision resistance. In the classical target-collision resistance game, the adversary is allowed to choose a target message M. Afterwards he learns a function (by learning a key K) and has to find a collision for the M under this function H_K. While the setup of the eTCR game is exactly the same, the adversary wins if he can present a new message M' and a (possibly new) key K' such that $\mathrm{H}_K(M) = \mathrm{H}_{K'}(M')$. Formally, the success probability of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ have shared memory, against ETCR is defined as follows:

$$\begin{aligned}\mathrm{Succ}^{\text{ETCR}}_{\mathcal{H}_n}(\mathcal{A}) = \mathsf{Pr}\,[M \xleftarrow{\$} \mathcal{A}_1(1^n); K \xleftarrow{\$} \{0,1\}^k; (M',K') \xleftarrow{\$} \mathcal{A}_2(K,M) :\\ M' \neq M \wedge \mathrm{H}_K(M) = \mathrm{H}_{K'}(M')]\,. \end{aligned} \tag{7}$$

Multi-target Extended Target Collision Resistance (m-eTCR). We can also define a multi target version (ETCR is inherently multi function anyway). To keep the definition readable we use a challenge oracle $\mathsf{Box}(\cdot)$ that on input of a message outputs a uniformly random function key. This oracle models the ability of $\mathcal{A}$ to adaptively obtain p eTCR challenges for the same function family. We denote by (M_i, K_i) the ith query-answer pair of $\mathsf{Box}(\cdot)$. The success probability

of an adversary $\mathcal{A}$ against m-eTCR that makes no more than p queries to $\mathsf{Box}(\cdot)$ is defined as:

$$\mathrm{Succ}^{\text{M-eTCR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \mathsf{Pr}\left[\, (M', K', i) \xleftarrow{\$} \mathcal{A}^{\mathsf{Box}(\cdot)}(1^n) : M' \neq M_i \wedge \mathrm{H}_{K_i}(M_i) = \mathrm{H}_{K'}(M') \right]. \tag{8}$$

2.2 Generic Security

To determine secure parameters for hash function families or constructions based on them, their security against generic attacks is analyzed. Generic attacks show which security level is achievable at all for a given property as they do not take any possibly existing function specific weaknesses into account. A hash function family is considered broken if the security level for one property is (significantly) lower than the generic security.

Classical Generic Security. The standard way to analyze the complexity of generic attacks against a security property of hash function families is analyzing the success probability of an adversary $\mathcal{A}$ against a random function family to which it is given black box access. The classical security is well understood in the literature. The security of the new notions we defined can be easily established as well. For completeness, we give brief justifications in Appendix A. Table 1 summarizes the classical and quantum generic security.

Quantum Generic Security. When we analyze the properties of hash functions under generic quantum attacks, we treat any hash function as a random function and the adversary can issue quantum superposition queries to the function. Namely, we are essentially working under the quantum random-oracle model [6]. When there are multiple functions, we assume they are independent random functions and the adversary can query them jointly in superposition. Namely, queries in the form of

$$\sum_{K,M,z} \alpha_{K,M,z}|K, M, z\rangle \mapsto \sum_{K,M,z} \alpha_{K,M,z}|K, M, z + H_K(M)\rangle,$$

are permitted[1]. This choice is meant to capture the fact that in reality all hash functions are public, and a quantum adversary can certainly evaluate them jointly in superposition. This is in contrast to the classical setting, where each query must specify an index, and hence the adversary only gets one value of *one* function per query. One can define a similar model in the quantum setting (i.e., each query must specify one and only one function index K) and study all the security properties therein. We stress that this model seems weaker than the one we choose, and in particular our lower bounds results are hence stronger. Namely, they hold against stronger quantum attacks. It is an interesting theoretical question as to determining whether the two models are indeed different.

[1] Alternatively, one can think of it as a global random function $(K, M) \mapsto O(K, M)$.

Table 1. Security against generic classical and quantum attacks. Entries represent the success probability of a q-query adversary (upper and lower bound).

	OW, MM-OW, SPR, MM-SPR	SM-OW, SM-SPR	eTCR	M-eTCR
Classical	$\frac{q+1}{2^n}$	$\frac{(q+1)p}{2^n}$	$\frac{(q+1)}{2^n} + \frac{q}{2^k}$	$\frac{(q+1)p}{2^n} + \frac{qp}{2^k}$
Quantum	$\Theta(\frac{(q+1)^2}{2^n})$	$\Theta(\frac{(q+1)^2 p}{2^n})$	$\Theta(\frac{(q+1)^2}{2^n} + \frac{q^2}{2^k})$	$\Theta(\frac{(q+1)^2 p}{2^n} + \frac{q^2 p}{2^k})$

We prove our results regarding quantum generic security in Sect. 3. Our findings are summarized in Table 1. Please note that the constant hidden in the Θ is small, i.e. 16 for the lower bounds.

3 Analyzing Quantum Generic Security

In the following we establish the generic security of hash function families against quantum attacks on the defined properties. For each security property, we give attacks and analyze their success probabilities. All attacks are based on Grover's quantum search algorithm, but we will need to analyze the complexity for random problem instances. More importantly, we establish matching lower bounds for all cases. The proofs of lower bounds follow a unified structure. Specifically, we first define a family of distributional search problems and bound the success probability of quantum algorithms against these problems. Then, we reduce various instances of the search problem to the task of breaking each of the security properties we care about. The hardness of the distributional search problems hence implies the generic security of hash functions for these security properties.

3.1 Toolbox

(Generalized) Grover's Quantum Search Algorithm. One of the most useful algorithmic tools in quantum computing is Grover's quantum search algorithm and its many generalizations (e.g., [7,8,10,18] to name a few). Here we just need a simple version for searching a universe with multiple marked items. We state it in the following Lemma.

Lemma 1. *Let $f : \mathcal{X} \to \{0,1\}$ be an oracle function and let $\mathcal{X}_f = \{x \in \mathcal{X} : f(x) = 1\}$. Then there is a quantum algorithm* QSearch *with q queries that finds an $x \in \mathcal{X}_f$ with success probability $\Omega(q^2 \frac{|\mathcal{X}_f|}{|X|})$.*

Most of the attacks we describe later will apply QSearch in a straightforward way. However, since our problem instances are generated randomly, we will need to give a new analysis of the average-case performance.

A Hard Average-Case Search Problem. It is well known that Grover's search algorithm is also optimal [4]. Namely, adopting notations from Lemma 1, any q-query algorithm can find a marked item with probability at most

$O(q^2\frac{|X_f|}{|X|})$. However since the security notions we defined all refer to average-case problems, the worst-case lower bound of Grover's search is not very useful. Here we introduce a distributional search problem, and prove a stringent hardness result.

Definition 1. *Let $\mathcal{F} := \{f : \{0,1\}^m \to \{0,1\}\}$ be the collection of all boolean functions on $\{0,1\}^m$. Let $\lambda \in [0,1]$ and $\varepsilon > 0$. Define a family of distributions D_λ on $\mathcal{F}$ such that $f \leftarrow_R D_\lambda$ satisfies*

$$f : x \mapsto \begin{cases} 1 & \text{with prob. } \lambda, \\ 0 & \text{with prob. } 1-\lambda \end{cases}$$

for any $x \in \{0,1\}^m$.

We define $\mathsf{Avg\text{-}Search}_\lambda$ to be the problem that given oracle access to $f \leftarrow D_\lambda$, finds an x such that $f(x) = 1$. For any quantum algorithm $\mathcal{A}$ that makes q queries, we define

$$\mathsf{Succ}^q_\lambda(\mathcal{A}) := \Pr_{f \leftarrow D_\lambda}[f(x) = 1 : x \leftarrow \mathcal{A}^f(\cdot)].$$

Theorem 1. $\mathsf{Succ}^q_\lambda(\mathcal{A}) \le 8\lambda(q+1)^2$ *holds for any quantum algorithm $\mathcal{A}$ with q queries.*

Note that this theorem matches the intuitive argument that for $f \leftarrow D_\lambda$, there are $2^m\lambda$ marked items on average and hence any quantum algorithm needs $\Theta(\sqrt{2^m/(2^m\lambda)}) = \Theta(1/\sqrt{\lambda})$ queries. We defer its proof to the full version.

Simulating Random Functions. In our reductions to show lower bounds, we usually assume we have access to some random function $f : \mathcal{X} \to \mathcal{Y}$. Ultimately, we will need to simulate f efficiently so that any algorithm with q queries cannot notice a difference. Fortunately, the following claim allows us to do so by sampling uniformly from a $2q$*-wise independent hash function family* $\mathcal{H}$.

Lemma 2. *[28, Theorem 6.1] For any quantum adversary that makes no more than q queries to either a truly random function or a function drawn uniformly from $\mathcal{H}$, the final states are identical.*

There exists a vast literature on efficient constructions of t-wise independent hash functions. Interested readers are referred to, e.g., [14,23,24]. There is a technical subtlety though. Most constructions of $\mathcal{H}$ consider output space $\mathcal{Y}$ with size being a prime or a prime power. We need one with $\mathcal{Y} = [N]$, $N = 2^n - 1$. A natural approach is to pick a prime $M >> N$ and construct a $2q$-wise independent family $\mathcal{H}_0 : \mathcal{X} \to [M]$. Then we would expect that $\mathcal{H} : x \mapsto \mathcal{H}_0(x) \mod N$ will suffice for our purpose, modulo a tiny error. However we were unable to identify a rigorous proof in the literature for the correctness of this "mod" construction, especially with respect to quantum attacks. We give a formal proof in the full version.

Sometimes we need a random function f that excludes some output $y \in \mathcal{Y}$. This is easy to realize as follows. We take a random function $g : \mathcal{X} \to [k]$ where $k = |\mathcal{Y}| - 1$. Then $f(x)$ will be obtained by applying g on x and then mapping the outcome to $\mathcal{Y} \backslash y$ according to some canonical isomorphism (e.g., any thing smaller than y remains unchanged, and anything else is incremented by 1.).

3.2 Hardness of Breaking the Security

We analyze in this section the hardness of generic quantum attacks on the various notions of hash functions. We give upper bounds on the success probabilities of any quantum adversary making at most q queries. Basically, we reduce $\mathsf{Avg\text{-}Search}_\lambda$ with various λ to the task of breaking the security notion generically. The hardness of Avg-Search then implies the security against generic quantum attacks. The bounds for OW, SPR and their variants are given in Propositions 1 and 2. While the proofs are quite similar we have to deal with a restriction for the OW notions that we did not figure out how to circumvent. Namely, we require that $2^m \gg 2^n$ (e.g. $m = 2n$) and $p \ll 2^n$, which is the case for most relevant hash function families. The complexity for ETCR and M-ETCR involves additional technical difficulty concerning programming a random oracle, and we analyze them in Proposition 3.

Proposition 1. *Let $m = cn$ for a positive real constant $c > 1$ and $p = o(n)$. For any quantum adversary with q queries, it holds that*

$$\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{H}_n}(\mathcal{A}) = O((q+1)^2/2^n), \mathrm{Succ}^{\mathrm{SM\text{-}OW}}_{\mathcal{H}_n}(\mathcal{A}) = O((q+1)^2 p/2^n),$$
$$\mathrm{Succ}^{\mathrm{MM\text{-}OW}}_{\mathcal{H}_n}(\mathcal{A}) = O((q+1)^2/2^n).$$

The proof is given in the full version.

Proposition 2. *For any quantum adversary with q queries, it holds that*

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{H}}(\mathcal{A}) = O((q+1)^2/2^n), \mathrm{Succ}^{\mathrm{SM\text{-}SPR}}_{\mathcal{H}_n}(\mathcal{A}) = O((q+1)^2 p/2^n),$$
$$\mathrm{Succ}^{\mathrm{MM\text{-}SPR}}_{\mathcal{H}_n}(\mathcal{A}) = O((q+1)^2/2^n).$$

We give the proof for MM-SPR. The others can be proven analogously and are deferred to the full version.

Proof (Hardness of MM-SPR). Given an Avg-Search instance, we construct an instance of MM-OW in Fig. 1:

Note that the way that f is generated ensures that each constructed $\tilde{H}_i$ is distributed identically to a uniformly random function $H : \{0,1\}^m \to \{0,1\}^n$. Therefore the output instance in the reduction is valid according to the definition Eq. 6. This implies that any q-query attacker solving MM-SPR will give rise to a $2q$-query algorithm for $\mathsf{Avg\text{-}Search}_\lambda$. As a consequence

$$\mathrm{Succ}^{\mathrm{MM\text{-}SPR}}_{\mathcal{H}_n}(\mathcal{A}) \leq \mathrm{ADV}^{2q}_{\mathcal{A}}(\lambda) \leq 16(q+1)^2/2^n,$$

Given: $f \leftarrow D_\lambda : \{0,1\}^m \to \{0,1\}^n$, $\lambda = p/2^n$.

1. For $i = 1, \ldots, p$, sample $M_i \leftarrow \{0,1\}^m$ and $y_i \leftarrow \{0,1\}^n$ independently and uniformly at random. Denote $S = \{M_i\}$ and $T = \{y_i\}$.
2. For $i = 1, \ldots, p$, let $g_i : \{0,1\}^m \to \{0,1\}^n \backslash T$ be random function. We construct $\tilde{H}_i : \{0,1\}^m \to \{0,1\}^n$ as follows: for any $x \in \{0,1\}^m$

$$x \mapsto \begin{cases} y_i & \text{if } x = M_i \\ y_i & \text{if } x \neq M_i \wedge f(i\|x) = 1 \\ g_i(x) & \text{otherwise.} \end{cases}$$

Output: MM-SPR instance $(S, \{\tilde{H}_i\}_{i=1}^p)$. Namely an adversary is given M_i and oracle access to $\tilde{H}_i$, and the goal is to find (i^*, x^*) such that $x^* \neq M_{i^*}$ and $\tilde{H}_{i^*}(x^*) = y_{i^*}$.

Fig. 1. Reducing Avg-Search to MM-SPR.

follows by Theorem 1. We remark that, as mentioned in Sect. 3.1, $\tilde{H}_i$ can be implemented efficiently.

Proposition 3. *Let $\varepsilon = 8(q+1)^2/2^n$ and $\delta = 4q^2/2^k$. For any quantum adversary with q queries, it holds that*

$$\mathrm{Succ}^{\mathrm{eTCR}}_{\mathcal{H}_n}(\mathcal{A}) \leq \varepsilon + 2\delta, \quad \mathrm{Succ}^{\text{M-eTCR}}_{\mathcal{H}_n}(\mathcal{A}) \leq p(\varepsilon + 2\delta).$$

To prove the proposition, we need a lemma that allows us to adaptively program a quantum random oracle. The proof follows standard techniques (see similar analyses for different scenarios in [16,27]). Let $\mathcal{A}$ be an arbitrary quantum algorithm and let $H : \{0,1\}^m \times \{0,1\}^k \to \{0,1\}^n$ be a random function. Consider two games as follows:

- Game G_0: $\mathcal{A}$ gets access to H. In phase 1, after making at most q_1 queries to H, $\mathcal{A}$ outputs a message $M \in \{0,1\}^m$. Then a random $\hat{K} \in_R \{0,1\}^k$ is sampled and $(\hat{K}, H_{\hat{K}}(M))$ is handed to $\mathcal{A}$. $\mathcal{A}$ continues to the second phase and makes at most q_2 queries. $\mathcal{A}$ outputs $b \in \{0,1\}$ at the end.
- Game G_1: $\mathcal{A}$ gets access to H. After making at most q_1 queries to H, $\mathcal{A}$ outputs a message $M \in \{0,1\}^m$. Then a random $\hat{K} \in_R \{0,1\}^k$ is sampled as well as a random range element $y \in_R \{0,1\}^n$. Program $H_{\hat{K}}(M) = y$ and call the new oracle H'. $\mathcal{A}$ receives $(\hat{K}, y = H'_{\hat{K}}(M))$ and proceeds to the second phase. After making at most q_2 queries, $\mathcal{A}$ outputs $b \in \{0,1\}$ at the end.

Lemma 3. $|\Pr[\mathcal{A}(G_0) = 1] - \Pr[\mathcal{A}(G_1) = 1]| \leq 2\delta$, *with* $\delta = 4q^2/2^k$.

The proof of the Lemma is described in the full version. Using the Lemma, we can prove Proposition 3.

Proof (Proof of Proposition 3). We give a reduction from Avg-Search to breaking eTCR. Assume that there is $\mathcal{A}$ that breaks eTCR with probability η. We construct an adversary $\mathcal{A}'$ that solves Avg-Search with probability $\eta - 2\delta$. Note that as long as $\mathcal{A}$ does not notice that we reprogrammed H, its view would be identical to that of the standard eTCR game, and by assumption $\mathcal{A}$ wins with probability at least η. By Lemma 3, reprogramming only incurs an additive error $2\delta = 4q^2/2^k$. We claim that $\Pr[f(K^*, M^*) = 1] \geq \eta - 2\delta$. But we know that the success probability of Avg-Search is at most $\varepsilon := 8(q+1)^2/2^n$ by Theorem 1. Therefore $\eta \leq \varepsilon + 2\delta$ and this proves Proposition 3. We can generalize the arguments above to the multi-target case easily.

Given: $f \leftarrow D_\lambda : \mathcal{X} := \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$, $\lambda = 1/2^n$.

1. Sample $y \leftarrow \{0,1\}^n$ uniformly at random.
2. Let $g : \mathcal{X} \to \{0,1\}^n \setminus \{y\}$ be a random function. Construct $H : \mathcal{X} \to \{0,1\}^n$ as follows: for any $x \in \mathcal{X}$

$$x \mapsto \begin{cases} y & \text{if } f(x) = 1, \\ g(x) & \text{otherwise.} \end{cases}$$

3. $\mathcal{A}$ accesses H and issues q_1 queries. $\mathcal{A}$ outputs M at end of Phase 1.
4. Sample $K \leftarrow \{0,1\}^k$. Program $H_K(M) = y$. Denote the new oracle H'. Send (K, y) to $\mathcal{A}$.
5. $\mathcal{A}$ makes q_2 queries to H'. Outputs (K^*, M^*).

Output: (K^*, M^*).

Fig. 2. Reducing Avg-Search to eTCR

3.3 Quantum Attacks

In this section, we apply quantum search algorithm QSearch to attack the various notions generically. In most cases, we get bounds on success probabilities matching the hardness results we have shown in Sect. 3.2.

Proposition 4. *There exist quantum adversaries $\mathcal{A}_1, \ldots, \mathcal{A}_8$ all of which making $\Theta(q)$ queries, such that*

$$\begin{gathered} \mathrm{Succ}^{\mathrm{OW}}_{\mathcal{H}_n}(\mathcal{A}_1) = \Omega(q^2/2^n), \mathrm{Succ}^{\text{SM-OW}}_{\mathcal{H}_n}(\mathcal{A}_2) = \Omega(q^2 p/2^n), \\ \mathrm{Succ}^{\text{MM-OW}}_{\mathcal{H}_n}(\mathcal{A}_3) = \Omega(q^2/2^n); \mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{H}_n}(\mathcal{A}_4) = \Omega(q^2/2^n), \\ \mathrm{Succ}^{\text{SM-SPR}}_{\mathcal{H}_n}(\mathcal{A}_5) = \Omega(q^2 p/2^n), \mathrm{Succ}^{\text{MM-SPR}}_{\mathcal{H}_n}(\mathcal{A}_6) = \Omega(q^2/2^n); \\ \mathrm{Succ}^{\mathrm{eTCR}}_{\mathcal{H}_n}(\mathcal{A}_7) = \Omega(q^2/2^n), \mathrm{Succ}^{\text{M-eTCR}}_{\mathcal{H}_n}(\mathcal{A}_8) = \Omega(q^2 p/2^n). \end{gathered}$$

The proof is adapting standard analysis to the average case. We illustrate the basic idea by proving the case of preimage-resistance. The others are left to the full version.

Proof (Quantum attack on OW*).* We describe a $O(q)$-query attacker $\mathcal{A}_1$ as follows. Given y and oracle access to H, $\mathcal{A}_1$ will apply QSEARCH to search for x such that $H(x) = y$. More specifically, $\mathcal{A}_1$ constructs $g_H : \{0,1\}^m \to \{0,1\}$ such that $g_H(x) = 1$ iff. $H(x) = y$. Each evaluation on g_H can be realized efficiently by two queries to h. For any $h \in \mathcal{H}$, let $p_h := \Pr_{H \leftarrow \mathcal{H}}[H = h]$ and let $X_H = |H^{-1}(y)|$ be the random variable representing the preimage size of y. Then by Lemma 1 we can see that

$$\begin{aligned}\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{H}_n}(\mathcal{A}_1) &= \sum_h p_h \cdot \Omega(q^2 \frac{X_h}{2^m}) = \Omega(\frac{q^2}{2^m} \sum_h p_h X_h) \\ &= \Omega(\frac{q^2}{2^m} \cdot \mathbb{E}(X_H)) = \Omega(\frac{q^2}{2^n}).\end{aligned}$$

In the last step we observe, by linearity of expectation, that

$$\mathbb{E}(X_H) = \sum_{x \in X} \Pr_{H \leftarrow \mathcal{H}}[H(x) = y] = 2^m/2^n.$$

4 XMSS-T

The eXtended Merkle Signature Scheme (XMSS) was proposed by Buchmann, Dahmen, and Hülsing in [12]. The original proposal for XMSS essentially combines a collision-resilient version of the Winternitz one-time signature scheme (WOTS) from [11] with the collision-resilient hash tree construction from [15] and adds two different kinds of pseudorandom key generation, one leading an EU-CMA-secure and one a forward-secure signature scheme. Under the name XMSS^{MT} Hülsing, Rausch, and Buchmann [22] later proposed a multi-tree version of XMSS.

In this work we introduce XMSS-T, XMSS with tightened security. In contrast to XMSS, XMSS-T avoids multi-target attacks. To this end, XMSS-T uses a new hash tree construction and a new WOTS variant WOTS-T. XMSS-T is based on XMSS^{MT}. The main difference in the construction of XMSS^{MT} and XMSS-T is the use of independent function keys and bitmasks for *every* call to a hash function inside of the hash trees or WOTS-T. XMSS^{MT} used a single fixed key per function family and the same bitmask per internal tree level or chain position. The function keys and bitmasks used by XMSS-T are needed for verification. To keep the public key small these values are generated pseudorandomly, using a hash-based pseudorandom function family and a seed value that becomes part of the public key. In the following we describe XMSS-T.

Parameters. XMSS-T uses several parameters and several functions. The main security parameter is $n \in \mathbb{N}$, the message digest length $m \in \text{poly}(n)$. The functions include two keyed, short-input cryptographic hash functions $\text{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $\text{H} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^n$; one arbitrary-input randomized hash function $\mathcal{H} : \{0,1\}^m \times \{0,1\}^* \to \{0,1\}^m$; and two ensembles of pseudorandom function families $\mathcal{F}_n : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$, $\mathcal{F}_m : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$, where we denote by $\{0,1\}^*$ the ability to handle arbitrary input lengths up to some practical limit (e.g. 2^{64} bits as in the case of the SHA family). Of course, these functions can all be built from a single cryptographic hash function, but the security analysis gets easier separating the functions according to the required properties.

XMSS-T uses a hyper-tree (a tree of trees) of total height $h \in \mathbb{N}$, where h is a multiple of d and the hyper-tree consists of d layers of trees, each having height h/d. WOTS allows for a space-time trade-off using the Winternitz parameter $w \in \mathbb{N}, w > 1$. The Winternitz parameter w and the length of the bit string that is signed λ determine ℓ the number of function chains for WOTS:

$$\ell_{1,\lambda} = \left\lceil \frac{\lambda}{\log(w)} \right\rceil, \quad \ell_{2,\lambda} = \left\lfloor \frac{\log(\ell_1(w-1))}{\log(w)} \right\rfloor + 1, \quad \ell_\lambda = \ell_1 + \ell_2.$$

The bit strings signed using WOTS are the m-bit message digests on the lowest layer and the n-bit root nodes of the layer below on all other layers.

As a running example we present concrete numbers for XMSS-T-256; the choices are explained in Sect. 6. For XMSS-T-256 we use $n = 256, m = 316$, $h = 60, d = 3, w = 16$ which leads to $\ell_n = 67$ and $\ell_m = 82$.

Addressing Scheme. XMSS-T requires an addressing scheme for hash function calls. Every addressing scheme that assigns to every call to either F or H within the virtual structure of a XMSS-T hyper-tree a unique address can be used (e.g. numbering all the calls in some order). We suggest to use a recursive addressing scheme that numbers sub-structures (e.g. a OTS key pair) inside a structure (e.g. a tree). The addressing scheme generates an address for a substructure, taking the address of the structure and appending the index of the substructure. For trees, which contain three different kinds of substructures (OTS key pairs, L-trees, and nodes), an additional identifier for the type of substructure is added. Below we assume that a function $\textsc{GenAddr}(\mathbf{a}_s, index)$ exists that takes the address of the structure and the index of the substructure and outputs a unique address for this substructure within an XMSS-T key pair. The advantage of this addressing scheme is that it only uses information that is available when the hash call is executed.

The addressing scheme is publicly known and the same addresses can be used for all XMSS-T key pairs. The resulting addresses are used as inputs to PRF $\mathcal{F}_n$ to pseudorandomly generate function keys and bitmasks.

WOTS-T. We now describe the new WOTS version. The construction differs from [20] in that it uses fresh keys and bitmasks for each hash function call.

We denote the message length by $\lambda \in \{n, m\}$ and to improve readability we write ℓ, ℓ_1, and ℓ_2 instead of $\ell_\lambda, \ell_{1,\lambda}$, and $\ell_{2,\lambda}$. We include pseudorandom key generation, meaning that a seed value takes the place of a secret key in our description. We describe the algorithms as used by XMSS-T, hence, they take global secret and public information. For a standalone version, this information would have to be generated during key generation.

The difference between all WOTS variants is in the way the so called chaining function is constructed. WOTS-T uses the function F to construct the following chaining function:

Chaining Function $c^{i,j}(x, \mathbf{a}_\mathcal{C}, \textsc{Seed})$*:* On input of value $x \in \{0,1\}^n$, iteration counter $i \in \mathbb{N}$, start index $j \in \mathbb{N}$, chain address $\mathbf{a}_\mathcal{C}$, and (public) seed Seed, the chaining function works the following way. In case $i = 0$, c returns x, i.e., $c^{0,j}(x, \mathbf{a}_\mathcal{C}, \textsc{Seed}) = x$. For $i > 0$ we define c recursively as

$$c^{i,j}(x, \mathbf{a}_\mathcal{C}, \textsc{Seed}) = \mathrm{F}(k_{i,j}, c^{i-1,j}(x, \mathbf{a}_\mathcal{C}, \textsc{Seed}) \oplus r_{i,j}),$$

where key $k_{i,j} = \mathcal{F}_n(\textsc{Seed}, \textsc{GenAddr}(\mathbf{a}_\mathcal{C}, 2 \cdot (j + i)))$ and bitmask $r_{i,j} = \mathcal{F}_n(\textsc{Seed}, \textsc{GenAddr}(\mathbf{a}_\mathcal{C}, 2 \cdot (j + i) + 1))$. I.e. in every round, the function first takes the bitwise xor of the previous value $c^{i-1,j}(x, \mathbf{a}_\mathcal{C}, \textsc{Seed})$ and bitmask $r_{i,j}$ and evaluates F with key $k_{i,j}$ on the result.

Now we describe the three algorithms of WOTS-T.

