# Lossy trapdoor primitives, zero-knowledge proofs and applications

Chen Qian

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

# THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
*Mathématiques et Sciences et Technologies de l'Information et de la Communication*
Spécialité : *Informatique*

Par

**Chen QIAN**

# Lossy trapdoor primitives, zero-knowledge proofs and applications.

**Thèse présentée et soutenue à Rennes, le 4 octobre 2019**
**Unité de recherche : IRISA**
**Thèse N° :**

**Rapporteurs avant soutenance :**

| | |
|---|---|
| Javier HERRANZ | Senior Lecturer Université Polytechnique de Catalogne, Barcelone, Espagne |
| Damien VERGNAUD | Professeur Sorbonne Université, Paris, France |

**Composition du Jury :**

| | | |
|---|---|---|
| Examinateurs : | Adeline ROUX-LANGLOIS | Chargée de Recherche CNRS IRISA, France, encadrante |
| | Michel ABDALLA | Directeur de Recherche, Ecole normale supérieure, France |
| | Olivier PEREIRA | Professeur, Université Catholique de Louvain-la-Neuve, Belgique |
| | Carla RAFOLS | Postdoc, Université Pompeu Fabra, Espagne |
| Dir. de thèse : | Pierre-Alain FOUQUE | Professeur Université de Rennes 1, France |
| | Benoît LIBERT | Directeur de Recherche CNRS, Ecole normale supérieure de Lyon, France, encadrant |

# Remerciements

En premier lieu, je remercie mes encadrants de thèse Pierre-Alain Fouque, Benoît Libert et Adeline Roux-Langlois. Merci de m'avoir accompagné pendant ces trois années de thèse. Pierre-Alain, merci de m'introduire le monde de la cryptographie en master et puis de m'encadrer en thèse, merci pour ton enthousiasme, tes disponibilité, tes vaste connaissance scientifique, sans toi, je n'aurais sans doute pas la chance de découvrir autant de possibilité dans la recherche de la cryptographie. Benoît, merci de ta disponibilité quand je visite Lyon et aussi avec les mails, j'ai particulièrement apprécié ton esprit scientifique, les critiques vraiment utiles et des idées magnifiques dans la recherche. J'ai beaucoup profité de ta vision de la recherche et ton goût scientifique, tout cela me sert comme un exemplaire idéal dans mon future. Adeline, merci pour tes disponibilité, merci pour m'accompagné pas à pas, pour les soumissions, les préparation des exposées, et aussi les discussions régulières vraiment constructives qui m'a donné beaucoup d'idées. Merci à vous trois pour votre encadrement, qui m'a beaucoup appris, non seulement dans le monde scientifique mais aussi dans la vie quotidienne.

I would also like to thank Javier Herranz and Damien Vergnaud, who accepted to review this thesis during summer in a relatively short time, and in general for their interest in my work. I also thank the rest the of the committee: Michel Abdalla, Olivier Pereira and Carla Ràfols.

I deeply thank my co-authors for your ideas and discussions: Benoît Libert, Pierre-Alain Fouque, Adeline Roux-Langlois, Thomas Peters, Alain Passelègue.

Merci beaucoup à tous les membres de l'équipe EMSEC à Rennes, Pauline et Guillaume, qui ont toléré de partager le bureau avec moi et m'ont supporté pendant ces trois ans. Baptiste et Claire, qui mont partagé des idées des cryptographie symmétrique et de l'algèbre linéaire. Weiqiang , merci pour le temps que tu as dépensé de discuter avec moi. Alban l'expert de la statistique qui me montre souvent une autre vision des choses. Et aussi pendant les pauses de café les humours de Florant et Angèle. Alexandre, avec qui j'ai gagné le champion de badminton de l'équipe. Je n'oublie évidemment pas les autre doctorants et post-docs que j'ai eu la chance de rencontré dans l'èquipe: Raphael, Pierre (Karpman et Lestringant), Adela, Yang, Wojciech, Loïc, Céline, Adina, Paul, Katharina, Solène, Victor, Gautier, Oliver, Guillaume, Joshua et Thomas.

Et je remercie aux personnes que j'ai renconté lorsque mes passages à Lyon surtout les personnes de la partie cryptographique de l'équipe Aric de ENS Lyon. Merci à Fabrice qui est mon frère scientifique, et également Alonso, Changmin, Adel, Alice, Miruna, Radu, Ida, Shi, Sanjay, Chitchanok, Junqing, Gottfried, Elena, Jiangtao, Alexandre, Somindu.

Je ne peux pas terminer ces remerciments sans penser à ma famille, mes parents qui m'a accompagné depuis mon enfance et qui m'a donné l'envie de de faire la recherche en informatique avec les programmation de Basic quand j'étais petit.

Enfin, merci Siyu pour ta patiance avec moi, pour m'avoir supporté quand je suis en mauvais humeur. Tu es pour beaucoup dans cette thèse.

# Résumé en Français

Au cours des vingt dernières années, avec le développement rapide d'objets connectés et le large déploiement d'Internet à haut débit dans notre vie quotidienne, une énorme quantité d'informations est transférée par les réseaux, y compris les secrets commerciaux et les données privées. Par conséquent, la sécurité de la communication ne concerne pas seulement les domaines militaires mais aussi les activités commerciales et notre vie privé.

La cryptologie est précisément la science qui étudie la sécurité des communications. Par définition, donnée dans le Handbook of Applied Cryptography [MOV96, Page 15], la cryptologie est l'étude de la cryptographie et de la cryptanalyse. La différence entre ces deux directions est que la cryptographie est l'étude des outils mathématiques liés à la confidentialité, l'authenticité et l'intégrité de la communication. La cryptanalyse, en revanche, vise à attaquer les propriétés ci-dessus et trouver les défaillances de la sécurité avant les vrais adversaires.