Key Generation Algorithm $((\mathsf{sk}, \mathsf{pk}) \longleftarrow \mathrm{WOTS.kg}(\mathcal{S}, \mathbf{a}_{\mathrm{OTS}}, \textsc{Seed}))$*:* On input of a global secret key seed $\mathcal{S} \in \{0,1\}^n$ (used for every WOTS-T keypair within a XMSS-T keypair), the address of the WOTS-T keypair within a tree $\mathbf{a}_{\mathrm{OTS}}$, and public seed Seed, the key generation algorithm computes the internal secret key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$ as $\mathsf{sk}_i \longleftarrow \mathcal{F}_n(\mathcal{S}, \textsc{GenAddr}(\mathbf{a}_{\mathrm{OTS}}, i))$, i.e., the ℓ n bit secret key elements are derived form the secret key seed using the address of the chain they are contained in. The public key pk is computed as

$$\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell) = (c^{w-1,0}(\mathsf{sk}_1, \mathbf{a}_{\mathcal{C}_1}, \textsc{Seed}), \ldots, c^{w-1,0}(\mathsf{sk}_\ell, \mathbf{a}_{\mathcal{C}_\ell}, \textsc{Seed})),$$

where $\mathbf{a}_{\mathcal{C}_i} = \textsc{GenAddr}(\mathbf{a}_{\mathrm{OTS}}, i)$. Note that $\mathcal{S}$ requires less storage than sk; thus we generate sk and pk on the fly when necessary.

Signature Algorithm $(\sigma \longleftarrow \mathrm{WOTS.sign}(M, \mathcal{S}, \mathbf{a}_{\mathrm{OTS}}, \textsc{Seed}))$*:* On input of a λ-bit message M, the global secret key seed $\mathcal{S} \in \{0,1\}^n$, the address of the WOTS-T keypair within a tree $\mathbf{a}_{\mathrm{OTS}}$, and public seed Seed, the signature algorithm first computes a base-w representation of M: $M = (M_1 \ldots M_{\ell_1})$, $M_i \in \{0, \ldots, w-1\}$. That is, M is treated as the binary representation of a natural number x and then the w-ary representation of x is computed. Next it computes the checksum $C = \sum_{i=1}^{\ell_1}(w - 1 - M_i)$ and its base w representation $C = (C_1, \ldots, C_{\ell_2})$. The length of the base w representation of C is at most ℓ_2 since $C \leq \ell_1(w-1)$. We set $B = (b_1, \ldots, b_\ell) = M \parallel C$, the concatenation of the base w representations of M and C. Then the internal secret key is generated

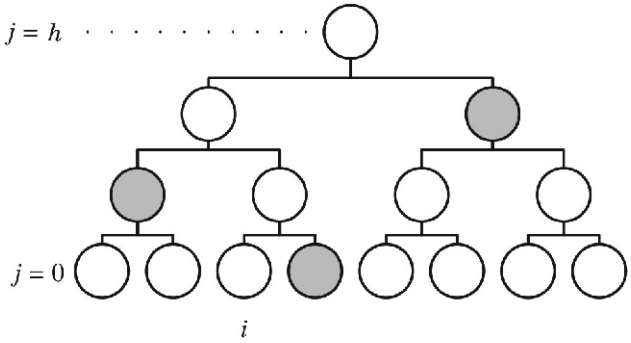


Fig. 3. The authentication path for leaf i.

using $\mathsf{sk}_i \longleftarrow \mathcal{F}_n(\mathcal{S}, \text{GenAddr}(\mathbf{a}_{\text{OTS}}, i))$ the same way as during key generation. The signature is computed as

$$\sigma = (\sigma_1, \ldots, \sigma_\ell) = (c^{b_1,0}(\mathsf{sk}_1, \mathbf{a}_{\mathcal{C}_1}, \text{Seed}), \ldots, c^{b_\ell,0}(\mathsf{sk}_\ell, \mathbf{a}_{\mathcal{C}_\ell}, \text{Seed})),$$

where $\mathbf{a}_{\mathcal{C}_i} = \text{GenAddr}(\mathbf{a}_{\text{OTS}}, i)$ as above.

Verification Algorithm ($\mathsf{pk}' \longleftarrow \text{WOTS.vf}(M, \sigma, \mathbf{a}_{\text{OTS}}, \text{Seed})$)*:* On input of a λ-bit message M, a signature σ, the address of the WOTS-T keypair within a tree $\mathbf{a}_{\text{OTS}}$, and public seed Seed, the verification algorithm first computes the b_i, $1 \leq i \leq \ell$ as described above. Then it returns:

$$\mathsf{pk}' = (\mathsf{pk}'_1, \ldots, \mathsf{pk}'_\ell) = (c^{w-1-b_1,b_1}(\sigma_1, \mathbf{a}_{\mathcal{C}_\ell}, \text{Seed}), \ldots, c^{w-1-b_\ell,b_\ell}(\sigma_\ell, \mathbf{a}_{\mathcal{C}_\ell}, \text{Seed})).$$

A formally correct verification algorithm would compare pk' to a given public key and output true on equality and false otherwise. In XMSS-T this comparison is delegated to the overall verification algorithm.

Binary Hash Trees. The central elements of a Merkle tree signature scheme are full binary hash trees. We use a new construction that allows multi-target-attack resilience. In XMSS-T, a binary hash tree of height h always has 2^h leaves which are n bit strings L_i, $i \in [2^h - 1]$. Each node $N_{i,j}$, for $0 < j \leq h$, $0 \leq i < 2^{h-j}$, of the tree stores an n-bit string. For the leaf nodes define $N_{i,0} = L_i$. The values of the internal nodes $N_{i,j}$ are computed as

$$N_{i,j} = \text{H}_{k_{i,j}}((N_{2i,j-1} \| N_{2i+1,j-1}) \oplus (r_{i,j})),$$

where key $k_{i,j} = \mathcal{F}_n(\text{Seed}, \text{GenAddr}(\mathbf{a}_{\text{Tree}}, 4 \cdot (j + i)))$ and bitmask $r_{i,j} = (\mathcal{F}_n(\text{Seed}, \text{GenAddr}(\mathbf{a}_{\mathcal{C}}, 4 \cdot (j+i)+1)) \| \mathcal{F}_n(\text{Seed}, \text{GenAddr}(\mathbf{a}_{\mathcal{C}}, 4 \cdot (j+i)+2)))$. We also denote the root as $\text{Root} = N_{0,h}$.

An important notion is the authentication path $\mathsf{Auth}_i = (\mathsf{A}_0, \ldots, \mathsf{A}_{h-1})$ of a leaf L_i shown in Fig. 3. Auth_i consists of all the sibling nodes of the nodes contained in the path from L_i to the root. For a discussion on how to compute authentication paths, see Sect. 6. Given a leaf L_i together with its authentication path Auth_i, the root of the tree can be computed using Algorithm 1.

Input: Leaf index i, leaf L_i, authentication path $\mathsf{Auth}_i = (\mathsf{A}_0, \ldots, \mathsf{A}_{h-1})$ for L_i.
Output: Root node ROOT of the tree that contains L_i.

Set $P_0 \leftarrow L_i$;
for $j \leftarrow 1$ **up to** h **do**
  Set $i' = \lfloor i/2^j \rfloor$;
  $$P_j = \begin{cases} \mathrm{H}_{k_{i',j}}((P_{j-1}||\mathsf{A}_{j-1}) \oplus r_{i',j}), & \text{if } \lfloor i/2^{j-1} \rfloor \equiv 0 \mod 2; \\ \mathrm{H}_{k_{i',j}}((\mathsf{A}_{j-1}||P_{j-1}) \oplus r_{i',j}), & \text{if } \lfloor i/2^{j-1} \rfloor \equiv 1 \mod 2; \end{cases}$$
end
return P_h

Algorithm 1. Root Computation

L-Tree. In addition to the full binary trees above, we also use unbalanced binary trees called L-Trees as in [15]. These are exclusively used to hash WOTS-T public keys. The ℓ_λ leaves of an L-Tree are the elements of a WOTS-T public key and the tree is constructed as described above but with one difference: A left node that has no right sibling is lifted to a higher level of the L-Tree until it becomes the right sibling of another node. Apart from this the computations work the same as for binary trees. The L-Trees have height $\lceil \log \ell_\lambda \rceil$.

4.1 XMSS-T

Given all of the above we can finally describe the algorithms of the XMSS-T construction. An XMSS-T keypair completely defines a hyper-tree of height h that consists of d layers of trees of height h/d. Each of these trees looks as follows. The leaves of a tree are $2^{h/d}$ L-Tree root nodes that each compress the public key of a WOTS-T key pair. Hence, a tree can be viewed as a key pair that can be used to sign $2^{h/d}$ messages. The hyper-tree is structured into d layers. On layer $d-1$ it has a single tree. On layer $d-2$ it has $2^{h/d}$ trees. The roots of these trees are signed using the WOTS-T key pairs of the tree on layer $d-1$. In general, layer i consists of $2^{(d-1-i)(h/d)}$ trees and the roots of these trees are signed using the WOTS-T key pairs of the trees on layer $i+1$. Finally, on layer 0 the WOTS-T key pairs are used to sign the message digests.

To improve readability, we only give a functional description of the algorithms of XMSS-T. To obtain a practical scheme, this has to be combined with the distributed signature generation method from [22] which in turn makes use of the BDS algorithm [13] for efficient tree traversal.

Key Generation Algorithm ($(\mathsf{SK}, \mathsf{PK}) \longleftarrow \mathsf{kg}(1^n)$)*:* The key generation algorithm first samples two secret values $(\mathsf{SK}_1, \mathsf{SK}_2) \in \{0,1\}^n \times \{0,1\}^n$. The value $\mathsf{SK}_1 = \mathcal{S}$ is the seed used for pseudorandom key generation in WOTS-T. The value SK_2 is used to generate pseudorandom values to randomize the message hash in sign. Also, the public seed SEED $\xleftarrow{\$} \{0,1\}^n$ is sampled as a uniform random value.

The remaining part of kg consists of generating the root node of the tree on layer $d-1$. Towards this end the WOTS-T key pairs for the single tree on

layer $d-1$ are generated using SK_1 as $\mathcal{S}$. The ith leaf L_i of the tree is the root of an L-Tree that compresses pk_i. Finally, a binary hash tree is built using the constructed leaves and its root node becomes PK_1.

Besides the secret values and SEED, the secret key also contains the index i of the next WOTS-T key pair to use for message signing. The index takes h bits and is initialized with the all 0 bit string. The XMSS-T secret key is $\mathsf{SK} = (i = 0^h, \mathsf{SK}_1, \mathsf{SK}_2, \text{SEED})$, the public key is $\mathsf{PK} = (\mathsf{PK}_1, \text{SEED})$. kg returns the key pair $((\mathsf{SK}_1, \mathsf{SK}_2, \text{SEED}), (\mathsf{PK}_1, \text{SEED}))$.

Signature Algorithm $((\Sigma, \mathsf{SK}) \longleftarrow \mathsf{sign}(M, \mathsf{SK}))$*:* On input of a message $M \in \{0,1\}^*$ and secret key $\mathsf{SK} = (i, \mathsf{SK}_1, \mathsf{SK}_2, \text{SEED})$, sign computes a randomized message digest $D \in \{0,1\}^m$: First, a pseudorandom $R \in \{0,1\}^m$ is computed as $R \longleftarrow \mathcal{F}_m(\mathsf{SK}_2, M)$. Then, $D \longleftarrow \mathcal{H}(R, M)$ is computed as the randomized hash of M using R as randomness. Note that signing is deterministic, i.e., we need no real randomness as all required 'randomness' is pseudorandomly generated using PRF $\mathcal{F}_m$.

Given index i, the ith WOTS-T key pair on layer $d = 0$ is used to sign D. More specifically, this is the i_0th WOTS-T keypair in the i'_0th tree on layer 0, where i_0 is given by the last h/d bits of i and i'_0 by the remaining $(d-1)h/d$ bits of i. Next, the authentication path Auth_{i_0} for the i_0th leaf of the i'_0th tree is computed as well as the root of that tree. Now, for every layer $1 \leq \delta \leq d-1$ the same procedure is repeated with the difference that $i = i'_{\delta-1}$ and the root computed on layer $\delta - 1$ is signed. So, to sign the root from layer $\delta - 1$, the i_δth WOTS-T keypair in the i'_δth tree on layer δ is used, where i_δ is given by the last h/d bits of $i'_{\delta-1}$ and i'_δ by the remaining $(d-1)h/d$ bits of $i'_{\delta-1}$. Then the authentication path Auth_{i_δ} for the i_δth leaf of the i'_δth tree is computed as well as the root of that tree. The XMSS-T signature $\Sigma = (i, R, \sigma_{\mathrm{W},0}, \mathsf{Auth}_{i_0}, \ldots, \sigma_{\mathrm{W},d-1}, \mathsf{Auth}_{i_{d-1}})$ contains the used index i, randomness R and one WOTS-T signature – authentication path pair $\sigma_{\mathrm{W},j}, \mathsf{Auth}_{i_j}$, $j \in [d-1]$ per layer.

Finally, sign updates the secret key SK setting $i = i + 1$ and outputs the pair(Σ, SK).

Verification Algorithm $(b \longleftarrow \mathsf{vf}(M, \Sigma, \mathsf{PK}))$*:* On input of a message $M \in \{0,1\}^*$, a signature Σ, and a public key PK, the algorithm computes the message digest $D \longleftarrow \mathcal{H}(R, M)$ using the randomness R contained in the signature. Using i, the indices i_δ, i'_δ are computed for $0 \leq \delta \leq d-1$. The message digest D and the SEED from PK are used to compute the first WOTS-T public key $\mathsf{pk}_{\mathrm{W},0} \longleftarrow \text{WOTS.}\mathsf{vf}(D, \sigma_{\mathrm{W},0}, \mathbf{a}_{\mathrm{OTS}_0}, \text{SEED})$, where $\mathbf{a}_{\mathrm{OTS}_0}$ is the address of the i_0th WOTS-T keypair in the i'_0th tree on layer 0. An L-Tree is used to compute L_{i_0}, the leaf corresponding to $\mathsf{pk}_{\mathrm{W},0}$. Then, the root ROOT_0 of the respective tree is computed using Algorithm 1 with index i_0, leaf L_{i_0} and authentication path Auth_{i_0}.

Then, this procedure gets repeated for layers 1 to $d-1$ with the following two differences. First, on layer $1 \leq \delta \leq d-1$ the root of the previously processed tree $\text{ROOT}_{\delta-1}$ is used to compute the WOTS-T public key $\mathsf{pk}_{\mathrm{W},\delta}$. Second, the leaf computed from $\mathsf{pk}_{\mathrm{W},\delta}$ using an L-Tree is L_{i_δ}. The result of the final repetition

on layer $d-1$ is a value ROOT_{d-1} for the root node of the single tree on the top layer. This value is compared to the first element of the public key, i.e., $\mathsf{PK}_1 \stackrel{?}{=} \text{ROOT}_{d-1}$. If the comparison holds, vf returns true, otherwise false.

5 Security

In the following we give a security reduction for XMSS-T. First, we review the required security definitions. Afterwards we give a security reduction for XMSS-T.

Existential Unforgeability Under Adaptive Chosen Message Attacks. The standard security notion for digital signature schemes is existential unforgeability under adaptive chosen message attacks (EU-CMA) [17] which is defined using the following experiment. By $\text{DSS}(1^n)$ we denote a signature scheme with security parameter n.

Experiment $\mathsf{Exp}^{\text{EU-CMA}}_{\text{DSS}(1^n)}(\mathcal{A})$
$(\mathsf{sk}, \mathsf{pk}) \longleftarrow \mathsf{kg}(1^n)$
$(\text{Msg}^\star, \sigma^\star) \longleftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
Let $\{(\text{Msg}_i, \sigma_i)\}_1^q$ be the query-answer pairs of $\mathsf{sign}(\mathsf{sk}, \cdot)$.
Return 1 iff $\mathsf{vf}(\mathsf{pk}, \text{Msg}^\star, \sigma^\star) = 1$ and $\text{Msg}^\star \notin \{\text{Msg}_i\}_1^q$.

For the success probability of an adversary $\mathcal{A}$ in the above experiment we write

$$\text{Succ}^{\text{EU-CMA}}_{\text{DSS}(1^n)}(\mathcal{A}) = \mathsf{Pr}\left[\mathsf{Exp}^{\text{EU-CMA}}_{\text{DSS}(1^n)}(\mathcal{A}) = 1\right].$$

A signature scheme is called EU-CMA-secure if any PPT adversary has only negligible success probability:

Definition 2 (EU-CMA). *Let $n \in \mathbb{N}$, DSS a digital signature scheme as defined above. We call* DSS EU-CMA*-secure if for all $q, t = poly(n)$ the maximum success probability* $\text{InSec}^{\text{EU-CMA}}(\text{DSS}(1^n); t, q)$ *of all possibly probabilistic adversaries $\mathcal{A}$ running in time $\leq t$, making at most q queries to* Sign *in the above experiment, is negligible in n:*

$$\text{InSec}^{\text{EU-CMA}}(\text{DSS}(1^n); t, q) \stackrel{def}{=} \max_{\mathcal{A}}\{\text{Succ}^{\text{EU-CMA}}_{\text{DSS}(1^n)}(\mathcal{A})\} = \text{negl}(n).$$

To be precise, XMSS-T is a so-called key-evolving signature scheme which automatically updates the secret key after each signature. We capture this, assuming that the oracle $\mathsf{sign}(\mathsf{sk}, \cdot)$ in the above experiment replaces the secret key sk after each signature with the one returned by XMSS-T.sign and that it returns the empty string when $i \geq 2^h$, i.e. when the maximum number of signatures were done.

Pseudorandom Function Families. In the following we give the missing definition for the properties of (hash) function families that we use, namely pseudorandomness. In our definition we use the definition of (hash) function families from Sect. 2. In the definition of the success probability of an adversary against pseudorandomness (PRF) the adversary gets black-box access to an oracle Box. Box is either initialized with a function from $\mathcal{H}_n$ or a function from the set $\mathcal{G}(m,n)$ of all functions with domain $\{0,1\}^m$ and range $\{0,1\}^n$. The goal of the adversary is to distinguish both cases:

$$\begin{aligned}\mathrm{Succ}^{\mathrm{PRF}}_{\mathcal{H}_n}(\mathcal{A}) = \Big|&\mathsf{Pr}[\mathsf{Box} \xleftarrow{\$} \mathcal{H}_n : \mathcal{A}^{\mathsf{Box}(\cdot)} = 1] \\ &-\mathsf{Pr}[\mathsf{Box} \xleftarrow{\$} \mathcal{G}(m,n) : \mathcal{A}^{\mathsf{Box}(\cdot)} = 1]\Big|. \end{aligned} \tag{9}$$

Using this success probability, we define a pseudorandom function family the following way.

Definition 3 (PRF). *Let $\mathcal{H}_n$ be defined as above. We call $\mathcal{H}_n$ a pseudorandom function family, if it is efficient and for all $t = poly(n)$ the maximum success probability* $\mathrm{InSec}^{\mathrm{PRF}}(\mathcal{H}_n; t)$ *of all possibly probabilistic adversaries $\mathcal{A}$, running in time $\leq t$, is negligible in n:*

$$\mathrm{InSec}^{\mathrm{PRF}}(\mathcal{H}_n; t) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{PRF}}_{\mathcal{H}_n}(\mathcal{A})\} = \mathrm{negl}(n).$$

5.1 Security Reduction

We now proof the security of XMSS-T. We will base the security of the core scheme on the multi-function multi-target second-preimage resistance of F, H, the pseudorandomness of $\mathcal{F}_n$, the multi-target extended target collision resistance of $\mathcal{H}$ and a functional requirement on F defined below in the random oracle model. Please note that the random oracle model is only required to show that we can hand out the seed SEED used to generate the public function keys and bitmasks. Towards this end, we have to split the use of $\mathcal{F}_n$ into two parts. Assume two functions $\mathcal{F}^1_n$ and $\mathcal{F}^2_n$. We assume $\mathcal{F}^1_n$ is used in place of $\mathcal{F}_n$ for pseudorandom (secret) key generation and generation of the message hash randomness. For $\mathcal{F}^1_n$ we require standard model pseudorandomness. On the other hand, $\mathcal{F}^2_n$ is used to replace $\mathcal{F}_n$ when generating the hash keys $k_{i,j}$ and bitmasks $r_{i,j}$. In the proof, *only* $\mathcal{F}^2_n$ is modeled as random oracle (using the concatenation of key and input as input to the RO).

As mentioned above we need an additional requirement on F. Informally we require that every element in the image of F has at least two preimages, i.e.,

$$(\forall k \in \{0,1\}^n)(\forall y \in \mathrm{IMG}(\mathrm{F}_k))(\exists x, x' \in \{0,1\}^n) : x \neq x' \wedge \mathrm{F}_k(x) = f_k(x'). \tag{10}$$

Please note that this requirement meets the expectation for a random function. This additional requirement is needed to not having to use the one-wayness of F. If we had to use the one-wayness of F, we still would have to guess the messages

an adversary sends to the oracle. The reason is that plugging a challenge image into a chain means not knowing any previous value of the chain. Hence, we could not answer a query where the signature contains such a previous value of a chain. This would imply a security loss of roughly h bits. Given the above property we can instead extract a second preimage if $\mathcal{A}$ inverts F with probability 1/2, loosing only 1 bit in the security level.

Now we got everything needed for the security reduction. We proof the following theorem:

Theorem 2. *XMSS-T is existentially unforgeable under adaptive chosen message attacks with respect to the random oracle model if*

- F *and* H *are multi-function multi-target second-preimage resistant function families,*
- F *fulfills the requirement of Eq. 10,*
- $\mathcal{F}_n^1, \mathcal{F}_m$ *are pseudorandom function families,*
- $\mathcal{F}_n^2$ *is modeled as a random oracle, and*
- $\mathcal{H}$ *is an multi -target extend target collision resistant hash function family.*

More specifically, the insecurity function $\mathrm{InSec}^{\text{EU-CMA}}\left(\mathit{XMSS\text{-}T}; \xi, 2^h\right)$ *describing the maximum success probability over all adversaries running in time* $\leq \xi$ *against the* EU-CMA *security of XMSS-T is bounded by*

$$\begin{aligned}&\mathrm{InSec}^{\text{EU-CMA}}\left(\mathit{XMSS\text{-}T}; \xi\right)\\ &\quad \leq \mathrm{InSec}^{\text{PRF}}\left(\mathcal{F}_n^1; \xi\right) + \mathrm{InSec}^{\text{PRF}}\left(\mathcal{F}_m; \xi\right)\\ &\quad + \max\{\mathrm{InSec}^{\text{M-eTCR}}\left(\mathcal{H}; \xi\right), 2\mathrm{InSec}^{\text{MM-SPR}}\left(\mathrm{F}; \xi\right), \mathrm{InSec}^{\text{MM-SPR}}\left(\mathrm{H}; \xi\right)\}\end{aligned}$$

The general idea of the proof follows that of previous hash-based schemes. There are a few mutually exclusive cases what could have happened if an adversary succeeded. First, the attacker could have broken the m-eTCR property of $\mathcal{H}$. This case is easily detected and can be handled in a straight-forward manner. Otherwise, the message digests have to differ. In this case the adversary has found a second preimage or a preimage for F or H with high probability. To extract a second preimage in this case, the reduction takes one MM-SPR challenge (M, K) per hash function call (H and F). Then, for this call the function is keyed with K and the bitmask is selected such that the input to the hash function is M. This means, if the input before the XOR with the bitmask is X, we use $X \oplus M$ as bitmask. Then the RO is programmed such that it generates this bitmask and key for this hash function call. This programming is done adaptively, only when the adversary queries the RO for a value. Now, any second preimage in the scheme will be a valid solution for MM-SPR of either H or F. The full proof can be found in Appendix B.

6 Implementation

In Sect. 4, we described XMSS-T, which builds on XMSS^{MT}, altering the functions that are used to construct WOTS chains and hash trees. We now examine

the cost of this change in terms of computation time. In order to measure the cost of the additional bitmasks and keys that are required for each application of the functions F and H, we have implemented and benchmarked XMSS-T and XMSS^{MT}. We use the BDS tree traversal algorithm [13] to speed up the authentication path computation, making the scheme practical.

We examine the scheme for two parameter configurations from the current Internet Draft for XMSS^{MT} [21], obtaining measurements for both a single-tree and a multi-tree set-up. For both settings, we use $w = 16$ and $m = n = 256$. For the first benchmark, we set $h = 20, d = 1$. We use the same subtree height for the second configuration, setting $h = 60, d = 3$ to construct three layers of subtrees with a height of twenty nodes each. We set $k = 2$ as the BDS parameter for both parameter sets, we rely on the SHA 256 function to construct F and H, and use ChaCha20 as the pseudorandom generator. These choices are also in accordance with [21]. For more parameter sets see [21].

For XMSS these parameters lead to a security level of 190 bits classical and 95 bits quantum for $h = 60$ (230 and 115 for $h = 20$). Following the security analysis in the last section and the lower bounds in Sect. 2, these parameters have a security level of more than 256 bits classical, and 128 bits quantum (assuming that each hash query requires more than 4 bit operations) for XMSS-T (without the message digest). With the message digest we get approximately 190 bits classical and 95 bits quantum like for XMSS as M-ETCR is still vulnerable to multi-target attacks. However, this can be fixed by increasing just the message digest size to $m = 276$ for $h = 20$ and $m = 316$ for $h = 60$. With this change we get 256 bit classical and 128 bit quantum security. However, to get an insight into the effects of the changes made in XMSS-T, we decided to run the experiments for the exact same parameters ($m = 256$). For more benchmarks with different parameters see the full version.

To carry out these benchmarks, we have used a single core of an Intel Core i7-4770K CPU, running at 3.5 GHz, although the implementation was not optimized specifically for this platform.In the first setting, with $h = 20$, we measure an average signing time of 12 488 458 clock cycles per signature for XMSS, and 34 862 033 clock cycles for XMSS-T. For the multi-tree version ($h = 60$), the scheme takes 13 014 401 clock cycles per signature for XMSS, and 37 025 552 cycles for XMSS-T.

This difference is quite significant. However, this was to be expected as the running time of the scheme is largely dominated by applications of F and H – precisely the functions that are changed for XMSS-T. For plain XMSS with the aforementioned parameters, these functions merely consist of calls to SHA-256 with inputs of 256 and 512 bits, respectively. Each of these inputs fits within the internal block size of SHA-256 (512 bits). When considering the Merkle-Damgård construction [25] that defines the structure of SHA-256, this implies a single application of the internal compression function. When transforming F and H into keyed hash functions, the input length increases. To ensure that the key and the input are in separate blocks, the key is prefixed with 256 zero-bits. This results in inputs of 768 and 1024 bits, respectively, implying the need for

two blocks, as well as two applications of the compression function. The straightforward calls to SHA-256 for F and H run in 1 072 and 1 924 cycles, while the keyed variants take 1 932 and 2 812 cycles, respectively.

A bigger factor weighing down F and H is the time needed to generate the keys and bitmasks pseudorandomly. Both these values require calls to the pseudorandom generator. For F, we require two output blocks of 256 bits each; H requires three. At an expense of 560 cycles per output block, generating randomness for the masks and keys carries a significant cost.

Altogether, the experiments show that the tightened security comes at the cost of a factor less than 3 increase in the runtime.

References

1. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. J. ACM **51**(4), 595–605 (2004)
2. Ambainis, A.: Quantum lower bounds by quantum arguments. J. Comput. Syst. Sci. **64**(4), 750–767 (2002)
3. Beals, R., Buhrman, H., Cleve, R., Mosca, M., De Wolf, R.: Quantum lower bounds by polynomials. J. ACM **48**(4), 778–797 (2001)
4. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.: Strengths and weaknesses of quantum computing. SIAM J. Comput. **26**(5), 1510–1523 (1997)
5. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 368–397. Springer, Heidelberg (2015)
6. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011)
7. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. arXiv preprint quant-ph/9605034 (1996)
8. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemp. Math. **305**, 53–74 (2002)
9. Brassard, G., Hoyer, P., Tapp, A.: Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002 (1997)
10. Brassard, G., Høyer, P., Tapp, A.: Quantum counting. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 820–831. Springer, Heidelberg (1998)
11. Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., Rückert, M.: On the security of the Winternitz one-time signature scheme. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 363–378. Springer, Heidelberg (2011)
12. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011)
13. Buchmann, J., Dahmen, E., Schneider, M.: Merkle tree traversal revisited. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 63–78. Springer, Heidelberg (2008)
14. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, pp. 106–112. ACM (1977)

15. Dahmen, E., Okeya, K., Takagi, T., Vuillaume, C.: Digital signatures out of second-preimage resistant hash functions. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 109–123. Springer, Heidelberg (2008)
16. Eaton, E., Song, F.: Making existential-unforgeable signatures strongly unforgeable in the quantum random-oracle model. In: 10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 20–22 May 2015, Brussels, Belgium, pp. 147–162 (2015)
17. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
18. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 212–219. ACM (1996)
19. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
20. Hülsing, A.: W-OTS+ – shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 173–188. Springer, Heidelberg (2013)
21. Hülsing, A., Butin, D., Gazdag, S., Mohaisen, A.: Xmss: extended hash-based signatures draft-irtf-cfrg-xmss-hash-based-signatures-01. Crypto Forum Research Group Internet-Draft (2015). https://tools.ietf.org/html/draft-irtf-cfrg-xmss-hash-based-signatures-01
22. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for XMSS^{MT}. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES Workshops 2013. LNCS, vol. 8128, pp. 194–208. Springer, Heidelberg (2013)
23. Joffe, A., et al.: On a set of almost deterministic k-independent random variables. Ann. Probab. **2**(1), 161–162 (1974)
24. Karloff, H., Mansour, Y.: On construction of k-wise independent random variables. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, pp. 564–573. ACM (1994)
25. Merkle, R.C.: Secrecy, authentication, and public key systems. Ph.D thesis, Stanford University (1979)
26. Mironov, I.: Collision-resistant no more: hash-and-sign paradigm revisited. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 140–156. Springer, Heidelberg (2006)
27. Unruh, D.: Quantum position verification in the random oracle model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 1–18. Springer, Heidelberg (2014)
28. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Heidelberg (2012)
29. Zhandry, M.: A note on the quantum collision and set equality problems. Quantum Inf. Comput. **15**(78), 557–567 (2015)

A Classical Generic Security

Preimage Resistance. For preimage resistance, analysis shows that the success probability of any classical $\mathcal{A}$ that makes q queries to its oracle is

$$\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{H}_n}(\mathcal{A}) = \left(\frac{q+1}{2^n}\right), \tag{11}$$

where the probability is taken over the internal coins of the oracle and the random choices of K and M. An attacker that makes no query but simply outputs a random domain element has success probability 2^{-n} of hitting the target Y. An attacker that makes one query can verify the first guess. If that one did not hit Y he can make another guess which he now can not verify anymore. Together this gives a success probability of $2/2^n$. Iterating this gives the above bound. Consequently, an attacker needs $\mathcal{O}(2^n)$ queries to reach a success probability of at least 0.5.

Single-Function Multi-target Preimage Resistance. For SM-OW a similar analysis shows a bound of

$$\mathrm{Succ}^{\text{SM-OW}}_{\mathcal{H}_n,p}(\mathcal{A}) = \left(\frac{(q+1)p}{2^n}\right), \tag{12}$$

The reason is that the success probability of a single guess is now $p/2^n$. Otherwise, the argument follows along the lines of the above argument. Consequently, the query complexity of a successful attack is $\mathcal{O}(2^n/p)$. Please note, we also can get this result using a reduction from OW. In this case, we replace a random Y_i by the given Y from the OW game. The reduction looses a factor 1/p.

Multi-function Multi-target Preimage Resistance. While the previous cases are more or less known results, for MM-OW we are not aware of any such results. The difference to SM-OW is that the adversary now basically plays p independent OW games at once. In contrast to the OW game $\mathcal{A}$ can not use a query he made to attack Y_i for any other Y_j for $j \neq i$. The reason is that different functions are associated to the different Y_i. So, in the classical case we get a query bound of

$$\mathrm{Succ}^{\text{MM-OW}}_{\mathcal{H}_n,p}(\mathcal{A}) = \left(\frac{q+1}{2^n}\right), \tag{13}$$

The reason is that a guess has success probability $1/2^n$. As one also has to guess (K_i, Y_i), every verification query can only check if a given M fulfills $Y_i = \mathrm{H}_{K_i}(M)$ for a single i. Viewed differently, each query has to fix K_i in advance and outputs the associated Y_i only with probability 2^{-n}. Consequently, we get the same query bound $\mathcal{O}(2^n)$ as for OW.

Second-Preimage Resistance. In the case of second-preimage resistance, the success probability of any $\mathcal{A}$ that makes q queries to its oracle is

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{H}_n}(\mathcal{A}) = \left(\frac{q+1}{2^n}\right), \tag{14}$$

where the probability is taken over the internal coins of the oracle and the random choices of K and M. The bound can easily be derived following the analysis for one-wayness. Consequently, an attacker needs $\mathcal{O}(2^n)$ queries to reach a success probability of at least 0.5.

Single-Function Multi-target Second-Preimage resistance. For SM-SPR a similar analysis shows a bound of

$$\mathrm{Succ}^{\text{SM-SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \left(\frac{(q+1)p}{2^n}\right). \tag{15}$$

Again, the analysis follows along the lines of the respective analysis for SM-OW. Consequently, the query complexity of a successful attack is $\mathcal{O}(2^n/p)$.

Multi-function Multi-target Preimage Resistance. While the previous cases are more or less known results, for MM-SPR we are not aware of any such results. The difference to SM-SPR is that the adversary now basically plays p independent SPR games at once. As for MM-OW, in the classical case, we get a query bound of

$$\mathrm{Succ}^{\text{MM-SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \left(\frac{q+1}{2^n}\right). \tag{16}$$

Again, the analysis follows along the lines of the respective analysis for MM-OW. Consequently, the query complexity of a successful attack is $\mathcal{O}(2^n)$.

Extended Target Collision Resistance. For eTCR, analysis shows that the success probability of any adversary $\mathcal{A}$ that makes no more than q queries to its oracle is

$$\mathrm{Succ}^{\mathrm{eTCR}}_{\mathcal{H}_n}(\mathcal{A}) \leq \left(\frac{q+1}{2^n} + \frac{q}{2^k}\right), \tag{17}$$

where the probability is taken over the internal coins of the oracle and the random choice of K.

Consider an arbitrary adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacking eTCR. A_1 makes q_1 queries and outputs a message M. Afterwards $\mathcal{A}_2$ obtains K and makes q_2 queries, with $q_1 + q_2 = q$. Without loss of generality, we assume that $\mathcal{A}_1$ stores all his query results in the shared memory. When $\mathcal{A}_2$ receives K, we can distinguish two mutually exclusive cases:

Case 1: $\mathcal{A}_1$ already queried the oracle for $\mathrm{H}_K(M)$. To simplify analysis, we consider this a success for $\mathcal{A}$. As K is a random key and $\mathcal{A}_1$ made queries for no more than q_1 different keys, this case occurs with probability

$$\epsilon_1 \leq \frac{q_1}{2^k}.$$

Case 2: $\mathcal{A}_1$ did not query $\mathrm{H}_K(M)$ before. In this case, every query made by $\mathcal{A}_1$ and every query made by $\mathcal{A}_2$ has probability 2^{-n} to hit $\mathrm{H}_K(M)$ and hence to be a solution. If $\mathcal{A}_2$ does not find a solution using all query results, he can make another guess that has the same success probability as the queries before. Hence, the success probability in this case is exactly

$$\epsilon_2 = \frac{q_1 + q_2}{2^n}.$$

The sum of the two bounds $\epsilon_1 + \epsilon_2$ takes its maximum for $q_1 = q$. This gives the claimed bound. Note that the analysis for case 1 above is very rough and could be tightened (This is only a success if $\mathcal{A}_1$ already found a pseudo-collision for (K, M)). However, in all relevant cases we know $k \gtrsim n$ and hence tightening is of little use.

Multi-target-eTCR. Now, switching to M-eTCR the complexities for the two cases change as follows: In both cases a factor of q is lost. In Case 1 this is caused by the fact that there are now q keys returned (over the game) that might hit a previously queried one. In Case 2 this is caused by the fact that each query works for all q targets (as in the case of MM-SPR). This leads to the bound

$$\mathrm{Succ}^{\text{M-eTCR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \frac{(q+1)p}{2^n} + \frac{qp}{2^k}.$$

B Proof of Theorem 2

In the following we omit indices for the MM-SPR challenge pairs to preserve readability. Assume that there exists an adversary $\mathcal{A}$ running in time ξ that breaks the EU-CMA security of XMSS with probability $\epsilon_{\mathcal{A}}$. In the following we will prove that $\epsilon_{\mathcal{A}} \leq \mathrm{InSec}^{\text{EU-CMA}}\left(\text{XMSS-T}; \xi, 2^h\right)$. First, consider the following two games:

Game 1. This is the original game.

Game 2. This is the same as Game 1 but instead of using random elements from $\mathcal{F}_n^1$ and $\mathcal{F}_m$ (by sampling $\mathcal{S}$ and SK_2 from the key space), two truly random functions $\mathrm{G}_n : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ and $\mathrm{G}_m : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$ are used.

The difference in the success probability of $\mathcal{A}$ playing one of these games must be bound by $\mathrm{InSec}^{\text{PRF}}\left(\mathcal{F}_n^1; \xi\right) + \mathrm{InSec}^{\text{PRF}}\left(\mathcal{F}_m; \xi\right)$, otherwise we could use $\mathcal{A}$ to distinguish $\mathcal{F}_n^1$ or $\mathcal{F}_m$ from a truly random function, breaking the pseudorandomness which would contradict the assumption. Hence, it suffices to analyze

the success probability of $\mathcal{A}$ in Game 2. Towards this end, we construct an oracle machine $\mathcal{M}^{\mathcal{A}}$ that breaks either the multi-function multi-target second-preimage resistance of F or of H, or the extended target collision resistance of $\mathcal{H}$. $\mathcal{M}^{\mathcal{A}}$ takes q_1 challenge pairs $\{(K_i, M_i)\}_1^{q_1}$ for F and q_2 challenge pairs $\{(K'_i, M'_i)\}_1^{q_2}$ for H where q_1 (q_2) is the number of calls to F (H) for the XMSS-T key pair. Each of these challenge pairs gets associated with one specific call to F or H within the XMSS-T key pair.

$\mathcal{M}^{\mathcal{A}}$ first samples a random seed $\text{SEED} \xleftarrow{\$} \{0,1\}^n$. The XMSS-T public key PKbecomes $(\mathsf{PK}_1 = \mathrm{H}_{K'_j}(M'_j), \text{SEED})$ for (K'_j, M'_j) – the pair associated with the call to H that computes the root. Now $\mathcal{A}$ is run on this PK. When $\mathcal{A}$ makes his ith query using some message Msg_i, $\mathcal{M}^{\mathcal{A}}$ first sends Msg_i to the M-ETCR challenger, receiving back a function key R_i. Then it computes the message digest as $D_i = \mathcal{H}(R_i, \text{Msg}_i)$ and sets the signature index to i. The next steps are the same for each tree involved in the signature. First, $\mathcal{M}^{\mathcal{A}}$ computes the chain indices b. The WOTS$^+$ signature is collected by selecting the challenge (K, M) for the b_jth call to F in the jth chain and computing the jth signature element as $\mathrm{F}_K(M)$. Similarly, the authentication path for the WOTS$^+$ key pair is generated by figuring out the nodes that are required. Then, these nodes are calculated as $\mathrm{H}_K(M)$ where (K, M) is the challenge pair associated with the call to H that computes this authentication path node. The same is done for the root node. Afterwards, the whole procedure is repeated for the parent tree, until the top tree is done. Then the XMSS-T signature Σ_i is sent back to $\mathcal{A}$.

If $\mathcal{A}$ queries the RO, the RO is adaptively programmed such that it outputs the right bitmasks and keys. W.l.o.g., we assume that $\mathcal{A}$ makes each query only once and stores the result. For any query $(X_1||X_2) \in \{0,1\}^n \times \{0,1\}^*$ where $X_1 \neq \text{SEED}$ or X_2 is no valid address of a bitmask or a key for a hash function call, the RO simply outputs a random value. Otherwise, assume for simplicity (again w.l.o.g.) that the adversary always queries the RO for the bitmask(s) and the key of a hash function call at once. Then the RO is programmed as follows: Assume (K, M) is the challenge pair for the associated hash call. If the address refers to the first hash call in a WOTS$^+$ function chain, the start value of that chain is generated as $X = \mathrm{G}(\mathsf{SK}_1, \mathbf{a})$ where $\mathbf{a}$ is the address of that function chain. Next, the RO is programmed to return bitmask $r = X \oplus M$ and key K. If the address refers to a hash call in a WOTS$^+$ function chain that is not the first in that chain, let (K', M') be the challenge template associated with the previous hash call in that chain. Then the RO is programmed to return bitmask $r = \mathrm{F}_{K'}(M') \oplus M$ and key K. Finally, if the address refers to a hash call to H, let (K_1, M_1), (K_2, M_2) be the challenge templates associated with the hash calls computing its two child nodes. Then the RO is programmed to return bitmask $r = (\mathrm{H}_{K_1}(M_1) \| \mathrm{H}_{K_2}(M_2)) \oplus M$ and key K (Note that the challenge templates might be associated with calls to F if the address is associate with the computation of a node on level 1 in an L-tree. In this case H is replaced by F in the computation of r).

When $\mathcal{A}$ outputs a forgery (Msg, Σ) with $\Sigma = (i, R, \sigma_{\mathrm{W},0}, \mathsf{Auth}_{i_0}, \ldots, \sigma_{\mathrm{W},d-1}, \mathsf{Auth}_{i_{d-1}})$, $\mathcal{M}^{\mathcal{A}}$ runs the verification algorithm on (Msg, Σ) and

$(\Sigma_i, \mathrm{Msg}_i)$. If the forgery is invalid, $\mathcal{M}^{\mathcal{A}}$ returns $\perp$. Otherwise, three mutually exclusive cases can occur. $\mathcal{M}^{\mathcal{A}}$ compares the values computed during the two verification runs in order of computation.

Case 1: If $D = \mathcal{H}(R, \mathrm{Msg}) = \mathcal{H}(R_i, \mathrm{Msg}_i) = D_i$, i.e., if the digests of the ith query is the same as that of the forgery, $\mathcal{M}^{\mathcal{A}}$ broke m-eTCR and returns (i, R, Msg). Hence, the probability that $\mathcal{A}$ outputs a case 1 forgery must be upper bounded by $\mathrm{InSec}^{\text{M-eTCR}}(\mathcal{H}; \xi)$ per assumption.

If the digests are different, the corresponding b_i are also different and hence, parts of the data computed by the two verification runs must also differ. Now, $\mathcal{M}^{\mathcal{A}}$ only compares the computed WOTS$^+$ public keys and the computed root values. By the pigeonhole principle, the signatures have to agree on one of these for the first time as they lead to the same root of the top tree.

Case 2: If the data generated verifying the two signatures first agrees on a WOTS$^+$ public key, the message digests or the root nodes signed with this WOTS$^+$ keypair where different. Hence, we got a WOTS$^+$ forgery. In this case, by the construction of the checksum there must be one chain j in this WOTS$^+$ keypair such that $b_j < (b_j)_i$, i.e. the jth signature value of the forgery belongs to an *earlier* hash call than the one of the answer to the ith query. As both chains end in the same public key value, they most collide at the output of some call to F. If this point is not the $(b_j)_i$th call it has to be a later one. In this case, the input to the colliding call to F computed from the forgery is a second preimage for the challenge template associated with that call to F and $\mathcal{M}^{\mathcal{A}}$ outputs it, breaking MM-SPR of F. Otherwise, the two chains collide on the output of the $(b_j)_i$th call to F, i.e., on $(\sigma_j)_i$, the jth value of the original signature. Let (K, M) be the challenge pair associated with the call to F that produced $(\sigma_j)_i$. According to Eq. 10, $(\sigma_j)_i$ has at least two preimages under F_k. As $\mathcal{A}$ has no information about the preimage, the value X that can be computed from the forgery and that leads $f_K(X) = (\sigma_j)_i$ is unequal to M with at least probability 1/2. In that case, $\mathcal{M}^{\mathcal{A}}$ found a second preimage of M under F_K and outputs it. Otherwise it returns $\perp$. Consequently, the probability that $\mathcal{A}$ outputs a case 2 forgery must be upper bounded by $2\mathrm{InSec}^{\text{MM-SPR}}(\mathrm{F}; \xi)$ per assumption.

Case 3: If the data generated verifying the two signatures first agrees on a root node, the WOTS$^+$ public keys that are used to compute this root node have to differ. A third time by the pigeonhole principle, there must be one call to H between the WOTS$^+$ public key and the root node where the output for the forgery and the correct signature agree for the first time. As the input data depends on previously computed outputs of H (or F), it must differ. Hence, for challenge pair (K, M), the input to this call to H_K is a second preimage for M, that $\mathcal{M}^{\mathcal{A}}$ returns breaking MM-SPR of H. Hence, the probability that $\mathcal{A}$ outputs a case 3 forgery must be upper bounded by $\mathrm{InSec}^{\text{MM-SPR}}(\mathrm{H}; \xi)$.

Combining the upper bounds from the three cases shows that the success probability $\epsilon_{\mathcal{A}}$ of $\mathcal{A}$ winning in Game 2 must be upper bounded by

$$\epsilon_{\mathcal{A}} \leq \max\{\mathrm{InSec}^{\text{M-eTCR}}(\mathcal{H}; \xi), 2\mathrm{InSec}^{\text{MM-SPR}}(\mathrm{F}; \xi), \mathrm{InSec}^{\text{MM-SPR}}(\mathrm{H}; \xi)\}.$$

Combining this with the result that the difference in $\mathcal{A}$'s success probability between playing in Game 1 and playing in Game 2 must be upper bounded by $\mathrm{InSec}^{\mathrm{PRF}}\left(\mathcal{F}_n^2;\xi\right)$, we get the claimed bound on the success probability of any adversary $\mathcal{A}$ running in time ξ:

$$\begin{aligned}\mathrm{Succ}^{\text{EU-CMA}}_{\text{XMSS-T}}(\mathcal{A}) \leq\ & \mathrm{InSec}^{\mathrm{PRF}}\left(\mathcal{F}_n^1;\xi\right) + \mathrm{InSec}^{\mathrm{PRF}}\left(\mathcal{F}_m;\xi\right) \\ & + \max\{\mathrm{InSec}^{\text{M-ETCR}}\left(\mathcal{H};\xi\right), 2\mathrm{InSec}^{\text{MM-SPR}}\left(\mathrm{F};\xi\right), \mathrm{InSec}^{\text{MM-SPR}}\left(\mathrm{H};\xi\right)\}\end{aligned}$$

□

Nearly Optimal Verifiable Data Streaming

Johannes Krupp[1(✉)], Dominique Schröder[1], Mark Simkin[1], Dario Fiore[2], Giuseppe Ateniese[3], and Stefan Nuernberger[1]

[1] CISPA, Saarland University, Saarbrücken, Germany
krupp@ca.cs.uni-saarland.de
[2] IMDEA Software Institute, Madrid, Spain
[3] Stevens Institute of Technology, Hoboken, NJ, USA

Abstract. The problem of verifiable data streaming (VDS) considers the setting in which a client outsources a large dataset to an untrusted server and the integrity of this dataset is publicly verifiable. A special property of VDS is that the client can append additional elements to the dataset without changing the public verification key. Furthermore, the client may also update elements in the dataset. All previous VDS constructions follow a hash-tree-based approach, but either have an upper bound on the size of the database or are only provably secure in the random oracle model. In this work, we give the first unbounded VDS constructions in the standard model. We give two constructions with different trade-offs. The first scheme follows the line of hash-tree-constructions and is based on a new cryptographic primitive called Chameleon Vector Commitment (CVC), that may be of independent interest. A CVC is a trapdoor commitment scheme to a vector of messages where both commitments and openings have constant size. Due to the tree-based approach, integrity proofs are logarithmic in the size of the dataset. The second scheme achieves constant size proofs by combining a signature scheme with cryptographic accumulators, but requires computational costs on the server-side linear in the number of update-operations.

1 Introduction

In this work we study the problem of verifiable data streaming (VDS) [21], where a (computationally weak) client outsources a dataset to an untrusted server, such that the following properties are maintained.

1. Naturally, the space requirement on the client-side should be sublinear in the size of the dataset.
2. Anyone should be able to retrieve arbitrary subsets of the outsourced data along with a corresponding proof and verify their integrity using the client's public key pk.
3. Notably, such a verification must guarantee that elements have not been altered and are maintained in the correct position in the dataset.
4. Appending new elements to the outsourced dataset requires only a single message from the client to the server including the data element and an authentication information.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 417–445, 2016.
DOI: 10.1007/978-3-662-49384-7_16

Table 1. Comparison to previous VDS constructions. Unbounded indicates whether the construction can authenticate an unbounded amount of elements. M is an upper bound on the size of the dataset, N denotes the number of elements in the dataset, and U is the number of update operations. The base of all logarithms is stated explicitly to highlight the hidden factors.

	Unbounded	Stand. model	Assump	Append	Query	Verify	Update	Size	
								π	pk
[21]	✗	✓	Dlog	$\mathcal{O}(\log_2 M)$	$\mathcal{O}(\log_2 M)$	$\mathcal{O}(\log_2 M)$	$\mathcal{O}(\log_2 M)$	$\mathcal{O}(\log_2 M)$	$\mathcal{O}(1)$
[22]	✓	✗	Dlog, RO	$\mathcal{O}(\log_2 N)$	$\mathcal{O}(\log_2 N)$	$\mathcal{O}(\log_2 N)$	$\mathcal{O}(\log_2 N)$	$\mathcal{O}(\log_2 N)$	$\mathcal{O}(1)$
CVC	✓	✓	CDH	$\mathcal{O}(1)$	$\mathcal{O}(\log_q N)$	$\mathcal{O}(\log_q N)$	$\mathcal{O}(\log_q N)$	$\mathcal{O}(\log_q N)$	$\mathcal{O}(1)$
ACC	✓	✓	q-strong DH	$\mathcal{O}(1)$	$\mathcal{O}(U)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

5. The client must be able to update any element in the remote storage efficiently. Any update operation results in a new public key pk', which invalidates the old data element so as to prevent rollback (aka replay) attacks.

Various applications for VDS protocols have been discussed in [21,22]. In this work we propose two novel solutions to the problem described above that are asymptotically and practically faster than all known previous constructions.

1.1 Our Contribution

We propose two novel and fundamentally different approaches to VDS that result in the first practical verifiable data streaming protocols in the standard model in which the size of the outsourced dataset is not bounded a priori (see Table 1 for a comparison). Somewhat surprisingly, our schemes are asymptotically more efficient even compared with solutions in the random oracle model ([22]). Specifically, our contribution consists of the following:

1. We introduce *Chameleon Vector Commitments* (CVCs), a new cryptographic primitive that combines properties of chameleon hashes and vector commitments [6]. We believe that CVCs may also be of independent interest in other contexts and provide a comprehensive formal treatment of this primitive in the full version [9].
2. We combine CVCs with a novel hash-tree-like structure to build a VDS protocol, that is faster than all previously known constructions, unbounded and provably secure in the standard model. This scheme is presented in Sect. 4.
3. We give a second VDS protocol based on signature schemes and cryptographic accumulators from [15], which achieves constant-size proofs, but has computational costs on the server-side linear in the number of update-operations and requires a stronger assumption. This scheme is presented in Sect. 5.
4. We provide a comprehensive evaluation of our schemes based on a full implementation in Java on Amazon EC2 instances, showing that our two constructions achieve practical performances. The results of our performance evaluation can be found in the full version [9].

1.2 Related Work

Verifiable data streaming is a generalization of verifiable databases (VDB) [2] and was first introduced by Schröder and Schröder [21], who also presented a construction for an a priori *fixed* number of elements M. In their construction, all operations have computational cost logarithmic in this upper bound M, and proofs also have size logarithmic in M. Recently, Schröder and Simkin suggested the first VDS protocol that gets rid of such an upper bound [22], but is only provably secure in the random oracle model. Table 1 compares their previous constructions with ours.

VDBs were first introduced by Benabbas, Gennaro, and Vahlis [2] with the main difference, compared to VDS, being that the size of the database is already defined during the setup phase, while in the VDS setting it may be unbounded. Furthermore, in a VDS protocol, elements can be appended to the dataset *non-interactively* by sending a single message to the server. Notably this message does not affect the verification key of the database. VDBs have been extensively investigated in the context of accumulators [4,5,15] and authenticated data structures [12,14,17,24]. More recent works, such as [2] or [6], also have an upper bound on the size of the dataset, and the scheme of [2] is not publicly verifiable.

Other works like streaming authenticated data structures by Papamanthou et al. [16] or the Iris cloud file system [23] consider untrusted cloud storage providers, but require a key update for every appended element.

"Pure" streaming protocols between a sender and possibly multiple clients, such as TESLA and their variants, e.g. [18,19], require the sender and receiver to be loosely synchronized. Furthermore, these protocols do not offer public verifiability. The signature based solution of [19] does not support efficient updates.

Very recently, the underlying technique of chameleon authentication trees has been applied to the problem of equivocation in distributed systems. In particular, Ruffing et al. showed that making conflicting statements to others in a distributed protocol, can be monetarily disincentivized by the use of crypto-currencies such as Bitcoin [20].

1.3 Straw Man Approaches

In a VDS protocol, the server must be able to prove the authenticity of each element under the public verification key. Here we discuss two simple approaches and their drawbacks.

One trivial idea might be to simply sign all elements and their position in the dataset: The client simply stores a key-pair for a signature scheme and the number of elements in the dataset. To append a new element to the dataset the client signs the new element and its position and gives both the new element and the signature to the server. However, it is not clear how to update an element in the dataset efficiently. Simply signing the new element is not sufficient, since the old element is not invalidated, i.e., upon a query for the updated index, the server could simply return the old element instead of the new one.

Another idea to construct a VDS protocol might be to use a Merkle tree over all elements and a signature on the root of this tree: To append a new element, the client recomputes the new Merkle root, signs it, and gives the new element and the signed root to the server. To update an existing entry in the dataset, the client computes the new Merkle root, picks a fresh key-pair for the signature, signs the root and publishes the new verification key of the signature scheme as the public verification key. While this approach does indeed work, it requires computations logarithmic in the size of the dataset on the client-side, whereas our constructions achieve constant-time.

1.4 Our Approach

Similar to previous approaches [21,22] our first scheme uses a Merkle-Tree-like structure to authenticate elements. Every node in the tree authenticates one element in the dataset, and the position in the dataset determines the position of the node in the tree. The root of the tree serves as the public key, and a proof that an element is stored at a certain position in the dataset consists of all nodes along the path from that element's node to the root. However, in contrast to a Merkle-Tree, our tree-structure does not distinguish between inner-nodes, which authenticate their children, and leaf-nodes, which authenticate elements. Instead, every node in our tree-structure authenticates both, a data element *and* its children in the tree. By using CVCs instead of a hash in each node, our tree-structure also allows us to add new layers to the tree *without* changing the root node. Furthermore, CVCs also allow us to increase the arity of the tree without increasing the size of proofs. Finally, by exploiting the arithmetic structure of CVCs, our first scheme allows to append new elements by sending a constant size message from the client to the server.

Our second scheme follows a completely different approach. As in the first straw man approach, the client holds a key-pair for a signature scheme and signs elements and their position in the dataset. We solve the problem of updates, by using a cryptographic accumulator as a black-list for invalidted signatures. To prevent the rollback attacks, in our scheme, the server has to give a proof-of-non-membership, to prove that a signature had not been revoked by some previous update.

2 Preliminaries

In this section we define our notation and describe some cryptographic primitives and assumptions used in this work.

The security parameter is denoted by λ. $a||b$ refers to an encoding that allows to uniquely recover the strings a and b. If A is an efficient (possibly randomized) algorithm, then $y \leftarrow A(x;r)$ refers to running A on input x using randomness r and assigning the output to y.