Du côté des fonctionnalités, depuis la dernière décennie, la vitesse d'Internet a été considérablement améliorée, très récemment la 5G (la technologie de réseau cellulaire de cinquième génération) porte la vitesse de téléchargement jusqu'à 10 gigabits par seconde en théorie. Les services *cloud*, y compris les calculs et le stockage, deviennent très populaire dans notre vie. L'idée est de télécharger nos données dans un serveur *cloud*, nous pouvons ensuite les récupérer quand nous voulons, et nous pouvons même effectuer des calculs complexes avec ces données sur le *cloud*. Grâce à cette approche, nous ne sommes plus limités par l'espace de stockage local et les ressources de calcul. Cependant, ce nouveau mode de vie nous pose de nombreux défis dans la protection de notre vie privée. Par exemple, l'exécution d'opérations sur les données privées, même les recherches qui sont les requêtes les plus simples, semble être en contradiction avec la confidentialité des données. En utilisant le système de chiffrement cherchable, nous pouvons atteindre cet objectif. En outre, Gentry a donné la première construction d'un système de chiffrement complètement homomorphe en 2009 [Gen09], qui nous permet d'effectuer n'importe quelle opération sur le *cloud* avec une quantité minimale de fuite d'informations.

Aujourd'hui, en réalisant des fonctionnalités de plus en plus complexes telles que les signatures d'anneaux et les systèmes de vote. Les nouvelles conceptions de cryptosystèmes deviennent de plus en plus complexes. Par conséquent, il est également plus difficile de protéger le système contre toutes sortes d'attaques. Pour résoudre ce problème, nous introduisons la preuve de sécurité, qui tente de modéliser tous les types possibles d'adversaires (par exemple, IND-CPA (Indistinguishability Chosen-Plaintext Attack) ou IND-CCA (Indistinguishability Chosen-Ciphertext Attack) pour les chiffrements), puis formellement

prouver que ces attaquants ne peuvent pas briser les propriétés de sécurité du système.

## 1. Sécurité prouvable et hypothèses cryptographiques

Une approche commune pour fournir une preuve de sécurité consiste à montrer que si un attaquant peut attaquer la construction, alors il peut résoudre certains problèmes durs en mathématiques et bien étudiés, tels que le problème RSA dû à Rivest, Shamir and Adleman [RSA78], Decisional Diffie-Hellman (DDH) [DH76], Decisional Composite Residuosity (DCR) [Pai99] ou Learning With Errors (LWE) [Reg05]. Certaines de ces hypothèses nous amènent à des primitives cryptographiques très efficaces qui fonctionnent même sur les plates-formes avec des ressources minimales telles que la carte de crédit ou le passeport électronique. Depuis quelques années, il y a eu d'une part la percée algorithmique quantique de Shor [Sho94] qui peut résoudre le problème du logarithme discret avec ordinateur quantique. D'autre part, il y a eu un développement rapide récent d'ordinateurs quantiques, y compris le système D-Wave (annoncé en 2011), IBM Q (annoncé par IBM en 2017) et beaucoup d'autres réalisations de l'ordinateur quantique. Ainsi, beaucoup de problèmes présumés durs (y compris RSA, DCR, DDH ...) deviennent faciles avec l'ordinateur quantique et seuls quelques problèmes restent difficiles (par exemple LWE) dans le monde post-quantique. Cependant, en utilisant ces hypothèses post-quantiques, les constructions sont souvent moins efficaces et ne sont pas tout à fait utilisable dans la pratique pour le moment. Ainsi, les recherches sur l'amélioration de l'efficacité des primitives cryptographiques classiques et des primitives post-quantiques sont très importante.

DDH, RSA et DCR. Les hypothèses cryptographiques classiques telles que DDH, RSA et DCR impliquent que les problèmes correspondants sont durs en mathématiques et bien étudiés. Depuis leur introduction, elles sont devenues des hypothèses essentielles dans les constructions cryptographiques.

DDH - Etant donné un groupe cyclique $\mathbb{G}$ d'ordre premier $p$, il est calculatoirement difficile de distinguer les distributions $g, g^a, g^b, g^{ab}$ et $g, g^a, g^b, g^c$, où $g$ est un élément dans $\mathbb{G}$ et $a, b, c$ sont des nombres aléatoires dans $\mathbb{Z}_p$.

RSA - Etant donné deux grands nombres premiers $p, q$, soit $N = pq$, et un entier $e$ tel que $2 < e < N$, tel que $e$ co-premier avec $\phi(N)$, et $C$ un entier tel que $0 \leq C < N$, il est calculatoirement difficile de calculer $M$ tel que $C = M^e \mod N$.

DCR - Etant donné un nombre composé $n$, il est calculatoirement difficile de décider si $z$ est un nombre aléatoire ou un $n$-résidu modulo $n^2$ (i.e. déterminer s'il existe un $y$ tel que $z = y^n \mod n^2$).

Le principal avantage de l'utilisation de ces trois hypothèses est leur efficacité. Par conséquent, dans cette thèse, nous donnons des constructions plus efficaces basées sur ces trois hypothèses de schémas cryptographiques existants comme les signatures d'anneau et les chiffrements KDM-CCA (Key-Dependent-Message Chosen-Ciphertext-Attack).

LWE. Dans cette thèse, nous étudions également les hypothèses post-quantiques. Au cours de la dernière décennie, avec l'investissement financier pour construire l'ordinateur quantique, il semble nécessaire de construire de nouvelles primitives cryptographiques

dans le monde post-quantique. Les hypothèses liées aux réseaux Euclidiens semblent être les plus prometteuses et fournir le plus de fonctionnalités pour le moment.