2.1 Bilinear Maps

Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic multiplicative groups of prime order p, generated by g_1 and g_2, respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$ be a bilinear pairing with the following properties:

1. Bilinearity $e(P^a, Q^b) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1$ and all $Q \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$;
2. Non-degeneracy $e(g_1, g_2) \neq 1$;
3. Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P \in \mathbb{G}_1$ and all $Q \in \mathbb{G}_2$.

If $\mathbb{G}_1 = \mathbb{G}_2$ then the bilinear map is called *symmetric*. A *bilinear instance generator* is an efficient algorithm that takes as input the security parameter 1^λ and outputs a random tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$.

2.2 Computational Assumptions

Now we briefly recall the definitions of the Computational Diffie-Hellman (CDH) assumption, the Square-Computational Diffie-Hellman (Square-CDH) assumption, and the q-strong Diffie-Hellman assumption.

Assumption 1 (CDH). *Let $\mathbb{G}$ be a group of prime order p, let $g \in \mathbb{G}$ be a generator, and let a, b be two random elements from $\mathbb{Z}_p$. The Computational Diffie-Hellman assumption holds in $\mathbb{G}$ if for every PPT adversary $\mathcal{A}$ the probability* $\text{Prob}\left[\mathcal{A}(g, g^a, g^b) = g^{ab}\right]$ *is negligible in λ.*

Assumption 2 (Square-CDH assumption). *Let $\mathbb{G}$ be a group of prime order p, let $g \in \mathbb{G}$ be a generator, and let a be a random element from $\mathbb{Z}_p$. The Square Computational Diffie-Hellman assumption holds in $\mathbb{G}$ if for every PPT adversary $\mathcal{A}$ the probability* $\text{Prob}\left[\mathcal{A}(g, g^a) = g^{a^2}\right]$ *is negligible in λ.*

It is worth noting that the Square-CDH assumption has been shown equivalent to the standard CDH assumption [1,13].

Assumption 3 (q-strong DH assumption). *Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a tuple generated by a bilinear instance generator, and let s be randomly chosen in $\mathbb{Z}_p^*$. We say that the q-Strong DH (q-SDH) assumption holds if every PPT algorithm $\mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, e, g, g^s, g^{s^2}, \ldots, g^{s^q})$ has negligible probability (in λ) of returning a pair $(c, g^{1/(s+c)}) \in \mathbb{Z}_p^* \times \mathbb{G}$.*

2.3 Chameleon Hash Functions

A chameleon hash function is a randomized collision-resistant hash function that provides a trapdoor to find collisions. This means that given the trapdoor *csk*, a message m, some randomness r and another message m', it is possible to find a randomness r' s.t. $\mathsf{Ch}(m; r) = \mathsf{Ch}(m'; r')$.

Definition 1 (Chameleon Hash Function). *A* chameleon hash function *is a tuple of PPT algorithms* $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{Ch}, \mathsf{Col})$*:*

$\mathsf{CHGen}(1^\lambda)$**:** *The key generation algorithm returns a key-pair* (csk, cpk) *and sets* $\mathsf{Ch}(\cdot) := \mathsf{Ch}(cpk, \cdot)$*.*

$\mathsf{Ch}(m; r)$**:** *The input of the hashing algorithm is a message* m *and some randomness* $r \in \{0,1\}^\lambda$ *and it outputs a hash value.*

$\mathsf{Col}(csk, m, r, m')$**:** *Upon input of the trapdoor* csk*, a message* m*, some randomness* r *and another message* m'*, the collision finding algorithm returns some randomness* r' *s.t.* $\mathsf{Ch}(m; r) = \mathsf{Ch}(m'; r')$*.*

Uniform Distribution: *The output of* Ch *is uniformly distributed, thus the output of* $\mathsf{Ch}(m; r)$ *is independent of* m*. Furthermore, the distribution of* $\mathsf{Col}(csk, m, r, m')$ *is identical to the distribution of* r*.*

A chameleon hash is required to be collision-resistant, i.e., no PPT adversary should be able to find (m, r) and (m', r') s.t. $(m, r) \neq (m', r')$ and $\mathsf{Ch}(m; r) = \mathsf{Ch}(m'; r')$.

Definition 2 (Collision Resistance). *A chameleon hash* $\mathcal{CH}$ *is* collision-resistant *if the success probability for any PPT adversary* $\mathcal{A}$ *in the following game is negligible in* λ*:*

Experiment $\mathsf{HashCol}_{\mathcal{A}}^{\mathcal{CH}}(\lambda)$
$(csk, cpk) \leftarrow \mathsf{CHGen}(1^\lambda)$
$(m, m', r, r') \leftarrow \mathcal{A}(\mathsf{Ch})$
if $\mathsf{Ch}(m; r) = \mathsf{Ch}(m'; r')$ *and* $(m, r) \neq (m', r')$ *output* 1
else output 0

2.4 Vector Commitments

A vector commitment [6] is a commitment to an ordered sequence of messages, which can be opened at each position individually. Furthermore, a vector commitment provides an interface to update single messages and their openings.

Definition 3 (Vector Commitment). *A* vector commitment *is a tuple of PPT algorithms* $\mathcal{VC} = (\mathsf{VGen}, \mathsf{VCom}, \mathsf{VOpen}, \mathsf{VVer}, \mathsf{VUpdate}, \mathsf{VProofUpdate})$*:*

$\mathsf{VGen}(1^\lambda, q)$**:** *The key generation algorithm gets as input the security parameter* λ *and the size of the vector* q *and outputs some public parameters* pp*.*

$\mathsf{VCom}_{\mathsf{pp}}(m_1, \ldots, m_q)$**:** *Given* q *messages, the commitment algorithm returns a commitment* C *and some auxiliary information* aux*, which will be used for proofs and updates.*

$\mathsf{VOpen}_{\mathsf{pp}}(m, i, \mathsf{aux})$**:** *The opening algorithm takes as input a message* m*, a position* i *and some auxiliary information* aux *and returns a proof* Λ_i *that* m *is the message at position* i*.*

$\mathsf{VVer}_{\mathsf{pp}}(C, m, i, \Lambda_i)$**:** *The verification algorithm outputs 1 only if* Λ_i *is a valid proof that* C *was created to a sequence with* m *at position* i*.*

$\mathsf{VUpdate}_{\mathsf{pp}}(C, m, m', i)$**:** *The update algorithm allows to change the i-th message in C from m to m'. It outputs an updated commitment C' and some update-information U.*

$\mathsf{VProofUpdate}_{\mathsf{pp}}(C, \Lambda_j, m', i, U)$**:** *The proof-update algorithm may be run by anyone holding a proof Λ_j that is valid w.r.t. C to obtain the updated commitment C' and an updated proof Λ'_j that is valid w.r.t. C'.*

The security definition of vector commitments requires a vector commitment to be position-binding, i.e., no PPT adversary $\mathcal{A}$ given pp can open a commitment to two different messages at the same position. Formally:

Definition 4 (Position-Binding). *A vector commitment is* position-binding *if the success probability for any PPT adversary $\mathcal{A}$ in the following game is negligible in λ:*

Experiment $\mathsf{PosBdg}_{\mathcal{A}}^{\mathcal{VC}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{VGen}(1^\lambda, q)$
$(C, m, m', i, \Lambda, \Lambda') \leftarrow \mathcal{A}(\mathsf{pp})$
if $m \neq m' \wedge \mathsf{VVer}_{\mathsf{pp}}(C, m, i, \Lambda) \wedge \mathsf{VVer}_{\mathsf{pp}}(C, m', i, \Lambda')$ *output* 1
else output 0.

2.5 The Bilinear-Map Accumulator

We briefly recall the accumulator based on bilinear maps first introduced by Nguyen [15]. For simplicity, we describe the accumulator using symmetric bilinear maps. A version of the accumulator using asymmetric pairings can be obtained in a straightforward way, and our implementation does indeed work on asymmetric MNT curves.

For some prime p, the scheme accumulates a set $\mathcal{E} = \{e_1, \ldots, e_n\}$ of elements from $\mathbb{Z}_p^*$ into an element $f'(\mathcal{E})$ in $\mathbb{G}$. Damgård and Triandopoulos [7] extended the construction with an algorithm for issuing constant size proofs of non-membership, i.e., proofs that an element $e \notin \mathcal{E}$. The public key of the accumulator is a tuple of elements $\{g^{s^i} | 0 \leq i \leq q\}$, where q is an upper bound on $|\mathcal{E}| = n$ that grows polynomially with the security parameter $\lambda = \mathcal{O}(\log p)$. The corresponding secret key is s. More precisely, the accumulated value $f'(\mathcal{E})$ is defined as

$$f'(\mathcal{E}) = g^{(e_1+s)(e_2+s)\ldots(e_n+s)}.$$

The *proof of membership* is a witness A_{e_i} which shows that an element e_i belongs to the set $\mathcal{E}$ and it is computed as

$$A_{e_i} = g^{\prod_{e_j \in \mathcal{E} : e_j \neq e_i} (e_j+s)}.$$

Given the witness A_{e_i}, the element e_i, and the accumulated values $f'(\mathcal{E})$, the verifier can check that

$$e(A_{e_i}, g^{e_i} \cdot g^s) = e(f'(\mathcal{E}), g).$$

The proof of *non-membership*, which shows that $e_i \notin \mathcal{E}$, consists of a pair of witnesses $\hat{w} = (w, u) \in \mathbb{G} \times \mathbb{Z}_p^*$ with the requirements that:

$$u \neq 0$$
$$(e_i + s) | [\prod_{e \in \mathcal{E}} (e + s) + u]$$
$$w^{(e_i + s)} = f'(\mathcal{E}) \cdot g^u$$

The verification algorithm in this case checks that

$$e(w, g^{e_i} \cdot g^s) = e(f'(\mathcal{E}) \cdot g^u, g).$$

The authors also show that the proof of non-membership can be computed efficiently without knowing the trapdoor s.

The security of this construction relies on the q-strong Diffie-Hellman assumption. In particular, Nguyen showed that the accumulator is collision-resistant under the q-SDH assumption:

Lemma 1 ([15]). *Let λ be the security parameter and $t = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a tuple of bilinear pairing parameters. Under the q-SDH assumption, given a set of elements $\mathcal{E}$, the probability that, for some s chosen at random in $\mathbb{Z}_p^*$, any efficient adversary $\mathcal{A}$, knowing only $t, g, g^s, g^{s^2}, \ldots, g^{s^q}$ $(q \geq |\mathcal{E}|)$, can find a set $\mathcal{E}' \neq \mathcal{E}$ $(q \geq |\mathcal{E}'|)$ such that $f'(\mathcal{E}') = f'(\mathcal{E})$ is negligible.*

Damgård and Triandopoulos showed that:

Lemma 2 ([7]). *Under the q-SDH assumption, for any set $\mathcal{E}$ there exists a unique non-membership witness with respect to the accumulated value $f'(\mathcal{E})$ and a corresponding efficient and secure proof of non-membership verification test.*

2.6 Verifiable Data Streaming

A VDS protocol [21] allows a client, who possesses a private key, to store a large amount of ordered data $d_1, d_2, \ldots$ on a server in a verifiable manner, i.e., the server can neither modify the stored data nor append additional data. Furthermore, the client may ask the server about data at a position i, who then has to return the requested data d_i along with a publicly verifiable proof $\tilde{\pi}_i$, which proves that d_i was actually stored at position i. Formally, a VDS protocol is defined as follows:

Definition 5 (Verifiable Data Streaming). *A* verifiable data streaming *protocol $\mathcal{VDS} = (\mathsf{Setup}, \mathsf{Append}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ is a protocol between a client $\mathcal{C}$ and a server $\mathcal{S}$, which are both PPT algorithms. The server holds a database DB.*

$\mathsf{Setup}(1^\lambda)$**:** *The setup algorithm takes as input the security parameter and generates a key-pair (pk, sk), gives the public verification key pk to the server $\mathcal{S}$ and the secret key sk to the client $\mathcal{C}$.*

$\mathsf{Append}(sk, d)$**:** *The append protocol takes as input the secret key and some data d. During the protocol, the client $\mathcal{C}$ sends a single message to the server $\mathcal{S}$, who will then store the new item d in DB. This protocol may output a new secret key sk' to the client, but the public key does not change.*

$\mathsf{Query}(pk, DB, i)$**:** *The query protocol runs between $\mathcal{S}(pk, DB)$ and $\mathcal{C}(i)$. At the end the client will output the i-th entry of the database DB along with a proof $\tilde{\pi}_i$.*

$\mathsf{Verify}(pk, i, d, \tilde{\pi}_i)$**:** *The verification algorithm outputs d, iff d is the i-th element in the database according to $\tilde{\pi}_i$. Otherwise it outputs $\perp$.*

$\mathsf{Update}(pk, DB, sk, i, d')$**:** *The update protocol runs between $\mathcal{S}(pk, DB)$ and $\mathcal{C}(i, d')$. At the end, the server will update the i-th entry of its database DB to d' and both parties will update their public key to pk'. The client may also update his secret key to sk'.*

Intuitively the security of a VDS protocol demands that an attacker should not be able to modify stored elements nor should he be able to add further elements to the database. In addition, the old value of an updated element should no longer verify. This can be formalized in the following game VDSsec:

Setup: First, the challenger generates a key-pair $(sk, pk) \leftarrow \mathsf{Setup}(1^\lambda)$. It sets up an empty database DB and gives the public key vp to the adversary $\mathcal{A}$.

Streaming: In this adaptive phase, the adversary $\mathcal{A}$ can add new data by giving some data d to the challenger, which will then run $(sk', i, \tilde{\pi}_i) \leftarrow \mathsf{Append}(sk, d)$ to append d to its database. The challenger then returns $(i, \tilde{\pi}_i)$ to the adversary. $\mathcal{A}$ may also update existing data by giving a tuple (d', i) to the challenger, who will then run the update protocol $\mathsf{Update}(pk, DB, sk, i, d')$ with the adversary $\mathcal{A}$. The challenger will always keep the latest public key pk^* and a ordered sequence of the database $Q = \{(d_1, 1), \ldots, (d_{q(\lambda)}, q(\lambda))\}$.

Output: To end the game, the adversary $\mathcal{A}$ can output a tuple $(d^*, i^*, \hat{\pi})$. Let $\hat{d} \leftarrow \mathsf{Verify}(pk^*, i^*, d^*, \hat{\pi})$. The adversary wins iff $\hat{d} \neq \perp$ and $(\hat{d}, i^*) \notin Q$.

Definition 6 (Secure VDS). *A VDS protocol is* secure*, if the success probability of any PPT adversary in the above game* VDSsec *is at most negligible in λ.*

3 Chameleon Vector Commitments

In this section we introduce chameleon vector commitments (CVCs). CVCs extend the notion of vector commitments [6,10]. CVCs allow one to commit to an ordered sequence of messages in such a way that it is possible to open each position individually, and the commitment value as well as the openings are concise, i.e., of size independent of the length of the message vector. In addition CVCs satisfy a novel chameleon property, meaning that the holder of a trapdoor may replace messages at individual positions *without* changing the commitment value.

3.1 Defining CVCs

We define CVCs as a tuple of seven efficient algorithms: a key generation algorithm CGen to compute a set of public parameters and a trapdoor, a commitment algorithm CCom to commit to a vector of messages, an opening algorithm COpen to open a position of a commitment, a collision finding algorithm CCol that uses the trapdoor to output the necessary information for opening a commitment to a different value, an updating algorithm $\mathsf{CUpdate}$ to update the values in a commitment without recomputing the entire commitment, a proof update algorithm $\mathsf{CProofUpdate}$ to update proofs accordingly, and a verification algorithm CVer to verify the correctness of an opening w.r.t. a commitment.

Definition 7 (Chameleon Vector Commitment). *A* chameleon vector commitment *is a tuple of PPT algorithms* $\mathcal{CVC} = (\mathsf{CGen}, \mathsf{CCom}, \mathsf{COpen}, \mathsf{CVer}, \mathsf{CCol}, \mathsf{CUpdate}, \mathsf{CProofUpdate})$ *working as follows:*

Key Generation $\mathsf{CGen}(1^\lambda, q)$**:** *The key generation algorithm takes as inputs the security parameter* λ *and the vector size* q*. It outputs some public parameters* $\mathtt{pp}$ *and a trapdoor* $\mathtt{td}$*.*

Committing $\mathsf{CCom}_{\mathtt{pp}}(m_1, \ldots, m_q)$**:** *On input of a list of* q *ordered messages, the committing algorithm returns a commitment* C *and some auxiliary information* $\mathtt{aux}$*.*

Opening $\mathsf{COpen}_{\mathtt{pp}}(i, m, \mathtt{aux})$**:** *The opening algorithm returns a proof* π *that* m *is the* i*-th committed message in a commitment corresponding to* $\mathtt{aux}$*.*

Verification $\mathsf{CVer}_{\mathtt{pp}}(C, i, m, \pi)$**:** *The verification algorithm returns 1 iff* π *is a valid proof that* C *was created on a sequence of messages with* m *at position* i*.*

Collision finding $\mathsf{CCol}_{\mathtt{pp}}(C, i, m, m', \mathtt{td}, \mathtt{aux})$**:** *The collision finding algorithm returns a new auxiliary information* $\mathtt{aux}'$ *such that the pair* $(C, \mathtt{aux}')$ *is indistinguishable from the output of* $\mathsf{CCom}_{\mathtt{pp}}$ *on a vector of* q *messages with* m' *instead of* m *at position* i*.*

Updating $\mathsf{CUpdate}_{\mathtt{pp}}(C, i, m, m')$**:** *The update-algorithm allows to update the* i*-th message from* m *to* m' *in the commitment* C*. It outputs a new commitment* C' *and an update information* U*, which can be used to update both* $\mathtt{aux}$ *and previously generated proofs.*

Updating Proofs $\mathsf{CProofUpdate}_{\mathtt{pp}}(C, \pi_j, i, U)$**:** *The proof-update-algorithm allows to update a proof* π_j *that is valid for position* j *w.r.t.* C *to a new proof* π'_j *that is valid w.r.t.* C' *using the update information* U*.*

A tuple $\mathcal{CVC}$ of algorithms as defined above is a chameleon vector commitment if it is *correct*, *concise* and *secure*.

Conciseness is a property about the communication efficiency of CVCs which is defined as follows:

Definition 8 (Concise). *A* $\mathcal{CVC}$ *is* concise*, if both the size of the commitment* C *and the size of the proofs* π_i *are independent of the vector size* q*.*

Informally, correctness guarantees that a CVC works as expected when its algorithms are honestly executed. A formal definition follows:

Definition 9 (Correctness). *A $\mathcal{CVC}$ is* correct *if for all $q = poly(\lambda)$, all honestly generated parameters $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{CGen}(1^\lambda, q)$ and all messages $(m_1, \ldots, m_q)$, if $(C, \mathsf{aux}) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(m_1, \ldots, m_q)$ and $\pi \leftarrow \mathsf{COpen}_{\mathsf{pp}}(i, m, \mathsf{aux})$, then the verification algorithm $\mathsf{CVer}_{\mathsf{pp}}(C, i, m, \pi)$ outputs 1 with overwhelming probability. Furthermore, correctness must hold even after some updates occur. Namely, considering the previous setting, any message m' and any index i, if $(C', U) \leftarrow \mathsf{CUpdate}_{\mathsf{pp}}(C, i, m_i, m')$ and $\pi' \leftarrow \mathsf{CProofUpdate}_{\mathsf{pp}}(C, \pi, i, U)$, then the verification algorithm $\mathsf{CVer}_{\mathsf{pp}}(C', i, m', \pi')$ must output 1 with overwhelming probability*

Security of CVCs. Finally, we discuss the security of CVCs which is defined by three properties: *indistinguishable collisions*, *position binding* and *hiding*. Informally speaking, a CVC has indistinguishable collisions if one is not able to find out if the collision finding algorithm had been used or not. Secondly, a scheme satisfies position binding if, without knowing the trapdoor, it is not possible to open a position in the commitment in two different ways. Thirdly, hiding guarantees that the commitment does not leak any information about the messages that the commitment was made to. In what follows we provide formal definitions of these properties. Since the hiding property is not needed for our application, we have deferred its definition to the full version.

Experiment $\mathsf{CollInd}^{\mathcal{CVC}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{CGen}(1^\lambda, q)$
$b \leftarrow \{0, 1\}$
$((m_1, \ldots, m_q), (i, m'_i)) \leftarrow \mathcal{A}_0(\mathsf{pp}, \mathsf{td})$
$(C_0, \mathsf{aux}^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(m_1, \ldots, m_i, \ldots, m_q)$
$\mathsf{aux}_0 \leftarrow \mathsf{CCol}_{\mathsf{pp}}(C_0, i, m_i, m'_i, \mathsf{td}, \mathsf{aux}^*)$
$(C_1, \mathsf{aux}_1) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(m_1, \ldots, m'_i, \ldots, m_q)$
$b' \leftarrow \mathcal{A}_1(C_b, \mathsf{aux}_b)$
if $b = b'$output 1
else output 0

Experiment $\mathsf{PosBdg}^{\mathcal{CVC}}_{\mathcal{A}}(\lambda)$
$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{CGen}(1^\lambda, q)$
$(C, i, m, m', \pi, \pi') \leftarrow \mathcal{A}^{\mathsf{CCol}(\cdot,\cdot,\cdot,\cdot,\mathsf{td},\cdot)}(\mathsf{pp})$
store (C, i) queried to CCol in Q
if $m \neq m' \wedge (C, i) \notin Q$
$\wedge \mathsf{CVer}_{\mathsf{pp}}(C, i, m, \pi)$
$\wedge \mathsf{CVer}_{\mathsf{pp}}(C, i, m', \pi')$
output 1
else output 0

Indistinguishable Collisions. This is the main novel property of CVCs: one can use the trapdoor to change a message in the commitment without changing the commitment itself. Intuitively, however, when seeing proofs, one should not be able to tell whether the trapdoor has been used or not. We call this notion indistinguishable collisions, and we formalize it in the game $\mathsf{CollInd}$. Observe that we require the indistinguishability to hold even when having knowledge of the trapdoor.

Definition 10 (Indistinguishable Collisions). *A $\mathcal{CVC}$ has* indistinguishable collisions *if the success probability of any stateful PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the game $\mathsf{CollInd}$ is only negligibly bigger than 1/2 in λ.*

Position-Binding. This property aims to capture that, without knowing the trapdoor, one should not be able to open the same position of a chameleon vector commitment to two different messages. In particular, we consider a strong

definition of this notion in which the adversary is allowed to use an oracle CCol for computing collisions. Namely, even when seeing collisions for some of the positions, an adversary must not find two different openings for other positions. This notion, called position-binding, is formalized as follows:

Definition 11 (Position-Binding). *A $\mathcal{CVC}$ satisfies* position-binding *if no PPT adversary $\mathcal{A}$ can output two valid proofs for different messages (m, m') at the same position i with non-negligible probability. Formally, the success probability af any PPT adversary $\mathcal{A}$ in the following game* PosBdg *should be negligible in λ. Whenever the adversary queries the collision oracle with a commitment, a position, two messages and some auxiliary information $(C, i, m, m', \mathtt{aux})$, the game runs the collision finding algorithm $\mathtt{aux}' \leftarrow \mathsf{CCol}_{\mathsf{pp}}(C, i, m, m', \mathtt{td}, \mathtt{aux})$ and returns $\mathtt{aux}'$ to the adversary.*

3.2 Construction of CVCs Based on CDH

In this section we show a direct construction of CVCs based on the Square-CDH assumption in bilinear groups, which – we note – has been shown equivalent to the standard CDH assumption [1,13]. A generic construction of CVCs can be found in the fullversion [9]. Our direct construction can be seen as an aggregated variant of the Krawczyk-Rabin chameleon hash function [8], or as a generalization of the VC scheme due to Catalano and Fiore [6]. For simplicity we describe the scheme in symmetric pairings, but we stress that this scheme can be expressed using asymmetric pairings and our implementation does use asymmetric pairings.

Construction 1. *Let $\mathbb{G}, \mathbb{G}_T$ be two groups of prime order p with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.*

$\mathsf{CGen}(1^\lambda, q)$**:** *Let $g \in \mathbb{G}$ be a random generator. Choose $z_1, \ldots, z_q \leftarrow \mathbb{Z}_p$ at random, set $h_i = g^{z_i}$ for $i = 1, \ldots, q$ and $h_{i,j} = g^{z_i z_j}$ for $i, j = 1, \ldots, q, i \neq j$. Finally, set $\mathsf{pp} = (g, \{h_i\}_{i=1,\ldots,q}, \{h_{i,j}\}_{i,j=1,\ldots,q,i\neq j})$ and $\mathtt{td} = \{z_i\}_{i=1,\ldots,q}$.*

$\mathsf{CCom}_{\mathsf{pp}}(m_1, \ldots, m_q)$**:** *Choose $r \leftarrow \mathbb{Z}_p$ at random. Set $C = h_1^{m_1} \cdots h_q^{m_q} g^r$ and $\mathtt{aux} = (m_1, \ldots, m_q, r)$.*

$\mathsf{COpen}_{\mathsf{pp}}(i, m, \mathtt{aux})$**:** *Compute $\pi = h_i^r \cdot \prod_{j=1, j\neq i}^{q} h_{i,j}^{m_j}$.*

$\mathsf{CVer}_{\mathsf{pp}}(C, i, m, \pi)$**:** *If $e(C/h_i^m, h_i) = e(\pi, g)$ output 1, else output 0.*

$\mathsf{CCol}_{\mathsf{pp}}(C, i, m, m', \mathtt{td}, \mathtt{aux})$**:** *Parse $\mathtt{aux} = (m_1, \ldots, m_q, r)$. Compute $r' = r + z_i(m - m')$ and set $\mathtt{aux}' = (m_1, \ldots, m', \ldots, m_q, r')$.*

$\mathsf{CUpdate}_{\mathsf{pp}}(C, i, m, m')$**:** *Compute $C' = C \cdot h_i^{m'-m}$, set $U = (i, u) = (i, m' - m)$.*

$\mathsf{CProofUpdate}_{\mathsf{pp}}(C, \pi_j, j, U)$**:** *Parse $U = (i, u)$. Compute $C' = C \cdot h_i^u$. If $i \neq j$ compute $\pi'_j = \pi_j \cdot h_{j,i}^u$, else $\pi'_j = \pi_j$.*

We also show two additional algorithms that allow to "accumulate" several updates at different positions, and then to apply these updates to a proof in constant time.

$\mathsf{accumulateUpdate}_{\mathsf{pp}}(AU, U)$**:** Parse $AU = (j, au)$ and $U = (i, u)$. If $j \neq i$, compute $au = au \cdot h_{i,j}^{u}$.
$\mathsf{CProofUpdate}'_{\mathsf{pp}}(\pi_j, AU)$**:** Parse $AU = (j, au)$. Compute $\pi'_j = \pi_j \cdot au$.

Theorem 1. *If the CDH assumption holds, Construction 1 is a chameleon vector commitment.*

Please refer to the fullversion [9] for the proofs.

4 VDS from CVCs

In this section we present our VDS protocol from CVCs. The section is structured as follows: in Sect. 4.1 we explain the main idea of the construction, the formal description is then given in Sect. 4.2.

4.1 Intuition

The main idea is to build a q-ary tree where each node of the tree authenticates a data element and its q children. In our construction, every node is a CVC to a vector of size $q + 1$. The first component of this vector is the data element, and the q remaining components are the node's q children (or 0 as a dummy value for children that do not yet exist). The root of the tree is used as the public verification key, while the CVC trapdoor is the private key which enables the client to append further elements to the tree.

The tree is constructed in such a way that it grows dynamically from top to bottom. It works as follows: Initially, the tree consists only of the root node, which is a CVC to a vector of zeros. Elements are inserted into the tree from left to right and new children are linked to their parent by computing an opening for the appropriate position in the parent node.