Formellement, un réseau Euclidien est un sous-groupe discret de $\mathbb{R}^n$. Apprendre-avec-erreur (Learning-With-Errors ou LWE) est un problème difficile dans les réseaux Euclidiens que nous avons utilisé dans les constructions cryptographiques basées sur les réseaux. Etant donné d'un nombre premier $q$, deux entiers $m, n \in \mathbb{N}$ et une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, le problème LWE indique qu'il est calculatoirement difficile de décider si un vecteur $\mathbf{b} \in \mathbb{Z}^n$ est de la forme de $\mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod q$ , où $\mathbf{s}$ et $\mathbf{e}$ sont deux vecteurs de $\mathbb{Z}_q^m$ tirés selon une distribution Gaussienne discrète. Du point de vue du réseau Euclidien, nous pouvons voir l'hypothèse LWE comme il est difficile de déterminer si un point est assez proche d'un point du réseau à condition que l'écart-type de la Gaussienne soit petit devant le plus petit écart de points du réseau.

Sécurité étroite. Lorsque nous parlons de l'efficacité des primitives cryptographiques, nous parlons généralement du nombre d'éléments dans la structure algébrique sous-jacente (par exemple groupe $\mathbb{G}$ dans l'hypothèse DDH ou $\mathbb{Z}_q$ dans l'hypothèse LWE). Cependant, la taille de la structure sous-jacente est également essentielle. Habituellement, les preuves de sécurité transforment l'adversaire en un attaquant contre un problème difficile spécifique dans la structure algébrique sous-jacente avec certains facteurs. Ainsi, un facteur de réduction plus petit nous conduit à des éléments plus petits, donc des primitives cryptographiques plus efficaces. Si ce facteur devient constant, nous appelons une telle primitive étroitement sécurisée.

Pour parvenir à un monde protégé par des méthodes prouvés. Depuis plus de 20 ans, la cryptographie ne se contente pas seulement de chiffrer et déchiffrer. De plus en plus de fonctionnalités qui ont été créées, des chiffrements aux signatures en passant par les preuves de connaissance à divulgation nulle de connaissance, et les chiffrements entièrement homomorphes. Certains protocoles sont exclusifs ou plus efficaces sous différentes hypothèses. Savoir si certains schémas sont possible à construire restent encore un problème ouvert. Pour construire un protocole cryptographique fonctionnel complexe, la manière fiable est de les construire en combinant des primitives plus petites prouvées. Ainsi, la compacité des blocs de construction a un impact énorme sur l'efficacité de l'ensemble du protocole. Dans cette thèse, nous nous intéressons à la fois à la construction de nouvelles fonctionnalités et à l'amélioration de l'efficacité des systèmes existants.

## 2. Contributions : Les primitives *Lossy Trapdoor*

*Lossy Trapdoor Function* (LTF) est une notion relativement nouvelle introduite par Peikert et Waters [PW08], qui généralise l'idée de trappes à sens unique. Depuis lors, cette notion a été trouvée utile pour la construction de primitives cryptographiques plus complexes. Une *Lossy Trapdoor Function* est composée de deux fonctions, une injective et une autre à perte. La propriété de sécurité d'une telle primitive est que pour tout adversaire calculatoirement borné, la fonction injective est indistinguable de la fonction *lossy*. Cette stratégie capture les besoins de nombreuses constructions cryptographiques. Par exemple, pour les systèmes

de chiffrement, l'utilisateur peut déchiffrer le texte chiffré en utilisant le mode injectif et la clé d'inversion, mais dans la preuve de sécurité, nous pouvons passer en mode *lossy*, qui ne peut pas être remarqué par l'adversaire. Enfin, le mode *lossy* cache statistiquement le message de l'adversaire.

Nous définissons maintenant une notion dérivée de la LTF et décrivons les contributions.

### 2.1. Filtre algébrique à perte (LAF)

Le filtre algébrique à perte (LAF) est une variante de la LTF. Au lieu d'avoir seulement deux fonctions, le LAF est une famille de fonctions qui sont soit *lossy*, soit injectives. Chaque fonction est associée à une étiquette, qui détermine s'il s'agit d'une fonction *lossy* ou d'une fonction injective. Dans l'espace des étiquettes, il n'y a qu'un petit nombre d'étiquettes *lossy*, et toutes les autres sont injectives. Un LAF est associé à un espace d'étiquettes $\mathcal{T} = \mathcal{T}_a \cup \mathcal{T}_c$ qui est séparé en deux ensembles $\mathcal{T}_a$ étiquettes auxiliaires et $\mathcal{T}_c$ étiquettes de base. LAF doit aussi vérifie les propriétés suivantes :

- En mode injective, la fonction n'est pas nécessairement invertible, mais elle doit être injective.
- L'information divulguée par le mode *lossy* est préfixée par la clé d'évaluation et le message. A savoir, même lorsque la sortie de la fonction *lossy* est indépendante de l'étiquette $t$.

Dans cette thèse, nous donnons une construction plus efficace de filtre algébrique à perte, la taille de l'étiquette de notre construction est réduite de $\Theta(n^2)$ à $O(n)$ éléments de groupe où l'espace d'entrée est $\mathbb{Z}_p^n$. En tant qu'applications, en utilisant notre nouveau LAF, nous pouvons améliorer la construction des systèmes de chiffrement sécurisé KDM-CCA de Hofheinz [Hof13] et les extracteurs flous proposés par Wen et Liu [WL18]. Ce travail a été accepté à PKC2019 et est présenté dans le chapitre 3.

## 3. Contributions : Systèmes de preuve à divulgation nulle de connaissance et applications

La deuxième partie de cette thèse vise à développer de nouvelles primitives cryptographiques et à améliorer l'efficacité des protocoles existants avec des systèmes de preuve à divulgation nulle de connaissance.

La preuve à divulgation nulle de connaissance est une très puissante primitive cryptographique. Elle permet au prouveur de convaincre le vérifieur sur une déclaration sans divulguer d'informations sur le témoin. Formellement, la preuve à divulgation nulle de connaissance exige l'exhaustivité, la significativité et les propriétés de divulgation nulle de connaissance.

### 3.1. Signatures d'anneau de taille logarithmique étroitement sécurisées

Les signatures d'anneau sont introduites par Rivest, Shamir et Tauman [RST01]. Elles permettent au signataire de signer un message anonymement tout en convainquant les autres qu'il appartient à un certain anneau d'utilisateurs. Contrairement à la signature de groupe, la signature d'anneau n'a pas la phase de registre ou d'autorité de traçage. Malgré 15 ans de recherche, la plupart des constructions ont une signature de taille linéaire, mais deux signatures récentes donnent une signature anneau de taille logarithmique [GK15 ; LLNW16], elles utilisent toutes les deux forking lemma, par conséquent, elles souffrent toutes les deux de la perte de sécurité.

Dans cette thèse, nous contrevenons cette limitation en fournissant une signature d'anneau de taille logarithmique étroitement sécurisée dans le modèle de l'oracle aléatoire. Nous combinons le Groth-Kohlweiss $\Sigma$-protocol [GK15] avec les schémas de chiffrement en mode dual. Ce travail est accepté à ESORICS2018 et présenté dans le chapitre 4.

### 3.2. Preuves à divulgation nulle de connaissance non-intéractive et à vérifieur désigné et système de vote

Les preuves à divulgation nulle de connaissance non-intéractives et à vérifieur désigné (DVNIZK) sont une variante des systèmes de preuve à divulgation nulle de connaissance. Dans cette variante, le vérifieur ne peut vérifier la preuve qu'à l'aide d'une clé de vérification secrète. Une telle primitive semble déjà utile dans de nombreuses applications : nous pouvons construire un système de chiffrement CCA ou un système de vote. En 2006, Fazio et Nicolosi ont proposé la construction de DVNIZK dans le modèle standard à l'aide des chiffrements homomorphes. En 2015, Chaidos et Groth ont proposé une construction efficace [CG15] à l'aide du système de chiffrement Okamoto-Uchiyama.

Dans cette thèse, nous poussons cette approche un peu plus loin en fournissant une construction reposant sur les réseaux Euclidiens de DVNIZK dans le modèle standard et nous avons aussi donné une construction d'un système de vote post-quantique dans le modèle standard. Ce travail est en soumission et presenté dans le chapitre 5.

## 4. Liste de publications

- [LQ19] Lossy Algebraic Filters with Short Tags. Benoît Libert and Chen Qian.(PKC2019)
- [LPQ18] Logarithmic-Size Ring Signatures With Tight Security from the DDH Assumption. Benoît Libert and Thomas Peters and Chen Qian.(ESORICS 2018)

Publications qui ne sont pas dans cette thèse

- [QTG18] Universal Witness Signatures. Chen Qian and Mehdi Tibouchi and Rémi Géraud. (IWSEC 2018)

- [LPQ17] Structure-Preserving Chosen-Ciphertext Security with Shorter Verifiable Ciphertexts. Benoît Libert and Thomas Peters and Chen Qian. (PKC 2017)
- [FQ16] Fault Attacks on Efficient Pairing Implementations. Pierre-Alain Fouque and Chen Qian. (AsiaCCS 2016)
- [AFQ+14] Binary Elligator Squared. Diego F. Aranha and Pierre-Alain Fouque and Chen Qian and Mehdi Tibouchi and Jean-Christophe Zapalowicz. (SAC 2015)

# Contents

# 1 Introduction

Over the last twenty years, with the quick developments of connected objects and wide deployment of the internet, a huge amount of information is transferred through the internet, including commercial secrets and private data. Therefore, the communication security concerns not only the military area but also the commercial activities and our daily life.

The cryptology is the science studying the security of communications. By definition given in the Handbook of Applied Cryptography [MOV96, Page 15], the cryptology is the study of cryptography and cryptanalysis. The difference between these two directions is that the cryptography is the study of mathematical tools related to the confidentiality, authenticity, and integrity of the communications. The cryptanalysis, on the other hand, aims at attacking the above properties and finding security vulnerabilities before the real adversaries.

On the functionality side, from the last decade, the internet speed has been dramatically improved and very recently the 5G (the fifth generation cellular network technology) brings the download speed up to 10 gigabits per second in theory. Cloud computing and cloud storage became very popular in our life. The idea is to upload our data into a cloud server, and later we can retrieve the data when we want. We can also perform complex computations with these data on the cloud without needing to download all of them. Using this approach, we are no longer limited by local storage space and computation resources. However, this new fashion of life gives us many challenges to protect our privacy. For example, performing operations over the private data, even the simplest search queries, seems to be contradicting with the confidentiality of the data. Using the searchable encryption scheme, we can achieve the goal. Moreover, Gentry has given the first construction of a fully homomorphic encryption scheme in 2009 [Gen09], which allows to perform any operation on the cloud with a minimum amount of information leakage.

Nowadays, we can achieve more and more complex functionalities such as ring signatures and voting schemes and new cryptosystem designs become more and more complex. Therefore it is even harder to protect the system against all kinds of attacks. To solve this issue, cryptographers introduced the security proof, which try to model every possible type of adversary (for example, IND-CPA (Indistinguishability Chosen-Plaintext Attack) or IND-CCA (Indistinguishability Chosen-Ciphertext Attack) adversary for encryption schemes), then formally prove that these attackers cannot break the security properties of the system.