Consider for example the tree shown in Fig. 1 where $q = 2$. Blank spaces denote the fact, that no opening for this position has been computed yet (i.e. the corresponding child node does not exist yet). On the left side, the structure is depicted for a dataset that contains three elements (d_1, d_2, and d_3). To add the next element d_4, the client generates a new CVC to a vector of three elements, where the first component is the new element d_4 and all the other components are set to 0. The new node n_4 will be the second child of node n_1, thus the client computes an opening for the third component of n_1 to the new CVC n_4. It can compute such an opening by finding a collision. The resulting tree is shown on the right.

For simplicity, the root node will not authenticate any data element, but only its children.

The proof that a data element d is stored at a certain position in the tree (and thereby in a certain position in the dataset) consists of a list of CVC-openings and intermediate nodes of the tree:

$$\tilde{\pi}_i = (\pi_l, n_l, \ldots, n_1, \pi_0).$$

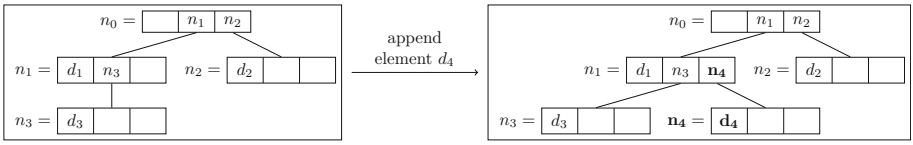


Fig. 1. Appending element d_4 to our authenticated datastructure for $q = 2$

Here π_l is an opening of the first component of n_l to the element d, π_{l-1} is an opening showing that the node n_l is a child of node n_{l-1}, and eventually π_0 is an opening showing that node n_1 is a child of the root node. The whole proof $\tilde{\pi}_i$ is called an *authentication path*.

To reduce the amount of data the client has to store, we exploit the key-features of our CVCs. Adding a new node to the tree requires knowledge of the parent of the new node, i.e., the value and the corresponding auxiliary information, but storing the entire tree at the client would defeat the purpose of a VDS protocol. To circumvent this problem, we make nodes recomputable. Namely, to create a new node n_i for element d_i, we first derive the randomness r_i of the CVC deterministically using a pseudorandom function, then compute the node $(n_i, \mathtt{aux}_i^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, 0, \ldots, 0; r_i)$ and find a collision $\mathtt{aux}_i \leftarrow \mathsf{CCol}_{\mathsf{pp}}(n_i, 1, 0, d, \mathtt{td}, \mathtt{aux}_i^*)$.

This idea allows us to split the tree between the client and the server in the following way: The client only stores the secret key $\mathsf{sp} = (k, \mathtt{td}, \mathtt{cnt})$, whereas the server simply stores all data $d_1, d_2, \ldots$ and all nodes $n_1, n_2, \ldots$ along with their openings. Now, whenever the client wants to add a new data element d, it determines the next free index $i = \mathtt{cnt}+1$ and the level l of the new node, as well as the index p of its parent and the position j this node will have in its parent. The client then computes the new node as $(n_i, \mathtt{aux}_i^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, 0, \ldots, 0; r_i)$, where the randomness r_i is computed using the PRF. It adds the new data element by finding a collision for the first component $\mathtt{aux}_i \leftarrow \mathsf{CCol}_{\mathsf{pp}}(n_i, 1, 0, d, \mathtt{td}, \mathtt{aux}_i^*)$. With this, the client then can compute an opening of the first component of n_i, showing that d is indeed stored in n_i in the first component: $\pi_l \leftarrow \mathsf{COpen}_{\mathsf{pp}}(1, d, \mathtt{aux}_i)$. To append this new node n_i in the tree, the client recomputes the parent node as $(n_p, \mathtt{aux}_p^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, 0, \ldots, 0; r_p)$ and finds a collision in the j-th position: $\mathtt{aux}_p \leftarrow \mathsf{CCol}_{\mathsf{pp}}(n_p, j, 0, n_i, \mathtt{td}, \mathtt{aux}_p^*)$. From there, the client can then compute an opening of the parent node to the new node: $\pi_{l-1} \leftarrow \mathsf{COpen}_{\mathsf{pp}}(j, n_i, \mathtt{aux}_p)$. We call (π_l, n_i, π_{l-1}) an *insertion path* (since it is a partial authentication path). The client then sends the new data element and the insertion path $(d, (\pi_l, n_i, \pi_{l-1}))$ to the server, who stores this tuple in its database.

To answer a query on index i, the server computes a full authentication path for i by concatenating the insertion paths of all nodes on the path from i to the root. Note that each insertion path is of constant size due to the conciseness requirement for CVCs (see Definition 8), and thus the size of a full authentication path is only in $\mathcal{O}(\log_q N)$, where N is the number of elements in the dataset so far.

At first glance, the idea described so far seems to work. However, there is one more issue that needs to be overcome: The client may perform updates (i.e., change a data element from d_i to d_i') and continue appending new elements afterwards. This changes the value of n_i, and the client cannot recompute this value without storing all update-information. However, when appending a new child of n_i, the client has to find an opening of a position of this node n_i.

We solve this issue by letting the server store "an aggregate" of all update information. This way, the client can compute an "outdated" opening (i.e., as before any update) and then this opening can be updated by the server in constant time. Precisely, in the general case the server has to store a list of all updates and apply them one by one on incoming openings (in linear time). However, in the case of our CDH-based CVCs, we can exploit their homomorphic property to "accumulate" update-information by only storing its sum, thus achieving *constant* insertion time.

4.2 Formal Description of Our Construction

In this section, we present the formal description of our construction.

For better readability, we define the following three functions. The first one computes the index of the parent of node i, the second one computes in which component node i is stored in its parent, and the third function computes in which level of the tree node i can be found.

$$\mathsf{parent}(i) = \lfloor \tfrac{i-1}{q} \rfloor$$
$$\#\mathsf{child}(i) = ((i-1) \mod q) + 2$$
$$\mathsf{level}(i) = \lceil \log_q((q-1)(i+1)+1) - 1 \rceil$$

Construction 2. *Let* $\mathcal{CVC} = (\mathsf{CGen}, \mathsf{CCom}, \mathsf{COpen}, \mathsf{CVer}, \mathsf{CCol}, \mathsf{CUpdate}, \mathsf{CProofUpdate})$ *be a CVC. Define* $\mathcal{VDS} = (\mathsf{Setup}, \mathsf{Append}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ *as follows:*

$\mathsf{Setup}(1^\lambda, \mathsf{q})$. *This algorithm picks a random PRF key* $k \leftarrow \{0,1\}^\lambda$*, computes a key-pair for the chameleon vector commitment* $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{CGen}(1^\lambda, q+1)$*, and sets the counter* $\mathtt{cnt} := 0$*. It computes* $r_0 \leftarrow \mathsf{f}(k, 0)$*, sets the root as* $(\rho, \mathsf{aux}_\rho) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, \ldots, 0; r_0)$*, the secret key* $sk := (k, \mathsf{td}, \mathtt{cnt})$*, and the public key* $pk := (\mathsf{pp}, \rho)$*. Finally, the secret key* sk *is kept by the client while the public key* pk *is given to the server.*

Before proceeding with the remaining algorithms, we summarize the information stored by the server. The server maintains a database DB consisting of tuples $(i, d_i, n_i, \pi_i, \pi_{p,j}, AU_i, \{AU_{i,j}\}_{j=1}^{q+1})$ *where:* $i \geq 0$ *is an integer representing the index of every DB element,* d_i *is DB value at index* i*,* n_i *is a CVC commitment,* π_i *is a CVC proof that* d_i *is the first committed message in* n_i*,* $\pi_{p,j}$ *is a CVC proof that* n_i *is the message at position* $j+1$ *committed in* n_p *(which is the CVC of* n_i*'s parent node),* AU_i *is the accumulated update information that can be used to update the proof* π_i*, and* $AU_{i,j}$ *are the accumulated update informations that can be used to update the children's proofs* $\pi_{i,j}$*.*

$\mathsf{Append}(sk, d)$. *The algorithm parses* $sk = (k, \mathtt{td}, \mathtt{cnt})$ *and determines the index* $i = \mathtt{cnt} + 1$ *of the new element, the index* $p = \mathsf{parent}(i)$ *of its parent node, the position* $j = \#\mathsf{child}(i)$ *that this element will have in its parent, and then increases the counter* $\mathtt{cnt}' = \mathtt{cnt} + 1$. *Next, it computes the new node as* $(n_i, \mathtt{aux}_i^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, 0, \ldots, 0; r_i)$ *where* $r_i \leftarrow \mathsf{f}(k, i)$ *and inserts the data* d *by finding a collision* $\mathtt{aux}_i \leftarrow \mathsf{CCol}_{\mathsf{pp}}(n_i, 1, 0, d, \mathtt{td}, \mathtt{aux}_i^*)$. *To append the node* n_i *to the tree, the algorithm recomputes the parent node as* $(n_p, \mathtt{aux}_p^*) \leftarrow \mathsf{CCom}_{\mathsf{pp}}(0, 0, \ldots, 0; r_p)$ *and inserts* n_i *as the* j*-th child of* n_p *by finding a collision in the parent node at position* j, *i.e., the client runs* $\mathtt{aux}_p \leftarrow \mathsf{CCol}_{\mathsf{pp}}(n_p, j, 0, n_i, \mathtt{td}, \mathtt{aux}_p^*)$. *It then computes* $\pi_{i,1} \leftarrow \mathsf{COpen}_{\mathsf{pp}}(1, d, \mathtt{aux}_i)$ *and* $\pi_{p,j} \leftarrow \mathsf{COpen}_{\mathsf{pp}}(j, n_i, \mathtt{aux}_p)$, *and sets the insertion path* $(\pi_{i,1}, n_i, \pi_{p,j})$.

The client $\mathcal{C}$ *sends the above insertion path and the new element* d *to the server* $\mathcal{S}$. $\mathcal{S}$ *then applies the accumulated update* $AU_{p,j}$ *to* $\pi_{p,j}$ *(i.e., compute* $\pi'_{p,j} \leftarrow \mathsf{CProofUpdate}'_{\mathsf{pp}}(\pi_{p,j}, AU_{p,j})$*) and stores these items in its database* DB.

$\mathsf{Query}(pk, DB, i)$. *In the query protocol, the client sends* i *to the server, who determines the level* $l = \mathsf{level}(i)$ *and constructs an authentication path:*

$\tilde{\pi}_i \leftarrow (\pi_{i,1})$
$a \leftarrow i$
$b \leftarrow \mathsf{parent}(i)$
for $h = l - 1, \ldots, 0$
  $c \leftarrow \#\mathsf{child}(a)$
  $\tilde{\pi}_i \leftarrow \tilde{\pi}_i :: (n_a, \pi_{b,c})$
  $a \leftarrow b$
  $b \leftarrow \mathsf{parent}(b)$

Finally, the server returns $\tilde{\pi}_i$ *to the client.*

$\mathsf{Verify}(pk, i, d, \tilde{\pi}_i)$. *This algorithm parses* $pk = (\mathsf{pp}, \rho)$ *and* $\tilde{\pi}_i = (\pi_l, n_l, \ldots, n_1, \pi_0)$. *It then proceeds by verifying all proofs in the authentication path:*

$v \leftarrow \mathsf{CVer}_{\mathsf{pp}}(n_i, 1, d, \pi_l) \wedge n_i \neq 0$
$a \leftarrow i$
$b \leftarrow \mathsf{parent}(i)$
for $h = l - 1, \ldots, 0$
  $c \leftarrow \#\mathsf{child}(a)$
  $v \leftarrow v \wedge \mathsf{CVer}_{\mathsf{pp}}(n_b, c, n_a, \pi_h) \wedge n_b \neq 0$
  $a \leftarrow b$
  $b \leftarrow \mathsf{parent}(b)$

If $v = 1$ *then output* d. *Otherwise output* $\perp$.

$\mathsf{Update}(pk, DB, sk, i, d')$. *In the update protocol, the client, given the secret key* sk, *sends an index* i *and a value* d' *to the server. The server answers by sending the value* d *currently stored at position* i *and the corresponding authentication path* $\tilde{\pi}_i = (\pi_l, n_l, \ldots, n_1, \pi_0)$ *(this is generated as in the* Query *algorithm). The client then checks the correctness of* $\tilde{\pi}_i$ *by running*

$\mathsf{Verify}(pk, i, d, \tilde{\pi}_i)$. *If the verification fails, the client stops running. Otherwise, it continues as follows. First, it parses* sk *as* $(k, \mathtt{td}, \mathtt{cnt})$, *it determines the level of the updated node* $l \leftarrow \mathsf{level}(i)$, *and computes the new root* $\rho' = n'_0$ *as follows:*

$(n'_l, U_l) \leftarrow \mathsf{CUpdate}_{\mathsf{pp}}(n_i, 1, d, d', \pi_l)$
$a \leftarrow i$
$b \leftarrow \mathsf{parent}(i)$
for $h = l - 1, \ldots, 0$
  $c \leftarrow \#\mathsf{child}(a)$
  $(n'_h, U_h) \leftarrow \mathsf{CUpdate}_{\mathsf{pp}}(n_h, c, n_{h+1}, n'_{h+1}, \pi_h)$
  $a \leftarrow b$
  $b \leftarrow \mathsf{parent}(b)$

On the other side, after receiving (i, d'), *the server runs a similar algorithm to update all stored elements and proofs along the path of the new node. It also accumulates the new update information for every node in this path:*

$(n'_i, U_i) \leftarrow \mathsf{CUpdate}_{\mathsf{pp}}(n_i, 1, d, d')$
for $j = 1, \ldots, q + 1$
  $AU_{i,j} \leftarrow \mathsf{accumulateUpdate}(AU_{i,j}, U_i)$
  $\pi'_{i,j} \leftarrow \mathsf{CProofUpdate}'_{\mathsf{pp}}(\pi_{i,j}, AU_{i,j})$
$(\cdot, \pi'_i) \leftarrow \mathsf{CProofUpdate}_{\mathsf{pp}}(n_i, \pi_i, i, U_i)$
$a \leftarrow i$
$b \leftarrow \mathsf{parent}(a)$
for $h = l - 1, \ldots, 0$
  $c \leftarrow ((a - 1) \mod q) + 2$
  $(n'_b, U_b) \leftarrow \mathsf{CUpdate}_{\mathsf{pp}}(n_b, c, n_a, n'_a)$
  for $j = 1, \ldots, q + 1$
    $AU_{b,j} \leftarrow \mathsf{accumulateUpdate}(AU_{b,j}, U_b)$
    $\pi'_{b,j} \leftarrow \mathsf{CProofUpdate}'_{\mathsf{pp}}(\pi_{b,j}, U_b)$
  $a \leftarrow b$
  $b \leftarrow \mathsf{parent}(b)$

In the above algorithms, a *is the index of the changed node,* b *the index of its parent node, and* c *the position of node* a *in* b. *Basically, the algorithms change the value of node* i, *and then this change propagates up to the root* $(\rho' = n'_0)$.

Finally, both the client and the server compute the new public key as $pk = (\mathtt{pp}, \rho')$.

4.3 Security

In this section, we show that the VDS protocol described in the previous section is secure.

Theorem 2. *If* f *is a pseudorandom function and* $\mathcal{CVC}$ *is a secure CVC, then Construction 2 is a secure VDS.*

Proof. We prove the theorem by first defining a hybrid game in which we replace the PRF with a random function. Such hybrid is computationally indistinguishable from the real VDSsec security game by assuming that f is pseudorandom. Then, we proceed to show that any efficient adversary cannot win with non-negligible probability in the hybrid experiment by assuming that the CVC scheme is secure. In what follows we use $Gm_{i,\mathcal{A}}(\lambda)$ to denote the experiment defined by Game i run with adversary $\mathcal{A}$.

Game 0: this identical to the experiment VDSsec.

Game 1: this is the same as Game 0 except that the PRF f is replaced with a random function (via lazy sampling). It is straightforward to see that this game is negligibly close to Game 0 under the pseudo randomness of f, i.e., $\text{Prob}[Gm_{0,\mathcal{A}}(\lambda) = 1] - \text{Prob}[Gm_{1,\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

Now, consider Game 1. Let $(i^*, d^*, \hat{\pi})$ be the tuple returned by the adversary at the end of the game, d be the value currently stored in the database at index i^*. Recall that Game 1 outputs 1 if $\mathsf{Verify}(pk, i^*, d^*, \hat{\pi}) = 1$ and $d \neq d^*$. Consider a honestly computed authentication path $\tilde{\pi}$ for (i^*, d) (this is the path which can be computed by the challenger), and observe that by construction the sequence in $\hat{\pi}$ ends up at the public root. Intuitively, this means that $\hat{\pi}$ and $\tilde{\pi}$ must deviate at some point in the path from i^* up to the root. We define dcol as the event that the two authentication paths deviate exactly in i^*, i.e., that $n_i = n_i^*$. Dually, if dcol does not occur, it intuitively means that the adversary managed to return a valid authentication path that deviates from a honestly computed one in some internal node. Clearly, we have:

$$\begin{aligned}\text{Prob}[Gm_{1,\mathcal{A}}(\lambda) = 1] = \text{Prob}[Gm_{1,\mathcal{A}}(\lambda) = 1 \wedge \mathsf{dcol}] \\ + \text{Prob}\left[Gm_{1,\mathcal{A}}(\lambda) = 1 \wedge \overline{\mathsf{dcol}}\right]\end{aligned}$$

Our proof proceeds by showing that both

$$\text{Prob}[Gm_{1,\mathcal{A}}(\lambda) = 1 \wedge \mathsf{dcol}]$$

and

$$\text{Prob}\left[Gm_{1,\mathcal{A}}(\lambda) = 1 \wedge \overline{\mathsf{dcol}}\right]$$

are negligible under the assumption that the CVC is position-binding.

Case dcol. In this case we build a reduction $\mathcal{B}$ against the position-binding property of the underlying CVC.

On input pp, the reduction $\mathcal{B}$ computes the root node as described in the catGen algorithm, sets the counter $\mathtt{cnt} := 0$, and sets $pk \leftarrow (\mathsf{pp}, \rho)$. It then runs $\mathcal{A}(\mathsf{vp})$ by simulating the VDSsec game.

Whenever the adversary $\mathcal{A}$ streams some data element d, the reduction proceeds as described in the catAdd algorithm (except that pseudorandom values are now sampled randomly, as per Game 1). However, $\mathcal{B}$ does not know the full secret key sk – it does not know the CVC trapdoor – but it can use its collision-oracle to compute the necessary collisions in the CVC in order to add new nodes to the tree. So, the reduction $\mathcal{B}$ returns $(i, \tilde{\pi}_i)$ to the adversary and stores the tuple $(d, i, \tilde{\pi}_i)$ in a list L.

Whenever the adversary $\mathcal{A}$ wants to update the element at position i to the new value d', the reduction proceeds as described in the Update algorithm. Note that the CVC trapdoor is not needed in this phase. $\mathcal{B}$ then returns the updated proof $\tilde{\pi}'_i$ to $\mathcal{A}$ and updates the tuple $(d', i, \tilde{\pi}'_i)$ in L.

Eventually the adversary outputs $(d^*, i^*, \hat{\pi}^*)$. The reduction then finds the actual data d and the corresponding proof $\tilde{\pi}_i$ for position i by searching for the tuple $(d, i^*, \tilde{\pi}_i)$ in L, parses $\hat{\pi}^* = (\pi^*, n_i^*, \dots)$ and $\tilde{\pi}_i = (\pi, n_i, \dots)$ and outputs $(n_i, 1, d, d^*, \pi, \pi^*)$.

For the analysis now observe that $\mathcal{B}$ is efficient as so is $\mathcal{A}$, and searching for a tuple in an ordered list can be done in polynomial time. It is easy to see that $\mathcal{B}$ perfectly simulates the view for $\mathcal{A}$ as in the game VDSsec. Now, whenever dcol happens, we know that $n_i^* = n_i$. Hence both (π, d) and (π^*, d^*) must verify correctly whenever $\mathcal{A}$ wins. Furthermore, observe that $\mathcal{B}$ never uses its collision-oracle on position 1, since the Append algorithm always uses CCol on index $j > 1$. This means essentially means that

$$\begin{aligned}\mathrm{Prob}[\mathit{Gm}_{1,\mathcal{A}}(\lambda) = 1 \wedge \mathsf{dcol}] &\leq \mathrm{Prob}[\mathsf{PosBdg}_{\mathcal{B}}(\lambda, q) = 1] \\ &= \mathsf{negl}(\lambda)\end{aligned}$$

Case $\overline{\mathit{dcol}}$ Recall that in this case Game 1 outputs 1 only if the adversary wins by returning an authentication path which deviates from the correct one at some internal node in the path from i^* up to the root. In this case too, we build a reduction $\mathcal{B}$ against the position-binding property of the underlying CVC.

On input pp, the reduction $\mathcal{B}$. It then tries to set an upper limit on the number of elements the adversary will authenticate in the tree by choosing its depth $l = \lambda$. The reduction then builds a tree of CVCs of size l from bottom to top, where in each CVC every position which does not point to a child (especially the first position) is set to 0. Denote the root of this tree by ρ. Finally, the reduction $\mathcal{B}$ sets $\mathtt{cnt} := 0$, sets $pk \leftarrow (\mathtt{pp}, \rho)$ and runs $\mathcal{A}(pk)$ by simulating Game 1 to it.

Whenever the adversary $\mathcal{A}$ streams some data element d to the reduction, the reduction determines the index $i = \mathtt{cnt} + 1$ for the new data element, increases cnt by one and inserts the new element into the tree by finding a collision in the first component of node n_i using its collision-oracle. It then computes an authentication path $\tilde{\pi}_i$ for d as described in the Append algorithm. The reduction returns $(i, \tilde{\pi}_i)$ to the adversary and stores $(d, i, \tilde{\pi}_i)$ in some list L. If the adversary exceeds the number of elements l, the reduction stops the adversary $\mathcal{A}$, increases $l \leftarrow l \cdot \lambda$, and starts again.

Whenever the adversary $\mathcal{A}$ wants to update the element at position i to some new value d' the reduction proceeds as described in the Update algorithm. It then

returns the updated proof $\tilde{\pi}'_i$ to the adversary and updates the tuple $(d', i, \tilde{\pi}'_i)$ in L.

At the end of the game the adversary outputs $(d^*, i^*, \hat{\pi}^*)$. The reduction parses $\hat{\pi}^* = (\pi_0^*, n_0^*, \pi_1^*, \dots)$ and finds the largest j for which $n_i^* = n_j$, i.e., for which the authentication path $\hat{\pi}^*$ still agrees with the actual tree. Also $\mathcal{B}$ finds the authentication path $\tilde{\pi}_{i^*} = (\pi_0^{i^*}, n_0^{i^*}, \pi_1^{i^*}, \dots)$ up to i^*, if i^* was streamed by the adversary, or otherwise up to the deepest ancestor of i^*. Clearly, π_i^* then must be a proof that n_{i-1}^* is "stored" in n_i^* at some position h.

If n_j is the deepest node in the path towards i^* that is stored by the challenger then the h-th message committed in n_j by the challenger is 0. In this case $\mathcal{B}$ can produce a honest proof $\pi_{h,0}$ that 0 is the h-th message committed in n_j, and then outputs $(n_j, h, 0, n_{i-1}^*, \pi_{h,0}, \pi_i^*)$.

Otherwise, if n_j is not the deepest node, the honest path $\tilde{\pi}_{i^*}$ must contain a node $n_k^{i^*} = n_j$ as well as a proof $\pi_k^{i^*}$ that the node $n_{k-1}^{i^*}$ is the h-th child of n_j. In this case $\mathcal{B}$ outputs $(n_j, h, n_{k-1}^{i^*}, n_{i-1}^*, \pi_k^{i^*}, \pi_i^*)$. As one can check, this case also captures the on in which $h = 1$ and $n_{k-1}^{i^*} = d \neq d^* = n_{i-1}^*$.

For the analysis see that $\mathcal{B}$ is efficient because $\mathcal{A}$ is and because the limit l will be large enough after a polynomial number of times. Now observe that $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in Game 1 (otherwise the underlying CVC would not have indistinguishable collisions). It is easy to see that whenever $\mathcal{A}$ wins the pair (π_i^*, n_{i-1}^*) verifies w.r.t. n_j. As $\mathcal{B}$ honestly computed $n_{k-1}^{i^*}$ and $\pi_k^{i^*}$, this pair will also verify w.r.t. n_j. Note that $\mathcal{B}$ only asks for collisions at position 1, but $h > 1$. Therefore $\mathcal{B}$ wins whenever $\mathcal{A}$ does. Since the underlying CVC is position-binding, this probability is at most negligible.

$$\begin{aligned} \mathrm{Prob}\left[Gm_{1,\mathcal{A}}(\lambda) = 1 \wedge \overline{\mathsf{dcol}} \right] &\leq \mathrm{Prob}\left[\mathsf{PosBdg}_{\mathcal{B}}(\lambda, q) = 1 \right] \\ &= \mathsf{negl}(\lambda) \end{aligned}$$

Since both parts are at most negligible in λ, the overall success probability of any efficient adversary can only be negligible in λ, hence Construction 2 is secure according to Definition 6.

4.4 Reducing the Public Key Size

Instantiating the scheme from Construction 2 with the CVCs from Construction 1 results in a public key of size $\mathcal{O}(q^2)$ due to the values $h_{i,j}$ that are stored in the public parameters pp, which are part of the VDS public key. As an interesting extension, we show how to reduce the public key size of our VDS protocol to $\mathcal{O}(1)$ by carefully inspecting our usage of the CVC scheme from Construction 1. In particular, note that the public key of the VDS protocol consists of: elements that are required for public verifiability of queried data elements, and elements only needed by the server.

The elements $h_{i,j}$ are only needed for COpen in the Append protocol on the client-side, and in $\mathsf{CProofUpdate}$ in the Update protocol on the server-side. They are, however, not used for public verifiability in the VDS protocol. Therefore,

these values $h_{i,j}$ do not need to be part of the public verification key of the VDS protocol, and can be sent directly to the server only once. This already reduces the key size from $\mathcal{O}(q^2)$ to $\mathcal{O}(q)$.

To get rid of the remaining elements $h_1, \ldots, h_q$ in the public key, we observe that during each run of Verify, only one of these h_i values is needed at every level. This allows us to do the following change to the protocol: Instead of storing all $h_1, \ldots, h_q$ in the public key, we let the client sign each h_i, together with its index i, using a regular signature scheme. For every i one thus obtain a signature σ_i, and the server is required to store all pairs $(h_1, \sigma_1), \ldots, (h_q, \sigma_q)$. Later when the server has to provide the answer to a query, we require it to include all the necessary (h_i, σ_i) pairs in the authentication path. Note that these pairs are most q, and thus do not increase the asymptotic length of the authentication path. The verifier will eventually verify that σ_i is a valid signature on $h_i||i$, before using h_i in CVer. This change allows us to replace the q h_i values from the public key with a *single* verification key of a signature scheme, and it results in a VDS protocol with a public key size of size $\mathcal{O}(1)$.

5 VDS from Accumulators

Our second construction is conceptually very different from all previous VDS constructions, since it does not rely on any tree structure. The basic idea is to let the client sign each element of the dataset with a regular signature scheme, and to use a cryptographic accumulator to revoke old signatures once an element gets updated (otherwise the server could perform a rollback attack). For this, the client has a key-pair of a signature scheme, and his public key serves as the public verification key. Whenever an element is queried from the server, the server return the element, its signature, and a proof-of-non-membership, to show that this signature has not been revoked yet.

However, this idea does not work immediately. One reason is that some accumulators only support the accumulation of an a priori fixed number of elements such as [15]. Another issue is that the size of the public-key is typically linear in the number of accumulated values, and thus the client might not be able to store it.