## 1.1. Provable security and cryptographic assumptions

A common approach in proving the security is to reduce the adversary against the primitive into a solver of some classical well-studied hard mathematical problems such as the RSA problem proposed by Rivest, Shamir and Adleman (RSA) [RSA78], Decisional Diffie-Hellman (DDH) [DH76], Decisional Composite Residuosity (DCR) [Pai99], and Learning With Errors (LWE) [Reg05]. Some of these assumptions lead us to very efficient cryptographic primitives that work even on platforms with minimal resources such as credit card or e-passport. With the quantum algorithmic breakthrough by Shor [Sho94], and the recent quick development of quantum computers including D-Wave system (annonced in 2011), IBM Q (annonced by IBM in 2017) and many more other realisations of quantum computer, many of these assumed hard problems (including RSA, DCR, DDH ...) are becoming easy for quantum computers and only a few problems remain hard (for example LWE) in the post-quantum world. However, using those post-quantum assumptions, we can only achieve less efficient cryptographic primitives which are not entirely practical for the moment. Thus, the research both on improving the efficiency of classical cryptographic primitives and post-quantum ones are very attracting.

DDH, RSA AND DCR. The classical cryptographic assumptions such as DDH, RSA, and DCR correspond to well-studied hard mathematical problems. Since their introduction, they become essential assumptions to construct cryptographic primitives.

DDH - Given a cyclic group $\mathbb{G}$ of prime order $p$, it is computationally hard to distinguish the distributions of $g, g^a, g^b, g^{ab}$ and of $g, g^a, g^b, g^c$, where $g \in \mathbb{G}$ and $a, b, c$ are random elements in $\mathbb{Z}_p$.

RSA - Given two large primes $p, q$, let $N = pq$, and $e$ an integer that $2 < e < N$, that $e$ be coprime with $\phi(N)$, and $C$ an integer that $0 \leq C < N$, it is computationally hard to compute $M$ such that $C = M^e \mod N$.

DCR - Given a composite number $n$, it is computationally hard to decide whether $z$ is a random number modulo $n^2$ or a $n$-residue modulo $n^2$ (i.e. whether there exists a $y$ such that $z = y^n \mod n^2$).

The main advantage of using these three assumptions is their efficiency. Therefore, in this thesis, we give more efficient constructions based on these three assumptions of existing cryptographic primitives such as selective opening CCA secure encryption schemes, ring signatures, KDM-CCA secure encryptions, and deterministic encryptions.

LWE. In this thesis, we also use post-quantum assumptions to construct cryptosystems. During the last decade, with the financial investment of quantum computer, it seems necessary to build new cryptographic primitives in the post-quantum world. The assumptions based on lattice problems seem to be the most promising in term of security and efficiency and provide the most functionalities for the moment.

Formally, a lattice is a discrete subgroup of $\mathbb{R}^n$. Learning-With-Error is a hard problem over lattices we used to construct lattice-based cryptography. The LWE problem states that it is computationally hard to decide whether a vector $\mathbf{b}$ equals to $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ or a complete random vector, where $\mathbf{A}$ is a random matrix and both $\mathbf{s}$ and $\mathbf{e}$ are two vectors of $\mathbb{Z}_q$ drew

from a discrete Gaussian distribution. From a lattice point of view, we can see the LWE assumption as it is hard to determine if a point is close enough to a lattice point.



Tight security. When we talk about the efficiency of the cryptographic primitives, we are usually considering about the number of elements in the underlying algebraic structure used in the primitive. However, the size of the underlying structure (for example, $\mathbb{G}$ in case of DDH or $\mathbb{Z}_q$ in case of LWE) is also essential. Since the security proofs transform the adversary into an attacker against a specific hard problem in the underlying algebraic structure upon some factors, smaller reduction factor leads to smaller elements and therefore more efficient cryptographic primitives. If this factor becomes constant, we call such a primitive tightly secure.

Towards a provable protected world. During the last 20 years, the cryptography deals with more and more features beyond encryptions such as: signatures to zero-knowledge proofs, fully homomorphic encryptions. Some of these protocols are only constructed in some assumptions or more efficient in certain ones. There are still some possibility results remaining as open problems.

To construct a complex functional cryptographic protocol, the reliable way is to construct them by combining the smaller proven secure primitives. Thus, the compactness of building blocks has a huge impact on the efficiency of the entire protocol. In this thesis, we are interested in providing new functionalities and improving the efficiency of existing schemes.

## 1.2. My contributions: Lossy trapdoor primitives

A lossy trapdoor function (LTF) is a relatively new notion introduced by Peikert and Waters [PW08]. It generalizes the idea of trapdoor one-way functions. Since then, it has been found useful for building more complex cryptographic primitives. A lossy trapdoor function is composed of two functions, an injective one and a lossy one. The security property of such a primitive is that for any computational adversary, the injective function is indistinguishable from the lossy one. This strategy captures the need in many cryptographic proofs. For example, for encryption schemes, the user can decrypt the ciphertext using the injective mode and the inversion key, but in the security proof, we can switch into the lossy mode, which can not be noticed by the adversary. Besides, the lossy mode statistically hides the message from the adversary.

We now define a notion derived from LTF and describe our contributions.

### 1.2.1. Lossy Algebraic Filter (LAF)

A Lossy Algebraic Filter (LAF) is a variant of a LTF. Instead of having only two functions, a lossy algebraic filter is a family of functions that are either lossy or injective. Each LAF is associated with a tag, which determines whether it is a lossy function or an injective function. In the tag space, there are only a few numbers of lossy tags, and all other functions are injective ones. A LAF is also associated with a tag space $\mathcal{T} = \mathcal{T}_a \cup \mathcal{T}_c$ which is split into two sets $\mathcal{T}_a$ auxiliary tags and $\mathcal{T}_c$ core tags. The LAF verifies the following properties:

- In the injective mode, the function is not necessarily invertible but still injective.
- The information leaked through the lossy mode is prefixed by the evaluation key. Namely, even when the output of the lossy function is independent of the tag $t$.

In this thesis, we give a more efficient construction of lossy algebraic filter, the tag size of our construction is reduced from $\Theta(n^2)$ down to $O(n)$ group elements where the input space is $\mathbb{Z}_p^n$. As applications, by using our new LAF, we can improve the construction of Hofheinz's KDM-CCA secure encryption schemes [Hof13] and fuzzy extractors proposed by Wen and Liu [WL18].This work was published in PKC2019 and presented in the chapter 3.

## 1.3. Contributions: zero-knowledge proof systems and its applications

The second part of this thesis aims at developing new cryptographic primitives and improving the efficiency of existing protocols with zero-knowledge proof systems.

Zero-knowledge proofs are very powerful cryptographic primitives. It allows the prover to convince the verifier on a statement without leaking information about the witness. Formally, the ZK proofs require completeness, soundness and zero-knowledge properties.

### 1.3.1. Logarithmic-size tightly-secure ring signatures

Ring Signatures are firstly introduced by Rivest, Shamir and Tauman [RST01]. It allows the signer to sign a message anonymously while convincing others that he belongs to a certain ring of users. Unlike Group Signatures, Ring Signatures do not have the registration phase or tracing authority. Despite 15 years of research, most of the constructions have linear-size signature, but two recent signatures give logarithmic size ring signature [GK15; LLNW16], they both use the Fiat-Shamir transformation, therefore, they both suffer from the security loss.

In this thesis, we bypass this limitation by providing a logarithmic-size tightly secure ring signature in the random oracle model. We combine the Groth-Kohlweiss $\Sigma$-protocol [GK15] with the dual mode encryption schemes. This work was accepted in ESORICS 2018 and presented in the chapter 4.

### 1.3.2. Designated-Verifier Zero-Knowledge proof and voting scheme

Designated-Verifier Non-Interactive Zero-Knowledge argument system (DVNIZK) is a variant of a zero-knowledge proof system. In this setting, the verifier can only verify the proof by using a secret verification key. Such setting seems already useful in many applications, to construct CCA-secure encryption scheme or voting scheme. In 2006, Damgård, Fazio and Nicolosi proposed a construction of DVNIZK in the standard model

using the homomorphic encryptions. In 2015, Chaidos and Groth have proposed an efficient construction [CG15] using Okamoto-Uchiyama encryption scheme.

In this thesis, we push this approach a little further by providing a lattice-based construction of DVNIZK in the standard model. This work is in submission and is presented in the chapter 5



## 1.4. Publication List

- [LQ19] Lossy Algebraic Filters with Short Tags. Benoît Libert and Chen Qian (PKC 2019)
- [LPQ18] Logarithmic-Size Ring Signatures With Tight Security from the DDH Assumption. Benoît Libert and Thomas Peters and Chen Qian(ESORICS 2018)

Publications Not in This Thesis

- [QTG18] Universal Witness Signatures. Chen Qian and Mehdi Tibouchi and Rémi Géraud. (IWSEC 2018)
- [LPQ17] Structure-Preserving Chosen-Ciphertext Security with Shorter Verifiable Ciphertexts. Benoît Libert and Thomas Peters and Chen Qian. (PKC 2017)
- [FQ16] Fault Attacks on Efficient Pairing Implementations. Pierre-Alain Fouque and Chen Qian. (AsiaCCS 2016)
- [AFQ+14] Binary Elligator Squared. Diego F. Aranha and Pierre-Alain Fouque and Chen Qian and Mehdi Tibouchi and Jean-Christophe Zapalowicz. (SAC 2015)

# 2 Primitives and Assumptions



## 2.1. Primitives

### 2.1.1. Lossy Trapdoor Function.

Firstly introduced by Peikert and Waters [PW08] is a generalization trapdoor function.

**Definition 2.1** ([PW08]). *Let $\lambda \in \mathbb{N}$ be a security parameter and $n : \mathbb{N} \to \mathbb{N}$, $l : \mathbb{N} \to \mathbb{N}$ be functions of $\lambda$. A collection of* $(n, l)$-lossy trapdoor functions *(LTFs) consists of PPT algorithms* $(\mathsf{InjGen}, \mathsf{LossyGen}, \mathsf{Eval}, \mathsf{Invert})$ *with the following specifications.*

***Sampling an injective function.*** *Given a security parameter $\lambda$ and an input length $n$, the randomized algorithm* $\mathsf{InjGen}(1^\lambda, 1^n)$ *outputs the index $ek$ of an injective function of the family and an inversion key $ik$.*

***Sampling a lossy function.*** *Given $\lambda$ and the input length $n$, the probabilistic algorithm* $\mathsf{LossyGen}(1^\lambda, 1^n)$ *outputs the index $ek$ of a lossy function.*

***Evaluation.*** *Given the index of a function $ek$ (produced by* InjGen *or* LossyGen*) and an input $X \in \{0,1\}^n$, algorithm* Eval *outputs $F_{ek}(X)$ such that:*

- *If $ek$ is an output of* InjGen*, then $F_{ek}(\cdot)$ is an injective function.*
- *If $ek$ was produced by* LossyGen*, then $F_{ek}(\cdot)$ has image size $2^{n-l}$. In this case, the value $n - l$ is called the* residual leakage.

*Moreover, we require the two following properties:*

***Inversion Correctness.*** *For any pair $(ek, ik) \leftarrow \mathsf{InjGen}(1^\lambda, 1^n)$ and any input $X \in \{0,1\}^n$, algorithm* Invert *returns* $\mathsf{Invert}(ik, F_{ek}(X)) = X$.

***Indistinguishability.*** *The two ensembles $\{ek \mid (ek, ik) \leftarrow \mathsf{InjGen}(1^\lambda)\}_{\lambda\in\mathbb{N}}$ and $\{ek \mid ek \leftarrow \mathsf{LossyGen}(1^\lambda)\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.*

In the construction of lossy algebraic filters in the section 3 we need the chameleon hash function.