In our construction we solve this issue by exploiting the specific algebraic properties of the bilinear-map accumulator as follows: The bilinear-map accumulator [15] and its extension to support non-membership proofs [7] use a private key s and a public key $g, g^s, \ldots, g^{s^q}$, where q is an upper bound on the number of elements in the accumulator. To compute a proof of (non-)membership for q accumulated elements, all the values $g, g^s, \ldots, g^{s^q}$ are needed. However, to *verify* a proof of (non-)membership only the values g and g^s is required. In our VDS scheme, we only require that proofs of non-membership can be verified publicly; they do not need to be publicly computable with only the public key and the accumulator value. Therefore we only put g and g^s into our public key. Only the client has to add additional items to the accumulator, but he can do so using his private key s and g to recompute $g^s, \ldots, g^{s^q}$. Furthermore, only the server

has to compute proofs of non-membership and thus needs to know $g, g^s, \ldots, g^{s^q}$, where q is the number of updates so far. Since we do not want to fix an upper bound on the number of updates a priori, the client extends the list of known values of the server with each update-operation. E.g. in the update protocol for the third element the client sends the value g^{s^3} to the server who then appends this value to a list.

This allows us to reduce the size of the public-key to $\mathcal{O}(1)$, and to support an unbounded number of updates.

5.1 Our Scheme

In this section we show how to use the bilinear-map accumulator described in Sect. 2.5 in combination with a signature scheme to build a VDS protocol.

Construction 3. *Let* $\mathsf{Sig} = (\mathsf{SKg}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a signature scheme and* $H : \{0,1\}^* \mapsto \mathbb{Z}_p^*$ *be a hash function. The VDS protocol* $\mathcal{VDS} = (\mathsf{Setup}, \mathsf{Append}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ *is defined as follows:*

$\mathsf{Setup}(1^\lambda)$. *The setup algorithm first generates a tuple of bilinear map parameters* $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. *Second, it chooses* s *at random from* $\mathbb{Z}_p^*$ *and computes* g^s. *Next, it generates a key-pair* $(ssk, vk) \leftarrow \mathsf{SKg}(1^\lambda)$, *initializes two counters* $\mathtt{cnt} := 0$ *and* $\mathsf{upd} := 0$, *and sets* $S_{last} = g$. *It also generates an initially empty accumulator* $f'(\mathcal{E}) := g$. *Note that during the lifetime of the scheme, the server will increasingly store* $g, g^s, \ldots, g^{s^{\mathsf{upd}}}$ *while the client only stores* $S_{last} = g^{s^{\mathsf{upd}}}$. *Finally, the algorithm outputs the secret key* $sk = (g, p, s, ssk, S_{last}, \mathtt{cnt}, \mathsf{upd})$ *which is kept by the client, and the public key* $pk = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^s, vk, f'(\mathcal{E}))$ *which is given to the server.*

$\mathsf{Append}(sk, d)$. *The append algorithm increases the counter* $\mathtt{cnt} := \mathtt{cnt} + 1$, *chooses a random tag* $\mathsf{tag} \leftarrow \{0,1\}^\lambda$, *and signs the value* $m := d \| \mathsf{tag} \| \mathtt{cnt}$ *by computing* $\sigma \leftarrow \mathsf{Sign}(ssk, m)$. *The pair* $((d, \mathsf{tag}, \mathtt{cnt}), \sigma)$ *is finally sent to* $\mathcal{S}$, *who stores it at position* $\mathtt{cnt}$ *in* DB.

$\mathsf{Query}(pk, DB, i)$. *To retrieve the* i*-th element from* DB, *the client sends* i *to* $\mathcal{S}$, *who computes the response as follows:* $\mathcal{S}$ *retrieves the pair* (m, σ) *from* DB *and computes a proof of non-membership* (w, u) *for the element* $e_i \leftarrow H(\sigma)$ *as described in Sect. 2.5. Finally,* $\mathcal{S}$ *returns* $(d, \pi) = ((d', \mathsf{tag}, i), (\sigma, w, u))$.

$\mathsf{Verify}(pk, i, d, \pi)$. *This algorithm parses* $pk = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^s, vk, f'(\mathcal{E}))$, $d = (d', \mathsf{tag}, i)$, *and* $\pi = (\sigma, w, u)$. *It sets* $e_i \leftarrow H(\sigma)$, *and outputs* d' *iff* $\mathsf{Vrfy}(vk, d' || \mathsf{tag} || i, \sigma) = 1$ *and* $e(w, g^{e_i} \cdot g^s) = e(f'(\mathcal{E}) \cdot g^u, g)$.

$\mathsf{Update}(sk, DB, sk, i, d')$. *The clients runs* $(d, \pi) \leftarrow \mathsf{Query}(pk, DB, i)$ *to retrieve the* i*-th entry from* DB. *Afterwards,* $\mathcal{C}$ *verifies the correctness of the entry by running the verification algorithm* $\mathsf{Verify}(pk, i, d, \pi)$. *If* Verify *outputs* 1, *then* $\mathcal{C}$ *increases the counter* $\mathsf{upd} := \mathsf{upd} + 1$, *signs the new element* $\sigma' \leftarrow \mathsf{Sign}(ssk, d' || \mathsf{tag} || i)$ *using a random tag* $\mathsf{tag} \leftarrow \{0,1\}^\lambda$, *and adds the old signature to the accumulator as follows. The client computes* $g^{s^{\mathsf{upd}}} = S_{last} := (S_{last})^s$, $e_i \leftarrow H(\sigma)$, *and* $f''(\mathcal{E}) := f'(\mathcal{E})^{e_i + s}$. *The client sends* $(g^{s^{\mathsf{upd}}}, f''(\mathcal{E}), \sigma')$ *to* $\mathcal{S}$. *The server stores* (σ', d') *at position*

i in DB, and $g^{s^{\text{upd}}}$ in its parameters. The public key is then updated to $pk' := (p, \mathbb{G}, \mathbb{G}_T, e, g, g^s, vk, f''(\mathcal{E}))$.

Theorem 3. *If* Sig *is a strongly unforgeable signature scheme, H a collision-resistant hash function, and* ACC *the collision-resistant accumulator as defined in Sect. 2.5, then Construction 3 is a secure VDS protocol.*

Proof. Let $\mathcal{A}$ be an efficient adversary against the security of Construction 3 as defined in game VDSsec, denote by $pk^* := (p, \mathbb{G}, \mathbb{G}_T, e_i, g, g^s, vk, f^*(\mathcal{E}))$ the public-key hold by the challenger at the end of the game, and let

$$Q := ((d_1||\mathtt{tag}_1||1, \sigma_1), (d_2||\mathtt{tag}_2||2, \sigma_2), \ldots, (d_n||\mathtt{tag}_n||n, \sigma_n))$$

be the state of the database DB at the end of the game. By $(w_1, u_1), \ldots, (w_n, u_n)$ we denote the witnesses of the non-membership proofs that be can be computed by the challenger using public values only, as discussed in Sect. 2.5, and denote by $Q' := ((d'_1||\mathtt{tag}'_1||i'_1, \sigma'_1), (d'_2||\mathtt{tag}'_2||i'_2, \sigma'_2), \ldots, (d'_n||\mathtt{tag}'_n||i_n, \sigma'_n))$ all queries sent by $\mathcal{A}$. Clearly, $Q \subseteq Q'$ and let $O := Q' \backslash Q$ be the set of queries that have been sent by the adversary and which are not in the current database. Let $\mathcal{A}$ be an efficient adversary that outputs $((d^*, \mathtt{tag}^*, i^*), (\sigma^*, w^*, u^*))$ such that $(\hat{d}, i^*) \notin Q$, $\mathsf{Vrfy}(vk, d^*||\mathtt{tag}^*||i^*, \sigma) = 1$, and $e(w^*, g^{e_i^*} \cdot g^s) = e(f'(\mathcal{E}) \cdot g^{u^*}, g)$, with $e_i^* := H(\sigma^*)$. Then, we define the following events:

- hcol is the event that there exists an index $1 \leq i \leq n$ such that $H(\sigma^*) = H(\sigma'_i)$ and $\sigma^* \neq \sigma'_i$.
- fake is the event that $e(w^*, g^{e_i^*} \cdot g^s) = e(f'(\mathcal{E}) \cdot g^{u^*}, g)$ and $e_i \in \mathcal{E}$.

Observe that the case where the adversary finds a collision in the accumulator and the one where he finds a fake witness for a membership of a non-member of $\mathcal{E}$, do not help to break the security of the VDS scheme. Given these events, we can bound $\mathcal{A}$'s success probability as follows:

$$\begin{aligned}\text{Prob}\left[\mathsf{VDSsec}_{\mathcal{A}}^{\mathcal{VDS}}(\lambda) = 1\right] \leq \text{Prob}[\mathsf{hcol}] + \text{Prob}[\mathsf{fake}] + \\ \text{Prob}\left[\mathsf{VDSsec}_{\mathcal{A}}^{\mathcal{VDS}}(\lambda) = 1 \ \wedge \ \overline{\mathsf{hcol}} \ \wedge \ \overline{\mathsf{fake}}\right]\end{aligned}$$

In the following we show that the parts of the sum are negligible. The fact that Prob[hcol] is negligible, follows trivially by the collision-resistance of the hash function. Furthermore, it is also easy to see that Prob[fake] is negligible as well due to proof of non-membership properties of the accumulator.

Claim. $\text{Prob}\left[\mathsf{VDSsec}_{\mathcal{A}}^{\mathcal{VDS}}(\lambda) = 1 \wedge \overline{\mathsf{hcol}} \wedge \overline{\mathsf{fake}}\right] \approx 0.$

This claim follows from the strong unforgeability of the underlying signature scheme. The intuition is that the correct proof non-membership guarantees that the signature is not stored in the accumulator. Since it is not stored in the accumulator, the adversary did not receive this signature from the signing oracle and thus, is a valid forger w.r.t. strong unforgeability.

More formally, let $\mathcal{A}$ be an efficient adversary against the security of the VDS protocol. Then we construct an algorithm $\mathcal{B}$ against the strong unforgeability as follows. The input of $\mathcal{B}$ is a public-key vk and it has access to a signing oracle. It generates random elements $(p, \mathbb{G}, \mathbb{G}_T, e, g, g^s)$ for some s chosen at random from $\mathbb{Z}_p^*$, counters $\texttt{cnt} := 0$ and $\texttt{upd} := 0$, and $\mathcal{B}$ also generates an initially empty accumulator $f'(\mathcal{E}) := g$. It runs $\mathcal{A}$ on $pk = (p, \mathbb{G}, \mathbb{G}_T, e, g, g^s, vk, f'(\mathcal{E}))$ in a black-box way. Whenever $\mathcal{A}$ wishes to append an element d, then $\mathcal{B}$ increments the counter $\texttt{cnt} := \texttt{cnt} + 1$, chooses a fresh tag $\texttt{tag}$, sends $m' := d||\texttt{tag}||\texttt{cnt}$ to its signing oracle and forwards the response σ together with the corresponding proof of non-membership to $\mathcal{A}$.

It is understood that $\mathcal{B}$ records all s queries and answers. If $\mathcal{A}$ wants to update the ith element, then $\mathcal{B}$ adds the corresponding signature σ_i to the accumulator, increases the counter $\texttt{upd} := \texttt{upd} + 1$, picks a fresh tag $\texttt{tag}'$ at random, sends $m' := d'||\texttt{tag}'||\texttt{cnt}$ to its singing oracle and forwards the response together with a proof of non-membership and $g^{s^{\texttt{upd}}}$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ stops, outputting $((d^*, \texttt{tag}^*, i^*), (\sigma^*, w^*, u^*))$. The algorithm $\mathcal{B}$ then outputs $((d^*||\texttt{tag}^*||i^*), \sigma^*)$.

For the analysis, it's easy to see that $\mathcal{B}$ is efficient and performs a perfect simulation from $\mathcal{A}$'s point of view. In the following, let's assume that $\mathcal{B}$ succeeds with non-negligible probability, i.e., $((d^*, \texttt{tag}^*, i^*), (\sigma^*, w^*, u^*))$ satisfies the following: $(\hat{d}, i^*) \notin Q$, $\mathsf{Vrfy}(vk, d^*||\texttt{tag}^*||i^*, \sigma) = 1$, and $e(w^*, g^{e_i^*} \cdot g^s) = e(f'(\mathcal{E}) \cdot g^{u^*}, g)$, with $e_i^* := H(\sigma^*)$. We now argue, that $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does. To see this observe that $(\hat{d}, i^*) \notin Q$ and that the proof of non-membership verifies. Thus, conditioning on $\overline{\mathsf{fake}}$ and on $\overline{\mathsf{hcol}}$ the claim follows.

6 Experimental Results

We implemented the construction based on chameleon vector commitments (Sect. 4) and on accumulators (Sect. 5) in Java 1.7.

In this section, we provide comprehensive benchmarks of all proposed schemes to evaluate their practicality. We investigate the computational and bandwidth overhead induced by our protocols. We use the PBC library [11] in combination with a java wrapper for pairing-based cryptographic primitives (using a type D-201 MNT curve), and the Bouncy Castle Cryptographic API 1.50 [3] for all other primitives. For the construction based on accumulators, we used RSA-PSS with 1024 bit long keys as our underlying signature scheme, and SHA-1 as our hash function. Our experiments were performed on an Amazon EC2 `r3.large` instance equipped with 2 vCPU Intel Xeon Ivy Bridge processors, 15 GiB of RAM and 32 GB of SSD storage running Ubuntu Server 14.04 LTS (Image ID `ami-0307d674`).

We stress that this is an unoptimized, prototype implementation, and that better performance results may be achieved by further optimizations.

DATASET: We evaluated our schemes by outsourcing and then retrieving 8 GB, using chunk-sizes of 256 kB, 1 MB, and 4 MB. We measured the insertion and the verification time on the client side, as well as the sizes of all transmitted proofs.

Branching Factor: The CVC-based VDS was instantiated with branching factors of $q = 32, 64, 128$, and 256, and the public-key consists of q^2 elements that amount to 2.7 MB (for $q = 256$). This is in contrast to the public key size in the accumulator-based VDS which is about 350 bytes.

Number of Updates: To quantify the impact of updates on the accumulator-based construction, we performed separate experimental runs. Therefore, we performed 0, 10, 50 and 100 updates prior to retrieving and verifying the outsourced data set.

Block Sizes: The block size is a parameter which depends heavily on the application. Larger block sizes reduce the number of authenticated blocks, and thus yield a smaller tree with more efficient bandwidth and computation performance in the CVC-based VDS. However, larger blocks decrease the granularity of on-the-fly verification, since one has to retrieve an entire block prior to verification.

We took measurements for block sizes of 256 kB, 1 MB and 4 MB, which we believe is a sensible range of block sizes for various applications. For example, consider HD-video streaming with a bandwidth of 8 MB/s. Using a block size of 4 MB means that one can perform a verification every 0.5 s. However, in other applications such as the verifiable stock market, one may want to only retrieve a small fraction of the outsourced data set, for which a smaller block size such as 256 kB might be more desirable.

6.1 Streaming Data

Both proposed constructions have constant client-side insertion times as well as constant bandwidth overheads. The insertion proofs in the accumulator-basedand the CVC-based construction are 236 respectively 1070 bytes large.

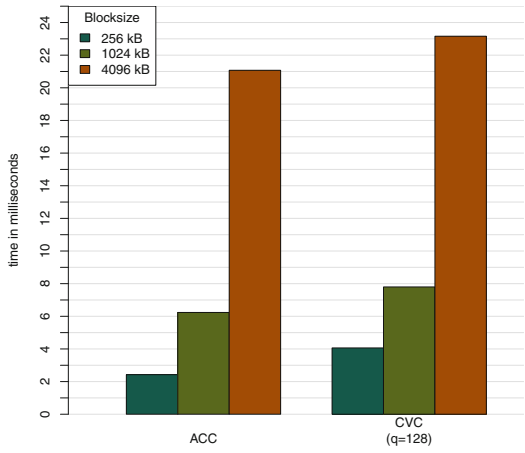


Fig. 2. Average insertion time for different block sizes (Color figure online)

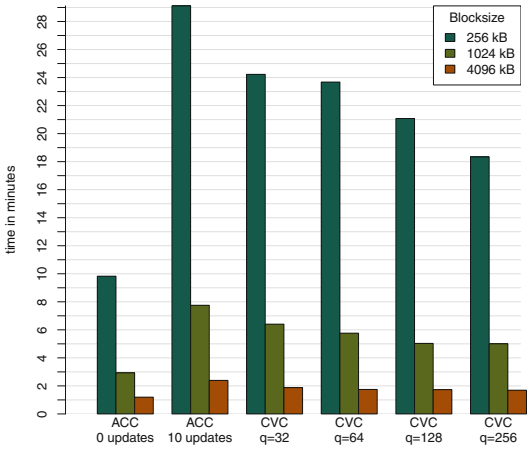


Fig. 3. Accumulated verification time for retrieving 8 GB (Color figure online)

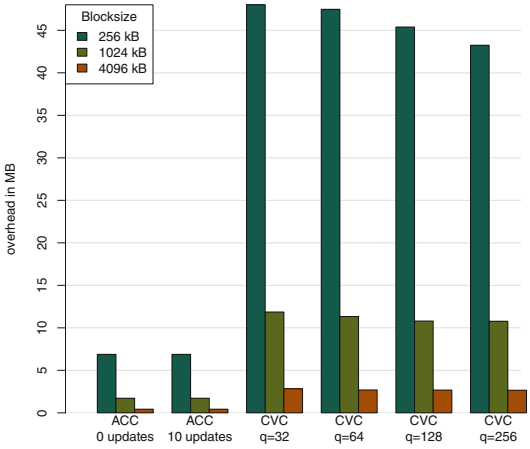


Fig. 4. Accumulated bandwidth overhead for retrieving 8 GB (Color figure online)

Rather than processing each block directly, we first compute its hash value and pass it to our VDS protocol. Since the insertion time is dominated by this hash function, we give the average insertion times for our different block sizes in Fig. 2. As one can see, the accumulator-based construction performs slightly better than the CVC-based one, and both give rise to quite practical timings for the insertion phase.

6.2 Verifying Retrieved Elements

The time needed for verifying the correctness of 8 GB data retrieved from the server is depicted in Fig. 3. In this figure, various insights about the performance behavior of our protocols are visible.

Firstly, the block size plays a crucial role for the overall performance, since the number of verifications needed to authenticate the 8 GB dataset depends linearly on the inverse of the block size.

Secondly, in the CVC-based construction higher tree arity results in shorter proofs, and therefore faster verification.

Thirdly, we observe the impact of updates on our accumulator-based construction. While without updates the accumulator-based construction is faster than the CVC-based one, adding just 10 updates decreases the performance of the accumulator dramatically. Still, for applications where only a few updates are expected, our accumulator-based construction is preferable.

6.3 Bandwidth Overhead

The bandwidth overhead incurred by retrieving 8 GB of data is shown in Fig. 4. As in Fig. 3 the impact of the block size is visible, as is the impact of the branching factor of the CVC-based construction. This figure also shows that the accumulator-based construction, although slower when handling updates, achieves a much smaller bandwidth overhead compared to the CVC-based construction. However, in this scenario for a block-size of 4 MB, both constructions introduce a total bandwidth overhead of less than 4 MB, or 0.05 %.

Acknowledgments. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – www.cispa-security.org) and the project PROMISE. Moreover, it was supported by the Initiative for Excellence of the German federal and state governments through funding for the Saarbrücken Graduate School of Computer Science and the DFG MMCI Cluster of Excellence. Part of this work was also supported by the German research foundation (DFG) through funding for the collaborative research center 1223. Dominique Schröder was also supported by an Intel Early Career Faculty Honor Program Award.

References

1. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003)
2. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
3. Castle, B.: The Legion of the Bouncy Castle. https://www.bouncycastle.org

4. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
5. Camenisch, J.L., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
6. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013)
7. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538 (2008). http://eprint.iacr.org/2008/538
8. Krawczyk, H., Rabin, T.: Chameleon signatures. In: ISOC Network and Distributed System Security Symposium - NDSS 2000. The Internet Society, February 2000
9. Krupp, J., Schröder, D., Simkin, M., Fiore, D., Ateniese, G., Nuernberger, S.: Nearly optimal verifiable data streaming (full version). Cryptology ePrint Archive, Report 2015/333 (2015). http://eprint.iacr.org/2015/333
10. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (2010)
11. Lynn, B.: PBC - C Library for Pairing Based Cryptography. http://crypto.stanford.edu/pbc/
12. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.G.: A general model for authenticated data structures. Algorithmica **39**, 2004 (2001)
13. Maurer, U.M., Wolf, S.: Diffie-Hellman oracles. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 268–282. Springer, Heidelberg (1996)
14. Naor, M., Nissim, K.: Certificate revocation and certificate update. IEEE J. Sel. Areas Commun. **18**(4), 561–570 (2000)
15. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
16. Papamanthou, C., Shi, E., Tamassia, R., Yi, K.: Streaming authenticated data structures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 353–370. Springer, Heidelberg (2013)
17. Papamanthou, C., Tamassia, R.: Time and space efficient algorithms for two-party authenticated data structures. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 1–15. Springer, Heidelberg (2007)
18. Perrig, A., Canetti, R., Song, D., Tygar, J.D.: Efficient and secure source authentication for multicast. In: ISOC Network and Distributed System Security Symposium - NDSS 2001, pp. 35–46. The Internet Society, February 2001
19. Perrig, A., Canetti, R., Tygar, J.D., Song, D.X.: Efficient authentication and signing of multicast streams over lossy channels. In: 2000 IEEE Symposium on Security and Privacy, pp. 56–73. IEEE Computer Society Press, May 2000
20. Ruffing, T., Kate, A., Schröder, D.: Liar, liar, coins on fire!: penalizing equivocation by loss of bitcoins. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–6 October 2015, pp. 219–230. ACM (2015)
21. Schröder, D., Schröder, H.: Verifiable data streaming. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) 19th Conference on Computer and Communications Security, ACM CCS 2012, pp. 953–964. ACM Press, October 2012

22. Schöder, D., Simkin, M.: VeriStream – a framework for verifiable data streaming. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 548–566. Springer, Heidelberg (2015)
23. Stefanov, E., van Dijk, M., Juels, A., Oprea, A.: Iris: a scalable cloud file system with efficient integrity checks. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC 2012, pp. 229–238. ACM, New York (2012)
24. Tamassia, R., Triandopoulos, N.: Certification and authentication of data structures. In: AMW (2010)

ARMed SPHINCS

Computing a 41 KB Signature in 16 KB of RAM

Andreas Hülsing[1(✉)], Joost Rijneveld[2(✉)], and Peter Schwabe[2(✉)]

[1] Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
andreas.huelsing@googlemail.com
[2] Digital Security Group, Radboud University, P.O. Box 9010,
6500 GL Nijmegen, The Netherlands
joost@joostrijneveld.nl, peter@cryptojedi.org

Abstract. This paper shows that it is feasible to implement the stateless hash-based signature scheme SPHINCS-256 on an embedded microprocessor with memory even smaller than a signature and limited computing power. We demonstrate that it is possible to generate and verify the 41 KB signature on an ARM Cortex M3 that only has 16 KB of memory available. We provide benchmarks for our implementation which show that this can be used in practice. To analyze the costs of using the stateless SPHINCS scheme instead of its stateful alternatives, we also implement XMSS^{MT} on this platform and give a comparison.

Keywords: Post-quantum cryptography · Hash-based signature schemes · Microcontroller · Resource-constrained devices · ARM Cortex M3 · SPHINCS-256 · XMSS^{MT}

1 Introduction

It is difficult to precisely predict the future of computing, but once large-scale quantum computers become feasible in practice, all of the asymmetric cryptography that is widely deployed today will be broken. Over the past few years, several schemes have been proposed that address this issue. One of the most promising classes of schemes that provide post-quantum secure digital signatures are hash-based schemes [21].

Hash-based schemes come with well-understood security guarantees, building only on the assumption of a secure cryptographic hash function. This makes them a very attractive, confidence inspiring choice. The keys they use and the signatures they produce are of practical sizes, and signing is reasonably fast.

This work was supported by the Netherlands Organisation for Scientific Research (NWO) under Veni 2013 project 13114 and by European Commission through the ICT program under contract ICT-645622 (PQCRYPTO). Permanent ID of this document: c7ea17f606835ab4368235a464e1f9f6. Date: 2015-10-27.

C.-M. Cheng et al. (Eds.): PKC 2016, Part I, LNCS 9614, pp. 446–470, 2016.
DOI: 10.1007/978-3-662-49384-7_17

Additionally, signature verification is very fast. Until recently, however, all practical hash-based schemes required a state that must be constantly kept up to date. In many real-world scenarios, this is a far reach from the signature schemes that are currently in use.

The introduction of SPHINCS [4] at Eurocrypt 2015 demonstrated that it is not strictly necessary to maintain a state for a hash-based scheme to be practical. Lifting this constraint does not come for free; it is paid for by an increase in signing time as well as signature size. Still, SPHINCS-256 remains fairly efficient in terms of these dimensions, with signatures of 41 KB and a signing rate of *"hundreds of messages per second on a modern 4-core CPU"* [4].

This shows that SPHINCS is a feasible solution on high-end servers and desktops, but the question remains, whether SPHINCS is also a feasible solution for small embedded "Internet-of-Things" devices. This is not merely a question of performance. It is a question of whether it is even possible to compute a 41 KB SPHINCS signature on a device with only little RAM. In this paper we demonstrate that it possible to compute and verify SPHINCS-256 signatures on an ARM Cortex M3 microcontroller with only 16 KB of RAM, but the performance results indicate that practical applications are limited to non-interactive contexts (such as sensor nodes sending signed data several times a day).

To illustrate the cost of eliminating the state in hash-based signatures, we furthermore implement the state-of-the-art stateful hash-based signature scheme XMSS^{MT} as described in a recent Internet draft [16]. To provide a fair comparison, we replaced all the used hash functions with similar functions as SPHINCS-256.

Availability of Software. We place all software described in this paper into the public domain. It is available online at https://joostrijneveld.nl/papers/armedsphincs.

1.1 Related Work

In [24], the potential for hash-based signature schemes on constrained microprocessors was first demonstrated. The authors establish that it is possible, to implement GMSS [6], an improvement of Merkle's original hash-based signature scheme, on an 8-bit AVR microprocessor at a speed comparable to RSA and ECDSA, although without key generation. The described platform offers 8 KB of program memory and 4 KB of SRAM.

A variant of XMSS was implemented on a 16-bit smart card [8]. The authors show that key generation can be done on the device and get even faster speeds than [24], further demonstrating practicality of (stateful) hash-based signature schemes on constrained devices.

Extensive side-channel analysis of a fast Merkle signature scheme implementation on an AVR ATxmega is presented in [10]. This paper introduces a new algorithm for the computation of authentication paths in a Merkle tree to significantly reduce (and actually bound) side-channel leakage during this computation.

Other post-quantum schemes also show promising results on embedded systems. In [12], a lattice-based signature scheme is shown to produce signatures of 9 KB, with keys of 2 KB and 12 KB in size, beating RSA in terms of speed, on a Xilinx Spartan-6 FPGA. While memory usage is slightly higher compared to hash-based schemes, [22] shows that lattice-based signatures can be very fast by providing an implementation on a Cortex-M4F. Multivariate-quadratic systems have also been implemented and proven to be practical on low-resource devices as well as ASICs, with keys of practical size [25].

The software presented in this paper is the first to describe (stateless) signature software achieving 128 bits of security against quantum attackers on an embedded microcontroller, which naturally makes comparison to previous results hard. On the one hand, none of the previous papers targets 128 bits of post-quantum security, and, unlike our software, both [10] and [8] use hardware accelerators for fast hashing. On the other hand, 8-bit AVR microcontrollers used in [6] and [10] and the 16-bit Infineon SLE 78 used in [8] are less powerful (and offer less RAM and ROM) than the more recent Cortex-M3 used in this paper. For many applications there is a trend to move from 8-bit and 16-bit microcontrollers towards more powerful 32-bit processors like the Cortex-M; mainly towards the low-end Cortex-M0, which is explicitely advertised to "achieve 32-bit performance at an 8-bit price point, bypassing the step to 16-bit devices" [20]. The Cortex-M3 is quite a bit more powerful than the Cortex-M0 and most importantly offers more RAM and ROM than the M0. Our memory usage suggests that it might be feasible to bring SPHINCS also to the M0 with 8 KB of RAM, but this would not leave any space for other applications and would thus be a mere academic challenge.