### 2.1.2. Chameleon Hash Functions

A chameleon hash function [KR00] is a tuple of algorithms $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ which contains an algorithm $\mathsf{CMKg}$ that, given a security parameter $1^\lambda$, outputs a key pair $(hk, td) \leftarrow \mathcal{G}(1^\lambda)$. The randomized hashing algorithm outputs $y = \mathsf{CMhash}(hk, m, r)$ given the public key $hk$, a message $m$ and random coins $r \in \mathcal{R}_{hash}$. On input of messages $m, m'$, random coins $r \in \mathcal{R}_{hash}$ and the trapdoor key $td$, the switching algorithm $r' \leftarrow \mathsf{CMswitch}(td, m, r, m')$ computes $r' \in \mathcal{R}_{hash}$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$. The collision-resistance property mandates that it be infeasible to come up with pairs $(m', r') \neq (m, r)$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$ without knowing the trapdoor key $tk$. Uniformity guarantees that the distribution of hash values is independent of the message $m$: in particular, for all $hk$, and all messages $m, m'$, the distributions $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m, r)\}$ and $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m', r)\}$ are identical.

### 2.1.3. Commitment Schemes

A non-interactive commitment scheme allows a sender to commit to a message $m$ by sending a commitment string to the receiver. Later on the sender can convince the receiver that the committed value was really $m$. A commitment scheme must satisfy two security properties called *h*iding and *b*inding. The former captures that the commitment hides any partial information about the message. The latter requires that the sender be unable to open the commitment to two distinct messages. Formally, a non-interactive commitment scheme is a pair of PPT algorithms $(\mathsf{Setup}, \mathsf{Com})$. The setup algorithm $ck \leftarrow \mathsf{Setup}(1^\lambda)$ generates a commitment key $ck$, which specifies a message space $\mathcal{M}_{ck}$, a randomness space $\mathcal{R}_{ck}$ and a commitment space $\mathcal{C}_{ck}$. The commitment algorithm $\mathsf{Com}$ defines a function $\mathsf{Com}_{ck} : \mathcal{M}_{ck} \times \mathcal{R}_{ck} \to \mathcal{C}_{ck}$. On input of $m \in \mathcal{M}_{ck}$, the sender randomly chooses $r \xleftarrow{R} \mathcal{R}_{ck}$ and computes a commitment string $c = \mathsf{Com}_{ck}(m, r) \in \mathcal{C}_{ck}$.

A commitment is *p*erfectly hiding if, for any $m \in \mathcal{M}_{ck}$, the distribution $\{\mathsf{Com}_{ck}(m, r) \mid r \xleftarrow{R} \mathcal{R}_{ck}\}$ is statistically independent of $m$. It is perfectly binding if any element of the commitment space $\mathcal{C}_{ck}$ uniquely determines the message. Groth and Kohlweiss [GK15] use the following additional properties.

**Definition 2.2.** *A commitment scheme* $(\mathsf{Setup}, \mathsf{Com})$ *is* ***s****trongly binding if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\varepsilon(\lambda)$ *such that*

$$\begin{aligned} |\Pr[ck \leftarrow \mathsf{Setup}(1^\lambda); (c, m_1, r_1, m_2, r_2) \leftarrow \mathcal{A}(PK) : \\ \mathsf{Com}_{ck}(m_1; r_1) = c \ \wedge \ \mathsf{Com}_{ck}(m_2; r_2) = c \ \wedge \ (m_1, r_1) \neq (m_2, r_2)]| < \varepsilon(\lambda). \end{aligned}$$

We consider a prime $q > 2^\lambda$ specified in the commitment key $ck$. The message space and the randomness space are both $\mathcal{M}_{ck} = \mathcal{R}_{ck} = \mathbb{Z}_q$.

**Definition 2.3.** *A commitment scheme* $(\mathsf{Setup}, \mathsf{Com})$ *is* ***a****dditively homomorphic if for all messages* $m_1, m_2 \in \mathcal{M}_{ck}$ *and all random coins* $r_1, r_2 \in \mathcal{R}_{ck}$*, we have* $\mathsf{Com}_{ck}(m_1; r_1) \cdot \mathsf{Com}_{ck}(m_2; r_2) = \mathsf{Com}_{ck}(m_1 + m_2; r_1 + r_2)$.

### 2.1.4. $\Sigma$-Protocols

**Definition 2.4** ([Cra96]). *Let a prover* $\mathsf{P}$ *and a verifier* $\mathsf{V}$*, which are PPT algorithms, and a binary relation* $\mathcal{R}$*. A protocol* $(\mathsf{P}, \mathsf{V})$ *is a* $\Sigma$*-protocol w.r.t.* $\mathcal{R}$*, the challenge set* $\mathcal{C}$*, the public input* $u$ *and the private input* $w$*, if it satisfies the following:*

- ***3****-move form: The protocol is of the following form:*
  - $\mathsf{P}$ *compute commitments* $\{c_i\}_{i=0}^{j}$*, where* $j \in \mathbb{N}$*, and sends* $\{c_i\}_{i=0}^{j}$ *to* $\mathsf{V}$*.*
  - *The verifier* $\mathsf{V}$ *generates a random challenge* $x \xleftarrow{R} \mathcal{C}$ *and sends* $c$ *to* $\mathsf{P}$*.*
  - *The prover* $\mathsf{P}$ *sends a response* $s$ *to* $\mathsf{V}$*.*
  - *On input of a transcript* $(\{c_i\}_{i=0}^{j}, x, s)$*,* $\mathsf{V}$ *outputs* $1$ *or* $0$*.*
- ***C****ompleteness: If* $(u, w) \in \mathcal{R}$ *and the prover* $\mathsf{P}$ *honestly generates the transcript*

$$(\{c_i\}_{i=0}^{j}, x, s)$$

  *for a random challenge* $x \xleftarrow{R} \mathcal{C}$ *sent by* $\mathsf{V}$*, there is a negligible function* $\varepsilon(\lambda)$ *such that* $\mathsf{V}$ *accepts with probability* $1 - \varepsilon(\lambda)$*.*
- $2$***-Special soundness:*** *There exists a PPT knowledge extractor* $\mathcal{E}$ *that, for any public input* $u$*, on input of two accepting transcripts* $(\{c_i\}_{i=0}^{j}, x, s)$ *and* $(\{c_i\}_{i=0}^{j}, x', s')$ *with* $x \neq x'$*, outputs a witness* $w'$ *such that* $(u, w') \in \mathcal{R}$*.*
- ***S****pecial Honest Verifier Zero-Knowledge (SHVZK): There is a PPT simulator* $\mathcal{S}$ *that, given* $u$ *and a random* $x \in \mathcal{C}$*, outputs a simulated transcript* $(\{c_i'\}_{i=0}^{j}, x, s')$ *which is computationally indistinguishable from a real one.*