2 The SPHINCS-256 Signature Scheme

This section explains the SPHINCS-256 signature scheme as proposed in [4]. The section follows a top-down approach: we first explain why there is a need for SPHINCS and present a high-level overview. Then, we fill in details on each of the components. Rather than discussing the SPHINCS scheme in general, we will directly and explicitly adhere to the parameters and functions proposed as SPHINCS-256 in the original paper. Refer to [4] for a more general description.

2.1 Eliminate the State

In [17], Lamport describes a one-time signature scheme (OTS) that forms the foundation for hash-based signatures. This scheme has later been used and improved by Merkle [21] to build a many-time signature scheme. This is done constructing a binary hash tree on top of a series of OTS key pairs, effectively joining them together under a single long-term public key. When a sequence of authentication nodes is supplied as part of the signature together with an OTS public key, a verifier is able to reconstruct the long-term public key at the root of the authentication tree. See Fig. 1 for an illustration of this. This approach is

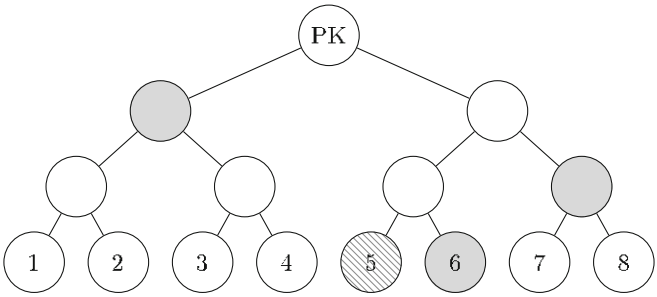


Fig. 1. The authentication nodes needed to authenticate leaf node 5 are coloured grey.

still the main construction used in modern hash-based signature schemes (such as XMSS [5]).

However, these constructions suffer from a fundamental problem. When using Merkle trees on top of OTS key pairs, the user should be very careful not to use a OTS key twice as this would undermine security. This implies that, in addition to storing the secret key (i.e., the seed that produces all OTS keys on the leaf nodes), one needs to store some indicator to keep track of the leaf nodes that have already been used[1]: the state. While this is not a problem in some applications, key management can quickly become an issue. In scenarios where multiple instances of the key are stored in different places (for example, backups, different machines used for load balancing, or different devices owned by the same user), the state needs to be constantly kept in sync among those copies. This makes such a signature scheme highly impractical and incompatible with many of today's systems.

Already in 1986, Goldreich recognized this problem and proposed a solution [11]: create a tree of such depth that, when randomly choosing an OTS key pair for each signature, the chance of accidentally reusing a certain OTS key pair becomes insignificantly small. This way, there is no need to keep track of the already used OTS keys. The obvious problem here is actually creating such a tree in the first place, but Goldreich is able to avoid this. By not simply hashing nodes together (as is the typical Merkle tree construction) but instead attaching an OTS key pair to each tree node and using that to sign the child nodes, it is never necessary to compute the entire tree. This requires that the OTS keys of the nodes along the path from a random leaf to the root node are deterministically generated out of order, but this can easily be done using pseudorandom function with a secret seed and the node index.

While Goldreich's system solves the issue of having to maintain a state, it introduces a new problem. As it replaces hashing with signing throughout the tree, it also replaces hash digests with OTS signatures for the authentication-path nodes included in each signature. This creates a new hurdle for practical

[1] In practice, this could be just the number of messages signed so far, as Merkle showed that using the keys sequentially is often preferable [21].

use, as it results in tremendously large signatures (more than 1 MB for reasonable parameters).

2.2 Overview of SPHINCS-256

As discussed above, the main problem with hash-based signature schemes is either the need to maintain a state or the size of the signatures. This has prevented hash-based solutions from being a drop-in replacement for the signature schemes that are currently in use. SPHINCS solves this by combining the approach of Goldreich with traditional Merkle trees in a nested construction and few-time signatures. This results in a stateless scheme with signatures of 41 KB and private and public keys of 1 KB each [4]. At "hundreds of messages per second on a modern 4-core 3.5 GHz Intel CPU", it is shown to be sufficiently fast for many practical applications. In later sections of this paper, we demonstrate that it is also practical on low-end devices with highly limited resources.

The nested trees construction forms the base of SPHINCS. The complete structure consists of a total of $h = 60$ layers, divided over $d = 12$ layers of sub-trees. This can be viewed as a hypertree of two levels of abstractions, where each node in the global tree represents a sub-tree. Each of these sub-trees then consists of $h/d = 5$ layers of nodes themselves. Let us refer to the sub-trees on layer i of the global tree as τ_i, where $i \in \{1, \ldots, d\}$. We refer to the nodes in a sub-tree as $\nu_{i,j}$, where $j \in \{1, \ldots, h/d\}$ is their level in the sub-tree and $i \in \{0, 2^j - 1\}$ the index within that level. There is no need to diversify between nodes or trees in the same layer at this point, as they each serve an identical purpose.

The trees τ_i are binary hash trees, only slightly varying from the original Merkle tree concept. Each of their nodes $\nu_{i,j}$ for $j \in \{1, \ldots, h/d - 1\}$ contains a digest of its child nodes, while the leaf nodes on layer h/d each contain the key of an OTS. For now, let us assume that we have some hash function H that generates these digests. As with Merkle trees, the digest at the root of the tree is used to authenticate the entire structure by constructing authentication paths.

All the sub-trees are then chained together as in Goldreich's system. Using the OTS keys in the leaf nodes $\nu_{i,h/d}$ of the trees τ_i, the root nodes of the trees τ_{i+1} are signed; a new sub-tree is chained to each of the leaf nodes. See Fig. 2 for a close-up of this construction. Nodes labeled H contain a hash of their child nodes; nodes labeled OTS include a key pair to authenticate their child node.

The OTS key pairs of the leaves of the trees on the bottom layer are not used to authenticate more sub-trees. Instead, they are used to authenticate the public key of a *few-time* signature scheme (FTS). An FTS behaves similarly to an OTS, but can be used several times before revealing too much of the secret key. By using an FTS rather than an OTS, SPHINCS does not require as many leaf nodes to maintain the same security level: the required maximal probability of selecting the same node repeatedly can be much higher without breaking the system. These FTS keys are used to sign the actual messages.

The above describes the basic outline of SPHINCS. This still far from a working algorithm, though, as we have assumed a number of black boxes: some

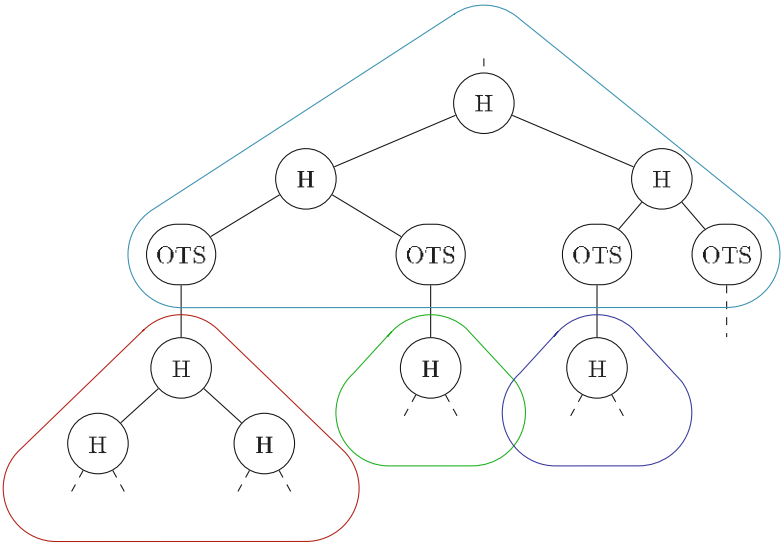


Fig. 2. Linking sub-trees together

hash function H to use in the sub-trees, an OTS to use between sub-trees and an FTS to sign the actual message at the bottom of the hypertree. In the following subsections, we will gradually collect the missing pieces.

2.3 Key Generation

Because of the hypertree structure, key generation for SPHINCS is a fairly cheap operation. We start by selecting some random values $SK_1 \in \{0,1\}^n$ and $SK_2 \in \{0,1\}^n$. For SPHINCS-256, $n = 256$. The first of these values is used for key generation, while the second is required for signing. This will be illustrated in the next subsection. Additionally, we generate a tuple Q of random bitmasks, each one also from $\{0,1\}^n$. These masks are used in the hash trees (as described in Sect. 2.6), as well as in the OTS and FTS – for now, let us merely acknowledge their existence. Thus $SK = (SK_1, SK_2, Q)$.

In order to generate the public key, we only need to generate the single tree in τ_1: the tree at the top of the structure. This requires generating the OTS keys along the bottom of this tree. Note that these keys need to be generated deterministically; using their address and SK_1 as input to some pseudorandom function we can derive a seed for this key. Then, a binary hash tree can be built on the public keys of the OTS key pairs, and the root node of this tree is part of the SPHINCS public key: PK_1. As the bitmasks are also needed for verification, they must also be included in the public key: $PK = (PK_1, Q)$.

It is worth noting that, while SK_1, SK_2 and PK_1 are all only 256 bits (or 32 bytes) in size, Q is significantly larger. SPHINCS-256 uses 32 bitmasks in Q, which add up to a total size of $32 \cdot 32 = 1024$ bytes. Bitmasks thus account for

the largest part of the keys. In general, the number of bitmasks is determined by the part of the scheme that requires the largest number of them – the FTS, the OTS or the hash trees.

2.4 Signing

Public-key signature schemes typically compute a hash of the message that is to be signed, and then sign that hash. This ensures that the input is of a constant, relatively small length. In a stateless scheme like Goldreich's, a random key pair at the bottom of the tree would then be selected to sign the hash. In SPHINCS, however, the key pair is selected based on the message hash itself. In order to prevent attackers from specifically targeting certain key pairs, some random or unknown factor still needs to be included – this is what SK_2 is for. We first compute a bitstring $(idx\|R)$ using a pseudorandom function that takes SK_2 and the message as input, and use idx to select an FTS key pair. The second part R is used to compute a randomized digest D of the message. This digest is what we will be signing. As a practical result of all this, the selection of an FTS key pair is completely deterministic with respect to a secret key SK_2 and a message M.

After selecting a particular FTS key pair, the secret key of this key pair needs to be generated (based on a seed derived from its location and SK_1) and is then used to sign D and produce the signature σ_{FTS}. Together with the message-specific randomness R generated above and the index idx of the selected key pair, this signature forms the first part of the SPHINCS signature Σ. As SPHINCS uses an OTS and an FTS for which the public keys can be derived from their respective signatures (as we will see in Sects. 2.7 and 2.8), there is no need to include the FTS public key here.

We then generate the OTS key pair for its parent node in $\nu_{d,h/d}$ in the relevant sub-tree in τ_d (again using its position in the tree in combination with SK_1), and use it to sign the FTS public key. Let us refer to the produced signature as $\sigma_{OTS,d}$. This signature is also added to Σ. The public key of this OTS needs to be authenticated, so we compute all nodes along its authentication path throughout the tree in τ_d and include those in Σ as well. We refer to the nodes along the authentication path in the selected tree on layer d as $Auth_d$. Upon reaching the root of the tree in this fashion, we generate the OTS key pair that belongs to its parent node in $\nu_{d-1,h/d}$ and use that to sign the root. This procedure continues all the way up to the root of the one tree in τ_1, which is, by construction, included in PK. While progressing up the hypertree, all OTS signatures and nodes along the authentication paths need to be added to Σ.

Altogether, the SPHINCS signature Σ now contains the message-specific randomness R, the index idx of the selected FTS key pair, the FTS signature σ_{FTS} and d pairs of OTS signatures and sequences of nodes along the authentication path $(\sigma_{OTS,i}, Auth_i)$. Everything combined, $\Sigma = (idx, R, \sigma_{FTS}, (\sigma_{OTS,1}, Auth_1), \ldots, (\sigma_{OTS,d}, Auth_d))$.

2.5 Signature Verification

The procedure for verifying a signature on M is very similar to signing. As we have seen above, the signature Σ contains the message-specific randomness R, the FTS signature and the OTS signatures and authentication paths. After computing D (using R and M), the FTS signature is verified. As mentioned above, the verification function of the OTS and FTS used in SPHINCS output the respective public key. Hence, the OTS signature on the FTS public key can now be verified, resulting in the respective OTS public key. As the authentication path is also given in Σ, the root node of the tree in τ_d can now be computed. Similar to the way the signature was generated, we now continue up the tree along the authentication paths while verifying the signatures on the root nodes of each sub-tree.

In the end, the verification eventually arravies at the root node of the single tree in τ_1. This root node should be equal to PK_1, included in PK. If this is the case, the signature is valid.

2.6 Hash Trees

At its core, SPHINCS heavily relies on hash trees. The construction of these trees is slightly different from the classical binary Merkle trees. After concatenating the values of the two child nodes, they are not immediately fed to a hash function to produce the parent node. Instead, two *bitmasks* are applied first; let Q_i, Q_{i+1} be such bitmasks and h_2, h_3 child nodes, then $h_1 = H((h_2 \| h_3) \oplus (Q_i \| Q_{i+1}))$.

In [2], XORing with bitmasks is introduced as part of a linear hashing scheme, and it is employed in [9] in order to construct binary hash trees that do not require the underlying hash function to be collision resistant. Instead, second-preimage resistance is sufficient to attain unforgeability. This hash-tree construction is the one described above, and is used in SPHINCS.

2.7 The FTS: HORST

At the bottom of the hypertree, SPHINCS relies on an few-time signature scheme. For this, a variation of an FTS called HORS [23] is used. This variant, referred to as HORST, adds a tree construction to plain HORS [4]. The configuration of HORST consists of two parameters that control the security level, signature size, and key sizes: t and k, where t is a power of 2. For SPHINCS-256, we have $t = 2^{16}$ and $k = 32$.

As mentioned in the overview of SPHINCS, the key is seeded based on SK_1 and the location of a particular HORST instance in the hypertree. This seed is expanded to t secret-key components to form $sk = (sk_0, \ldots, sk_t)$, which are then hashed to create the public-key components pk_i for $i \in \{0, \ldots, t\}$. In SPHINCS-256, each sk_i and pk_i has 32 bytes. The HORST variation then proceeds to build a hash tree on top of the public-key components. The root of this tree is the actual HORST public key pk. For this tree, SPHINCS also makes use of bitmasks as described in Sect. 2.6.

Signing a message M using a HORS key pair is done by splitting the message into k pieces of length $\log_2 t$. Each of these pieces M_i is then used as an index to address a piece of the secret key, sk_{M_i}, which is subsequently revealed. For HORST, the nodes along the authentication path from these hashes to the root of the tree are also required as part of the signature.

Verification is quite similar: first, the revealed pieces of sk are hashed, and the message is split into k parts. These parts are then interpreted as integers and used to place the pieces of sk on the appropriate leaves. Using the nodes supplied in the signature, the path to the root node can then be computed. This is done for all nodes and authentication paths checking that all agree on the same root. If this is not the case, verification fails. The verification algorithm then outputs this root as the public key – comparing it to the actual public key will reveal if the signature was valid.

What makes HORS usable as a few-time signature scheme (as opposed to an OTS) is the choice of a sufficiently large t in relation to k. This implies that only a small part of the secret key is revealed for each signature, and the chance of a successful forgery after obtaining just a few signatures diminishes. Note that this does require that an adversary cannot control the message hash for which a signature is obtained. As we have seen above, this requirement is satisfied in SPHINCS. In the original HORS scheme the combined size of public key and signature would increase linearly with t, but the HORST variation only incurs logarithmic growth in t, caused by the length of the authentication paths. This makes it possible to use HORST as the FTS in SPHINCS without dramatically increasing the key length.

2.8 The OTS: WOTS+

For the OTS that links the sub-trees together, SPHINCS uses WOTS+. This variation of the Winternitz OTS is proposed in [14], designed to reduce the signature size even further than other WOTS-based schemes. As in WOTS, the Winternitz parameter $w = 16$ is used to configure the efficiency trade-off. Likewise, one then derives ℓ (consisting of ℓ_1 and ℓ_2) from this parameter and the security setting $n = 256$ as follows.

$$\ell_1 = \left\lceil \frac{n}{\log w} \right\rceil, \qquad \ell_2 = \left\lfloor \frac{\log (\ell_1(w-1))}{\log w} \right\rfloor + 1, \qquad \ell = \ell_1 + \ell_2.$$

For the SPHINCS-256 configuration, it can be readily computed that $\ell_1 = 64$ and $\ell_2 = 3$, thus $\ell = 67$.

In the plain WOTS scheme, a function F is applied to the secret key several times to produce a hash chain. In WOTS+, however, we take into account the bitmasks. In each iteration, before applying F, the input is XORed with a round-specific bitmask Q_i. The chaining function then looks as follows (where the base case is $c^0(x) = x$):

$$c^i(x) = F(c^{i-1}(x) \oplus Q_i)$$

In order to guarantee deterministic signatures here as well, WOTS+ key pairs are also seeded using SK_1 and their location in the tree. This seed is then expanded to a secret key of $\ell = 67$ pieces, $sk = (sk_1, \ldots, sk_\ell)$. Generating the key is very similar to traditional WOTS; we simply apply the chaining function c for a total of $w-1$ times to each part of sk to obtain $(pk_1, \ldots, pk_\ell)$. In SPHINCS-256, each of the sk_i and pk_i has again 32 bytes. As the reader might be expecting by now, we proceed by building a hash tree on top of these public-key parts. These trees make up the third abstraction level of trees in the hypertree. However, ℓ is not necessarily a power of two – in fact, as w and n (and thus ℓ_1) typically are a power of two, ℓ is not. This requires the use of a slightly different tree construction: the L-Tree [9]. The structure is entirely identical to binary hash trees, except for the rightmost nodes. Whenever the number of nodes on the current layer is odd, the rightmost node is lifted up to the next layer instead. See Fig. 3 for an example with five nodes. The root of this tree is the public key pk.

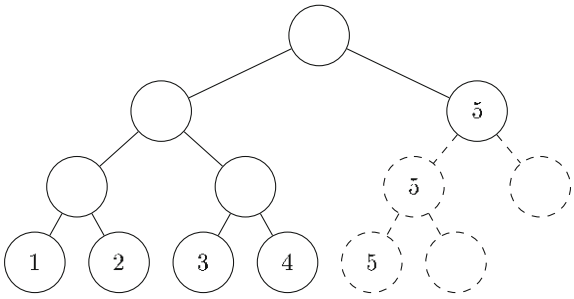


Fig. 3. An L-Tree with five leaf nodes

In order to sign a message M, we first interpret it as an integer in binary, and then express it in base w. This effectively splits M into a sequence of values that can be at most $w-1$, each. Note how there will be at most ℓ_1 values, as per the construction of ℓ_1. We write $M = (M_1, \ldots, M_{\ell_1})$. Furthermore, a checksum needs to be computed to prevent an attacker from being able to forge a signature: $C = \Sigma_{i=1}^{\ell_1}(w - 1 - M_i)$, which is also expressed in base w: $C = (C_1, \ldots, C_{\ell_2})$. The lists M and C are then chained together to form $B = (b_1, \ldots, b_\ell) = M \| C$. These values in B are then used as lengths for the Winternitz chains, producing the signature $(\sigma_1, \ldots, \sigma_\ell) = (c^{b_1}(sk_1), \ldots, c^{b_\ell}(sk_\ell))$.

Verifying a WOTS+ signature is very similar to the regular WOTS scheme, except that one needs to take special care to use the correct bitmasks when applying the chaining function[2]. Let us define the function v to account for this as

[2] Note that this approach differs slightly from the one presented in [14]. In the original definition this is solved by supplying the appropriate set of bitmasks as an argument to the chaining function.

$$v^{i,j}(x) = F(v^{i,j-1}(x) \oplus Q_{w-i+j-1}).$$

Like for the original chaining function, the base case is $v^{i,0}(x) = x$ and we abbreviate $v^{i,i}(x) = v^i(x)$.

We then compute B in the same way as in the signing procedure, and then compute the public key parts $(pk_1, \ldots, pk_\ell) = (v^{w-1-b_1}(\sigma_1), \ldots, v^{w-1-b_\ell}(\sigma_\ell))$. As was the case during the key generation step, the last step that remains is computing an L-Tree over these pieces. The verification algorithm then outputs the root of this tree. Like in HORST, this can then be compared to the actual public key to verify that the signature was valid.

2.9 *H* and *F*: ChaCha

Two key elements of SPHINCS have not yet been discussed. Practically the entire scheme consists of computing hashes. In WOTS+, some hash function $F : \{0,1\}^n \to \{0,1\}^n$ is used to construct the chaining function, and throughout the entire hypertree, $H : \{0,1\}^{2n} \to \{0,1\}^n$ is used to construct binary hash trees. The function F is also used to compute the HORST leaf nodes based on the secret key. For the performance of the scheme, it is crucial that these functions are sufficiently fast. An important characteristic of both of these functions is that they do not need to be able to take arbitrarily long input. This makes it unnecessary (and wasteful) to select a hash function that can. As it turns out, being able to accept arbitrarily long input is one of the properties that typically slow down hash functions. Instead, SPHINCS uses a permutation-based construction following the sponge design using the permutation from the ChaCha stream-cipher family [3].

The core of ChaCha is a 512-bit permutation, so in order to use it for F, the input needs to be padded to extend it. In SPHINCS, the authors chose the 32-byte ASCII string $C =$ "expand 32-byte to 64-byte state!". The output of F is obtained by truncating the output of the ChaCha permutation to 256 bits. H is constructed similarly. Let $Chop(M, i)$ be a function that truncates M to i bits, M_1 and M_2 be strings of 256 bits, and O be a string of 256 zero-bits, then

$$F(M) = Chop(\pi_{ChaCha}(M \| C), 256), \text{ and}$$
$$H(M_1 \| M_2) = Chop(\pi_{ChaCha}(\pi_{ChaCha}(M_1 \| C) \oplus (M_2 \| O)), 256).$$

Now only a few minor pieces of the puzzle remain. Creating the message-specific random value R is done by calling BLAKE-512$(SK_2 \| M)$ [1], and BLAKE-512 is used once more to create the digest D. In order to derive the secret keys for the HORST and WOTS+ key pairs based on their location and SK_1, the BLAKE-256 function is used as a pseudo-random function. Along the bottom of each of the trees, the ChaCha12 stream cipher is used to generate HORST and WOTS+ keys. An complete overview of the SPHINCS-256 hypertree is shown in Fig. 4.

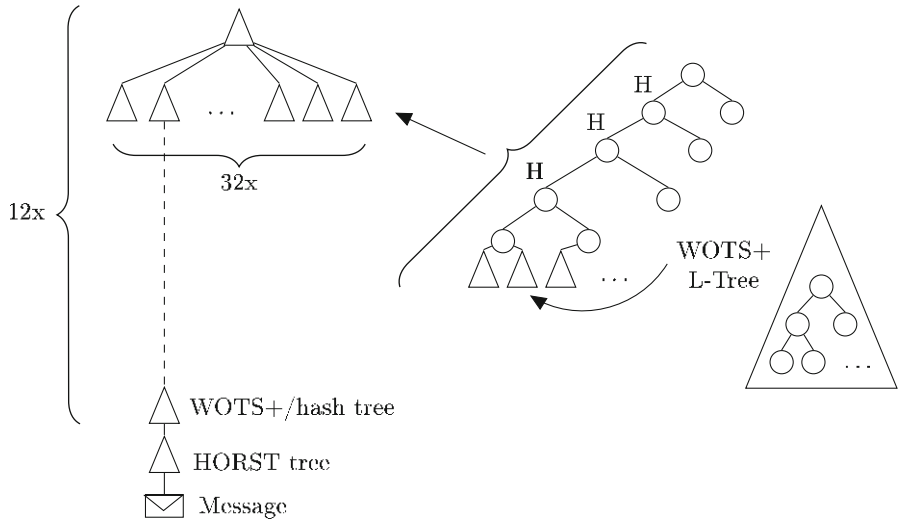


Fig. 4. The complete hypertree structure

3 The ARM Cortex-M3 Microcontroller

The platform of choice for this implementation is the ARM Cortex M3, and development was carried out using the STM32L100C discovery board. The Cortex M3 is a 32-bit microprocessor with sixteen 32-bit registers, thirteen of which are available for general-purpose computation (leaving three for the stack pointer, link register and program counter). This microcontroller is commonly found in embedded systems used in the automotive industry, small industrial systems and (wireless) sensors, but as we mentioned in Sect. 1.1, the low-end 32-bit Cortex-M processors are also starting to replace 8-bit and 16-bit microprocessors (given the similarly low production costs and efficiency, but larger computational power). As opposed to the ARMv6-M [18] architecture found among the smaller processors in the Cortex M-series, this processor supports the ARMv7-M [19] architecture.

The ARMv7-M architecture is aimed at microcontrollers, and is highly optimized for low-cost devices. It only supports the Thumb-2 instruction set (as opposed to the full ARM instruction set) and its register scheme is very straightforward – there are no SIMD instructions or extended register banks. This is somewhat compensated for by the efficient STM and LDM instructions. While these cannot be pipelined with other instructions, they provide internal pipelining when transferring more than one register to or from memory, making it not so costly to swap out the register contents. In Thumb-32 instructions, there are some limitations to their use on the SP, PC and LR registers, but none of this is a direct obstacle in our use-case.

The STM32L100C is part of the STM32 ultra-low-power series. The processor runs at a clock speed of 32 MHz, and the board offers 16 KB of RAM. This makes

it a highly constrained platform for the implementation of SPHINCS, given that the stack usage of the Haswell implementation [4] runs into several megabytes. Notably, this means that the available RAM is insufficient to store the signature, which weighs in at 41 KB.

In order to communicate with the device at runtime, we make use of serial communication over USART. This can be done efficiently using the direct-memory-access (DMA) controller, to prevent blocking the computation while waiting for the communication interface. Doing this, we are able to communicate reliably at a baud rate of 921 600 Bd. To be able to configure and use this from a more abstract level, we made use of the open-source libopencm3 firmware library.

4 Implementing SPHINCS-256 on the Cortex-M3

The main contribution of this work is to show that SPHINCS-256 can be implemented on resource-constrained devices. The Cortex M3 is quite constrained both in terms of available volatile memory as well as processor speed, serving as a proof of concept that this does not render SPHINCS unusable. In this section, we describe implementation-specific design choices and present the achieved speed results. It should further be noted that this implementation makes use of code from the SPHINCS reference implementation [4] as well as (parts of) implementations of BLAKE-256 and BLAKE-512 [1] and the ChaCha12 stream cipher [3].

4.1 Signing Within 16 KB

In a hash-based signature scheme, signing and verification are very similar in design, but they differ significantly when actually being performed. For verification, much of the work has already been done when producing the signature, and there is no need to construct entire trees from the ground up. The same is the case for SPHINCS. While verification is fairly straight-forward in terms of memory use, signing requires more effort to get right.

The general approach is as follows. In order to reduce the memory consumption of the signing operation, we split the computation into disjunct parts, and process the output of each part before continuing with the next part. This makes sure that we only have to account for the memory requirements of each such part at a time, instead of the consumption of the entire operation. The remaining task is now to find suitable points at which to split the computation such that memory use is sufficiently low in each of the parts, without introducing too much performance overhead.

Tree Storage. The SPHINCS scheme consists of a number of clearly distinct components, with the HORST trees and WOTS+/hash trees as the two most prominent subdivisions. While the memory usage is typically large at the base of a tree, it fans in again as one progresses towards the root. As each tree is stacked on top of the one below, it is not necessary to ever store more than one

tree in memory at a time before proceeding on to the next – this progression is highly sequential.

For the WOTS+/hash trees, the available memory is not an immediate problem. Producing a single WOTS+ signature impacts the available memory for only 67 bytes. At $32 \cdot 67 = 2144$ bytes, the WOTS+ public key itself is slightly larger, but this can be quickly reduced to a 32 byte root node by applying the L-Trees we have seen in Fig. 3. This does imply that one really has to perform this reduction before continuing with the next WOTS+ leaf node, but in the current context this has no negative impact. After processing all leaf nodes in this fashion, one is left with 32 leaf nodes of 32 bytes each. Each authentication tree contains only $h/d = 5$ layers of hashing, resulting in a total of $2^6 - 1 = 63$ nodes, which can be stored in memory all at once. After computing the entire tree, the nodes along the authentication path can be conveniently selected.

HORST, on the other hand, is a different beast entirely. With $t = 2^{16}$ and $k = 32$, the trees contain 131071 nodes spread over 16 layers of hashing, making these trees much higher than the hash trees on top of the WOTS+ signatures. This means that the method of first building the entire tree and then extracting the authentication path is not feasible. At 32 bytes per node, the nodes alone would require 4 MB of storage. There is no need to store the entire tree, though, as only a very specific set of nodes is relevant for the signature: the nodes along the 32 authentication paths, as well as the root node. As we do require the root node to authenticate the tree, there is definitely no escaping having to *compute* the entire tree.

Treehash. In [4], the authors mention that RAM usage and code size was not one of the concerns when writing the optimized implementation – the implementation was optimized for speed on a platform where memory was available in abundance. They remark that, if saving memory is a concern, the treehash algorithm could be used. This is what we will now apply in order to compute the HORST authentication path.

The treehash algorithm is another contribution by Merkle [21, Sect. 7]. It has since been used in various forms as the basis of tree-traversal algorithms. A common approach is expressed by the pseudo-code (based on [13]) in Algorithm 1, and discussed below.

Algorithm 1. One round of the treehash algorithm

Require: STACK, next leaf node N
Output: STACK is updated
1: **while** STACK.peek() is on same level as N **do**
2: neighbour ← STACK.pop()
3: N ← H(neighbour ‖ N)
4: **end while**
5: STACK.push(N)

The core idea is to grow a tree, using its leaves in subsequent order and only maintaining a collection of the currently relevant nodes: the 'heads' of the different branches. As new nodes are added, these branches are gradually grown to completion, and merged when needed. Any nodes that occur deeper in the tree can safely be forgotten (for the purpose of finding the root node), as each node is only required once to generate its parent. Each round of treehash consists of introducing the next leaf node and updating the heads of the branches until no more new nodes can be computed. For half of the leaves (i.e., the 'left neighbors'), their introduction does not allow for the computation of any new parent nodes, while a quarter of the leaves allows us to compute one parent node, etcetera.

When examining how the set of relevant nodes evolves, there is a strict ordering in when these nodes become relevant again, based on their level in the tree. It can be easily observed that nodes are always consumed in a last-in-first-out manner – the set is really a stack. After introducing all leaf nodes and completing the last round, the root node will be the only node left on the stack. Another important observation here is the fact that there are never two nodes of the same tree level on the stack at the same time. The nodes on the stack are inherently ordered by their tree level (nodes that occur higher in the tree are stored deeper down the stack). As leaves are consumed in subsequent order, two nodes at the same level have to be neighboring nodes that can immediately be used to produce their parent node. This allows us to conclude that using treehash for HORST requires a stack that can hold $log(t) = 16$ nodes. At 32 bytes per node, this easily fits in the available memory.

Besides computing the root node of a HORST tree, however, we are particularly interested in the nodes along the authentication paths from the leaves used to produce the signature, to the top of the tree. The position of these nodes on the stack is less easy to predict, but we do not want to compute parts of the tree more than once in order to gather all required nodes. Intuitively, a way to resolve this is by somehow recognizing the nodes that need to be included in the signature while performing the treehash rounds. Navigating through the tree without actually computing the node values is cheap, allowing us to trace the authentication paths from leaf to root and observe which nodes will need to be output. Rather than compiling a list of these nodes and performing costly lookups, we can compute and store in which treehash round they will be produced, as well as their position in the signature[3]. Algorithm 2 shows how to compute the round numbers of all nodes along the authentication path for a given leaf-node index.

Consider that the tree consists of $2^{17} - 1 = 131071$ nodes, but only 320 nodes[4] are relevant. Because of this, only a small subset of all treehash rounds contains

[3] As nodes of the various authentication paths will be generated interleaved, it is necessary to rearrange them accordingly.

[4] One might expect to require $32 \cdot 16 = 512$ nodes, as each of the 32 authentication paths results in 16 neighboring nodes. However, in order to prevent needless duplication in the top layers, the HORST signature always includes layer 6 in its entirety and truncates the authentication paths after 10 nodes, leaving it to the verifier to reconstruct the paths.

Algorithm 2. Computing treehash round numbers

Require: idx
Output: treehash round numbers of authentication nodes

$roundno \leftarrow idx + t$
$roundnumbers \leftarrow [\,]$
for $i \in \{1, \ldots, \log(t)\}$ **do**
  ▷ Find the neighbour node's round number..
  if $idx \bmod 2 = 1$ **then** ▷ (idx is a 'right-node')
    $roundno \leftarrow roundno - 2^{i-1}$
    $idx \leftarrow idx - 1$
  else ▷ (idx is a 'left-node')
    $roundno \leftarrow roundno + 2^{i-1}$
  end if
  $roundnumbers$.APPEND($roundno$)
  ▷ ..and move up to the parent node.
  if $idx \bmod 2 = 0$ **then**
    $roundno \leftarrow roundno + 2^{i-1}$
  end if
  $idx \leftarrow idx/2$
end for
return $roundnumbers$

relevant nodes. This makes it especially important to optimize recognizing relevant treehash rounds.

An efficient way to recognize which nodes need to be included in the signature while performing treehash is by storing bitmasks for each of the relevant rounds. By sorting these bitmasks by their round index, one can iterate over the mask-index pairs while processing each of the leaf nodes. Pointing an iterator at the current mask-index pair and only incrementing it when the index is equal to the index of the current leaf node will result in an overhead of only one comparison for each non-relevant round.

Streaming Out Signature Data. In the previous section, we have glossed over an important aspect of the signing process: constructing the signature. Where an implementation with an abundance of memory available would simply allocate 41 KB of memory and insert the different pieces of the signature in the right place as they are computed, this is not possible on our device. Instead, the signature is streamed out of the board over the serial port throughout the computation. For many applications, this is not much different from receiving the entire signature all at once after the entire computation has finished, so we believe this should not pose any immediate usability concerns.

As was discussed in Sect. 2.4, the SPHINCS signature consists of a number of different components. Recall that $\Sigma = (idx, R, \sigma_H, (\sigma_{W,1}, Auth_1), \ldots, (\sigma_{W,d}, Auth_d))$, where, for brevity, H and W now denote the HORST FTS signature and the WOTS+ OTS signatures, respectively.

The values idx and R are generated at the start of the signing procedure, and can be written to the output stream immediately. The WOTS+ signatures

$\sigma_{W,i}$ and sequences of nodes $Auth_i$ are generated in the same order as the order in which they are supposed to be arranged in Σ, so this does not lead to any difficulty, either – instead of storing them in memory, we simply write these values to the output stream as they are computed.

The HORST signature σ_H is a bit more complicated. It consists of k pairs of secret keys belonging to leaf nodes, and sequences of nodes along the path from each of these leaf nodes towards the top of the tree. As remarked in footnote 4 on page 16, all nodes on layer 6 are always included, so the last 6 nodes of these sequences are truncated. The issue here is the fact that the node sequences are not produced one at a time, but are each grown in an interleaved fashion as more and more of the tree is computed. When storing the hash values in a signature in memory, this does not pose a problem – each node value can be inserted in the right place. When streaming the output, however, one cannot go back and insert a node value. Instead, the node values will have to be tagged with what should have been their location in the signature, and rearranged accordingly on the receiving end of the communication. For each 32-byte node value, this adds an overhead of two bytes. While this may seem significant, in the end it results in an increase of 832 bytes (640 for the authentication path nodes, 128 for the nodes on layer 6 and 64 bytes for the secret keys). Considering that the entire SPHINCS signature is 41 KB, this can be considered acceptable.

HORST Key Material. Similarly, generating a HORST secret key (based on the seed SK_1 and its location in the hypertree) results in too much key material to fit in memory. With 2^{16} leaf nodes of 32 bytes each, this would amount to 2 MB. Instead, we can once more rely on the fact that treehash rounds consume the leaf nodes sequentially, and only generate the leaf node values when they are required. To achieve this, we briefly store the intermediate state of the ChaCha12 stream cipher instead, initially seeding it in the regular fashion for HORST. We then perform the next iteration based on the stored state whenever more key data is required. This allows for the generation of leaf node values on the fly. As ChaCha12 produces output blocks of 512 bits, every other leaf node requires a new chunk of output to be generated.

Streaming the Message. On the subject of streaming data, it should also be remarked that for the Cortex M3 to be able to sign messages of a length larger than the available memory, it is necessary to process the message in a streamed fashion as well. This is possible, but requires the message to be streamed twice. In Sect. 2.4, we described that the message is first used together with the secret key to generate a message-specific random value R, after which the message digest is generated. As this digest is computed as the hash of the concatenation of a part of R and the message (in that order), the message needs to be available twice. As storing it on the device is not an option for large messages, it needs to be streamed twice. The additional overhead is minimal as it can be streamed in block by block while performing the BLAKE512 hash using direct memory access.

4.2 Performance

The previous subsections show some of the adjustments required to be able to generate SPHINCS signatures on a platform with only 16 KB of volatile memory. Besides memory usage, time is also a relevant metric to consider. In fact, the running time very much determines usability in practice.

ChaCha Permutation. When considering SPHINCS-256, one of the key observations here is the repeated use of the ChaCha permutation. It is the fundamental building block in both WOTS+ and HORST, as well as the hash trees that make up the rest of the hypertree.

Recall that $t = 2^{16}$. In order to generate a HORST key and produce a signature, ChaCha is used $\frac{1}{2} \cdot t = 32768$ times to expand the seed and generate the secret keys, as the permutation outputs 512 bits and the keys are 256 bits each. These secret keys are then hashed using F to construct the leaf nodes at the cost of another $t = 65536$ permutations. Subsequently, treehash is used to hash together t leaf nodes, at a cost of two ChaCha permutations per parent node (as follows from the construction of the function H in Sect. 2.9), resulting in another $2 \cdot (t-1) = 131070$ permutations. All in all, this results in 229374 calls for one HORST signature.

WOTS+ is significantly cheaper. Recall that $\ell = 67$ and $w = 16$. Generating a WOTS+ key pair requires ℓ secret keys, which costs $\lceil \frac{1}{2} \cdot \ell \rceil = 34$ permutations to expand the seed, $\ell \cdot (w-1) = 1005$ invocations of F at one permutation each for the chaining function and 66 invocations of H to build the L-tree, totaling $34 + 1006 + 2 \cdot 66 = 1171$ permutations. Each of the trees in the hypertree has 32 WOTS+ leaf nodes, costing a total of $32 \cdot 1171 = 37472$ permutations per tree.

Constructing a tree with WOTS+ key pairs on the leaf nodes costs an additional 31 invocations of H. One of the WOTS+ nodes is used to produce a signature on the sub-tree below, at the average cost of $\lceil \frac{1}{2} \cdot \ell \cdot (w-1) \rceil = 503$ more invocations of F. As there are trees 12 in the hypertree, this leads to a total of $12 \cdot (37472 + 2 \cdot 31 + 503) = 456444$. Summing the cost of HORST and the WOTS+ trees, we arrive at a grand total of $229374 + 456444 = 685818$ permutations.

Because we perform so many ChaCha permutations, it is worthwhile to optimize this in ARMv7-M assembly. Internally, the ChaCha permutation operates on sixteen words of 32 bits each. These fit precisely in the 32-bit registers that are available to us on this platform, and the arithmetic in ChaCha is very simple to perform once the words are accessible. There are not enough registers available for all of these words, though, as register 13, 14 and 15 are reserved for the stack pointer, link register and program counter, respectively. This would imply that three of the sixteen words would need to be saved in memory at all times, at the cost of a load and a store whenever one of these is needed. While we need the program counter and stack pointer for the code to run properly, we are not making any function calls that require the link register – the extra cost of having to pop it from the stack in the end is easily compensated by the benefit of an extra general purpose register. Now that we have fourteen registers to work with, we can arrange the order of the round internals of the ChaCha

permutations such that we only need to switch out the two words on the stack once every round, on average. Doing so, we arrive at 738 cycles for one permutation; in the context of the ChaCha12 stream cipher, this corresponds to around 23 cycles per byte.

Key Generation. Generating a SPHINCS-256 key on the device takes 35 423 182 cycles. At 32 MHz, this amounts to just over a second. As one would expect, virtually all of these cycles can be attributed to WOTS+ key generation. When it comes to key generation, it should be noted that the STM32L100C discovery board that we used for these benchmarks is not equipped with a random number generator. Instead, we used a hard-coded 32-byte value that is included when we flash the device. While in practice, using this board, key generation would have to be done off the board, our results show that for similar boards with a TRNG on-board key generation is not only feasible but practical.

Signing. Producing a signature takes 729 942 616 cycles, or approximately 22.81 s. As described above, we cannot store the signature on the board – this requires communication to a host outside of the board. Using direct memory access, we can efficiently interleave control of this communication with computations. If we disable communication and instead discard the signature as it is being produced, the signing procedure requires 725 933 925 cycles (for messages of small length, so as to focus the benchmark on penalty of signature output). This shows that the overhead is noticeable but not significant. In practice, this is a factor that may vary slightly depending on the specific context and interfaces available.

In terms of RAM usage, the signing procedure ends up using 8 755 bytes of stack space. Note that some of this stack usage is the result of function inlining by the compiler, to prevent having to perform function calls. When disabling this behavior, the stack space consumption is reduced to 6 619 bytes. Furthermore, we observe that the current implementation requires 25 KB of flash memory (or 19 KB, without inlining). These results show that there is a sufficient amount of memory left on the device (in terms of both RAM and ROM) for other applications, but also indicate that moving to even smaller devices (such as the Cortex M0) would be quite challenging.

Verification. Verification is much more straight-forward. The memory limit does not necessitate any significant changes like it did for signature generation, as the verification procedure never requires the construction of a full tree. The signature needs to be streamed to the device, but this does not complicate processing, as the node values arrive in the order in which they are to be consumed. At 17 707 814 cycles, verification takes roughly 553 milliseconds. When ignoring the communication and operating on bogus data instead, verification requires 8 263 801 cycles. The communication penalty is in the same ballpark as the one incurred when signing, but still noticeably different. This can be accounted for by the way in which communication and computation can be interleaved in the two procedures: for verification, the windows in which

communication can be performed are much smaller, making it more difficult to schedule the computation and communication efficiently.

5 The Cost of Eliminating the State

As has been discussed earlier, being able to compute digital signatures in a stateless configuration has definite advantages over a scheme that has to maintain a state. The advantages are generally focused around practical applicability. In order to be able to get rid of the state, however, SPHINCS pays a significant price. In this section, we will review just how much it costs to get rid of the state. We do this by comparing the performance of SPHINCS on the Cortex M3 to that of Multi Tree XMSS [16] on the same platform, configured in such a way that both schemes offer a similar security level using similar primitives.

5.1 XMSS^{MT}

XMSS^{MT} [16] uses the XMSS construction [5] in such a way that it is possible to sign a much larger number of messages before having to generate a new key. Depending on the specific parameters and practical application, this limit is virtually non-existent.

In essence, XMSS (and, by extension, XMSS^{MT}) is the stateful counterpart of SPHINCS. The high-level design is very similar, using WOTS+ leaf nodes to sign messages and including the authentication path in the signature, as well as adding bitmasks to the hash tree layers. The differences originate from the fact that XMSS uses the leaf nodes sequentially to guarantee that they are only used once, while SPHINCS selects them at random with a negligible chance of duplication. As we have discussed earlier, SPHINCS reduces this chance by adding a layer of HORST nodes underneath the WOTS+ leaves and by greatly increasing the tree size. In order to feasibly operate on such a large tree, SPHINCS includes layers of WOTS+ signatures to link different subtrees together. These linked subtrees are precisely how XMSS^{MT} is also able to increase its tree size (and thus the number of available leaf nodes).

Besides being able to work with a smaller, more efficient tree, going through the leaf nodes sequentially also allows us to re-use parts of the previous authentication path when generating a new signature. By storing the authentication path and the WOTS+ signature, only very few new nodes need to be computed when the next signature is generated. The amount of work that needs to be done to update the authentication path varies wildly for the different leaf nodes, however, making the signing cost very diverse. In order to be able to efficiently compute each signature at the same costs, the authors of XMSS^{MT} suggest the use of the BDS traversal algorithm [7] with a distributed signature generation method[5].

[5] Where the costs for signature generation are equally distributed among all signature generations.

BDS Traversal. While this is not the right place to go into the precise details of the BDS algorithm [7], it is relevant and necessary to have a basic intuition. The goal of this algorithm is to have all the nodes for a certain authentication path available right when it is required, while still keeping the storage requirement to a minimum. This is done by maintaining an elaborate state and allocating 'updates' to each round (i.e. to be performed whenever an authentication path is returned). The state consists, among other structures, of instances of the treehash algorithm progressing through the current subtree, but also of work-in-progress instances of the next subtree, for each layer in the hypertree. The allocated updates are assigned to the treehash instances that have the most work to do relative to their deadline, guaranteeing that each node is produced when needed. Additionally, the algorithm configuration allows for a trade-off between the amount of work per update and the storage requirement by caching particularly expensive nodes high up in the trees.

For now, the main detail we need to make note of is the fact that in order to initialize the state and initial authentication path, it is necessary to compute the first full subtree on every layer. Additionally, it is relevant to remark that changing from one subtree to the next is a more costly operation, as this requires a new WOTS+ signature to effectively link the new tree to the existing parent tree.

5.2 Parameters

XMSS^{MT} offers a diverse set of parameters to make different trade-offs between the runtime and storage requirements. For the parameters selection, we tried to conform to the settings proposed in the XMSS^{MT} Internet-Draft [15]. This let to the choice of $m = 32$ and $n = 32$ for the function output sizes, a tree with a total height of $h = 20$, $d = 2$ subtree layers and a Winternitz parameter $w = 16$ (resulting in a length of $\ell = 67$).

In terms of running time, the performance would have benefited significantly from a larger number of subtree layers, d. However, each layer d implies the need to store an additional WOTS+ signature, quickly exceeding our memory constraint. Moreover, a signature contains one WOTS+ signature per layer, increasing the signature size significantly. For the BDS algorithm, we choose $k = 6$. This allows us to cache a fairly large number of expensive nodes in the limited memory that is available.

In order to be able fairly compare XMSS^{MT} to SPHINCS-256, we do not use SHA-256 and SHA-512 to compute the message digest or the parent nodes in the hash trees. Instead, we rely on the BLAKE hash functions [1] for the message digest, and use a construction based on the ChaCha permutation similar to the ones described in Sect. 2.9 for the functions H and F, listed below. As in SPHINCS, let C = "expand 32-byte to 64-byte state!", let $Chop(M, i)$ be a function that truncates M to i bits, π the ChaCha permutation, M_i strings of 256 bits and O a string of 256 zero-bits, then:

$$F(K, M) = Chop(\pi(\pi(K\|C) \oplus (M\|O)), 256)$$
$$H(K, M_1, M_2) = Chop(\pi(\pi(\pi(K\|C) \oplus (M_1\|O)) \oplus (M_2\|O)), 256)$$

For pseudo-random number generation, we replace ChaCha20 with ChaCha12, as this matches the choice for SPHINCS-256. All of this implies that we can use the same ARMv7-M assembly implementation of the ChaCha permutation that we used for SPHINCS.

5.3 Performance

The difficulty with an accurate performance estimate for XMSS^{MT} is that it highly depends on the practicalities of the platform it is deployed on, as well as the precise use-case. This is a result of the extra administration that comes with dealing with the state. As suggested above, part of the state is crucial for the security of the scheme (namely the index of the last processed leaf node), and while the structures that need to be stored for BDS traversal are needed for signing time optimization purposes. Writing persistent data is a relatively costly operation on most platforms, so different decisions will need to be made depending on use case specific requirements. On the STM32L100C, writing a well-aligned 4-byte word to non-volatile memory costs roughly 216 500 cycles on average, and scales linearly with the number of words written.

For our experiments, we assume that the device is powered on for a longer period of time, and is being queried for multiple signatures over this interval. This is an especially relevant scenario for XMSS^{MT}, as this is where the benefit of the BDS state comes into play most prominently.

Before outputting each new signature, it is necessary to write the updated secret key to persistent memory. This prevents re-use of a leaf node (and thus compromise of the key) when the power gets cut. As the BDS state is much larger and thus more expensive to store, it is only written to persistent memory when a graceful power-off occurs. In case this state is lost, it can be reinitialized based on the secret key seed and leaf node index. For the purpose of this comparison, this is considered out of scope.

Key Generation and Initialization. Compared to SPHINCS, the key generation phase for XMSS^{MT} is much more expensive, especially in the setting described here. The main reason for this is the fact that the two trees consist of 10 levels each, resulting in the computation of 2048 WOTS+ leaves (1024 on each level). As mentioned above, the generation of two trees is necessary to initialize the BDS state. Additionally, a WOTS+ signature needs to be computed for the bottom tree. For the specified parameters, the initialization phase takes 10 590 816 803 cycles. Each WOTS+ leaf computation costs 5 160 791 cycles, and the WOTS+ signature costs 1 922 418 cycles. This accounts for most of the work, leaving only a small fraction for the hash trees.

Signing and Verification. For signing, the cycle count is not precisely identical for each signature. The BDS algorithm tries to distribute costs equally among signature generations by running a fixed amount of treehash 'updates' for each

signature. However, for the first few signatures not all these updates are needed as all structures are initialized during key generation and only few values have to be computed during each signature generation. It turns out that during this "start-up phase" it is slightly more costly to update the state for 'right' leaf nodes than for their 'left' neighbors, signatures using a left leaf node come in at 25 289 355 cycles, while right nodes cost 20 135 696 cycles. Transitioning from one tree to the next does cost significantly more cycles than a regular signature: Signatures that require renewing the WOTS+ signature that binds the subtrees together cost 33 003 958 cycles. Overall, the average signing time is 22 725 092 cycles.

As one would expect of a hash-based signature scheme, verification is a much cheaper operation. At only 5 528 712 cycles, the relative gain in comparison to SPHINCS is not as dramatic as it is for the signing procedure, but it is still a significant difference.

6 Conclusions

Having to maintain a state for digital signature schemes can have several negative consequences. With this work, we have shown that it is feasible to run stateless hash-based signatures on a microcontroller, both in terms of performance and memory use. The fact that a SPHINCS signature itself does not fit in memory has proven to be a surmountable obstacle. This could make SPHINCS-256 well-suited as a cross-platform post-quantum signature scheme, especially when keeping a state is not a viable option.

However, we have also shown that eliminating the state does not come for free. While verification is fast for stateful and stateless schemes, signing with stateful XMSS is about 32 times faster than signing with stateless SPHINCS. The 729 942 616 cycles used by SPHINCs may be acceptable for non-interactive applications that need long term security and cannot maintain a state, but we believe that further algorithmic improvements to stateless hash-based signatures will be needed to enable deployment on a broader scale.

References

1. Aumasson, J.-P., Henzen, L., Meier, W., Phan, R.C.-W.: SHA-3 proposal BLAKE. Submission to NIST (2008). https://131002.net/blake/blake.pdf
2. Bellare, M., Rogaway, P.: Collision-resistant hashing: towards making UOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997). https://cseweb.ucsd.edu/~mihir/papers/tcr-hash.pdf
3. Bernstein, D.J.: ChaCha, a variant of Salsa20. SASC 2008: the state of the art of stream ciphers, Document ID: 4027b5256e17b9796842e6d0f68b0b5e (2008) http://cr.yp.to/papers.html#chacha
4. Bernstein, D.J., et al.: SPHINCS: practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 368–397. Springer, Heidelberg (2015). http://cryptojedi.org/papers/#sphincs

5. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011). https://huelsing.files.wordpress.com/2013/05/mssgesamt.pdf
6. Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., Vuillaume, C.: Merkle signatures with virtually unlimited signature capacity. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 31–45. Springer, Heidelberg (2007)
7. Buchmann, J., Dahmen, E., Schneider, M.: Merkle tree traversal revisited. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 63–78. Springer, Heidelberg (2008). https://www.cdc.informatik.tu-darmstadt.de/reports/reports/AuthPath.pdf
8. Hülsing, A., Busold, C., Buchmann, J.: Forward secure signatures on smart cards. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 66–80. Springer, Heidelberg (2013). https://huelsing.files.wordpress.com/2013/05/xmss-smart.pdf
9. Dahmen, E., Okeya, K., Takagi, T., Vuillaume, C.: Digital signatures out of second-preimage resistant hash functions. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 109–123. Springer, Heidelberg (2008). https://www.cdc.informatik.tu-darmstadt.de/~dahmen/papers/DOTV08.pdf
10. Eisenbarth, T., von Maurich, I., Ye, X.: Faster hash-based signatures with bounded leakage. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 223–244. Springer, Heidelberg (2014). http://users.wpi.edu/~teisenbarth/pdf/SignatureswithBoundedLeakageSAC.pdf
11. Goldreich, O.: Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 104–110. Springer, Heidelberg (1987). http://theory.csail.mit.edu/ftp-data/pub/people/oded/gmr.ps
12. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012). https://www.sha.rub.de/media/sh/veroeffentlichungen/2014/06/12/lattice_signature.pdf
13. Hülsing, A.: Practical Forward Secure Signatures Using Minimal Security Assumptions. Ph.D. thesis, TU Darmstadt (2013). http://tuprints.ulb.tu-darmstadt.de/3651/
14. Hülsing, A.: W-OTS+ – shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 173–188. Springer, Heidelberg (2013). https://huelsing.files.wordpress.com/2013/05/wotsspr.pdf
15. Hülsing, A., Butin, D., Gazdag, S., Mohaisen, A.: XMSS: extended hash-based signatures draft-irtf-cfrg-xmss-hash-based-signatures-01. Crypto Forum Research Group Internet-Draft (2015). https://tools.ietf.org/html/draft-irtf-cfrg-xmss-hash-based-signatures-01
16. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for XMSS^{MT}. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES Workshops 2013. LNCS, vol. 8128, pp. 194–208. Springer, Heidelberg (2013). https://huelsing.files.wordpress.com/2013/04/xmss-optimal.pdf
17. Lamport, L.: Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (1979)
18. ARM Limited. ARMv6-M Architecture Reference Manual. Document ID: ARM DDI0419C. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0419c/
19. ARM Limited. ARMv7-M Architecture Reference Manual. Document ID: ARM DDI0403E.B. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0403e.b/

20. ARM Limited. Cortex-m0 processor – ARM. http://www.arm.com/products/processors/cortex-m/cortex-m0.php
21. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990). http://www.merkle.com/papers/Certified1979.pdf
22. Oder, T., Pöppelmann, T., Güneysu, T.: Beyond ECDSA and RSA: lattice-based digital signatures on constrained devices. In: Design Automation Conference – DAC 2014, pp. 1–6. ACM (2014). https://www.sha.rub.de/media/attachments/files/2014/06/bliss_arm.pdf
23. Reyzin, L., Reyzin, N.: Better than BiBa: short one-time signatures with fast signing and verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 144–153. Springer, Heidelberg (2002). http://www.cs.bu.edu/~reyzin/papers/one-time-sigs.pdf
24. Rohde, S., Eisenbarth, T., Dahmen, E., Buchmann, J., Paar, C.: Fast hash-based signatures on constrained devices. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 104–117. Springer, Heidelberg (2008). https://www-old.cdc.informatik.tu-darmstadt.de/reports/reports/REDBP08.pdf
25. Yang, B.-Y., Cheng, C.-M., Chen, B.-R., Chen, J.-M.: Implementing minimized multivariate PKC on low-resource embedded systems. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 73–88. Springer, Heidelberg (2006). http://precision.moscito.org/by-publ/recent/39340073.pdf

Author Index