### 2.1.5. Zero-knowledge proof systems

**Definition 2.5** (Non-Interactive Zero-Knowledge proof)**.** *A non-interactive zero-knowledge proof system for a binary relation* $R = \{(x, w) \mid \mathcal{X} \times \mathcal{W}\}$ *is a pair of probabilistic polynomial time (PPT) algorithms* $(P, V)$ *such that* $P$ *takes as input* $x$ *and* $w$*, outputs a proof* $\pi$*,* $V$ *takes as input* $x$ *and* $\pi$*. outputs a bit* $b$*.*

***Completeness.*** *For all* $(x, w) \in R$*, we have* $\Pr[V(x, P(x, w)) = 1] > 1 - negl(|x|)$*.*

***Soundness.*** *For all* $x \in \mathcal{X}$ *and* $w \notin \mathcal{W}$*, for any PPT adversary* $\mathcal{A}$*,* $\Pr[V(x, mathcal A(x)) = 1] < negl(|x|)$*.*

***Zero-Knowledge.*** *There exits a PPT simulator* $S$ *such that the probability distribution* $\{x, P(x, w)\}_{(x,w\in R)}$ *and* $\{x, S(x)\}_{(x,w\in R)}$ *are computationally indistinguishable.*

In this thesis, we constructed a zero-knowledge argument system. The difference between a ZK proof system and a ZK argument system is that in argument system, the soundness is computational while in proof system the soundness is statistical.

## 2.2. Assumptions

In this section, we give different assumptions we will use during this thesis.

### 2.2.1. DDH assumptions and its variant.

The Decisional Diffie-Hellman (DDH) assumption has been proposed by Diffie and Hellman [DH76], and states that

**Definition 2.6.** *In a cyclic group $\mathbb{G}$ of prime order $q$, the **Decision Diffie-Hellman** (DDH) problem is to distinguish the distributions*

$$D_0 = \{(g, g^a, g^b, g^{ab}) \mid g \xleftarrow{R} \mathbb{G},\ a, b \xleftarrow{R} \mathbb{Z}_q\}$$

*and* $D_1 = \{(g, g^a, g^b, g^c) \mid g \xleftarrow{R} \mathbb{G},\ a, b, c \xleftarrow{R} \mathbb{Z}_q\}$.

Pairing In the construction of lossy algebraic filters in chapter 3, we need to use the pairing as a building block. A pairing is a bilinear function $e(\cdot, \cdot)$ that maps two groups $\mathbb{G}, \hat{\mathbb{G}}$ to a finite subgroup $\mathbb{G}_T$ of a finite field that must verify the following two properties:

- For all $P \in \mathbb{G}$, $Q \in \hat{G}$ and $a, b \in \mathbb{Z}$, we have $e(P^a, Q^b) = e(P, Q)^{ab}$.
- For all $P \in \mathbb{G}$, $Q \in \hat{G}$, if $e(P, Q) = 1$ then either $P = 1$ or $Q = 1$.

In the presence of pairing, we need the following variant of DDH.

**Definition 2.7.** *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be bilinear groups of order $p$. The **F**irst Decision 3-Party Diffie-Hellman (D3DH1) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT distinguisher can distinguish the distribution*

$$\begin{aligned} D_1 &:= \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, g^{abc}) \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}},\ a, b, c \xleftarrow{R} \mathbb{Z}_p\} \\ D_0 &:= \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, g^z) \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}},\ a, b, c, z \xleftarrow{R} \mathbb{Z}_p\}. \end{aligned}$$

The D3DH1 assumption has a natural analogue where the pseudorandom value lives in $\hat{\mathbb{G}}$ instead of $\mathbb{G}$.

**Definition 2.8.** *The **S**econd Decision 3-Party Diffie-Hellman (D3DH2) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT algorithm can distinguish between the distribution*

$$\begin{aligned} D_1 &:= \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^{abc}) \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}},\ a, b, c \xleftarrow{R} \mathbb{Z}_p\} \\ D_0 &:= \{(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^z) \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}},\ a, b, c, z \xleftarrow{R} \mathbb{Z}_p\}. \end{aligned}$$

We also need a computational assumption which is implied by D3DH2. The 2-3-CDH was initially introduced [KP06] in ordinary (i.e., non-pairing-friendly) discrete-logarithm hard groups. Here, we extend it to asymmetric bilinear groups.

**Definition 2.9** ([KP06]). *Let $(\mathbb{G}, \hat{\mathbb{G}})$ be a bilinear groups of order $p$ with generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. The 2-**out-of**-3 **Computational Diffie-Hellman** (2-3-CDH) assumption says that, given $(g, g^a, \hat{g}^a, g^b, \hat{g}^b)$ for randomly chosen $a, b \xleftarrow{R} \mathbb{Z}_p$, no PPT algorithm can find a pair $(g^r, g^{r \cdot ab})$ such that $r \neq 0$.*

It is known (see, e.g., [LV08]) that any algorithm for solving the 2-3-CDH problem can be used to break the D3DH2 assumption. On input of $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c, \hat{g}^z)$, where $z = abc$ or $z \in_R \mathbb{Z}_p$, the reduction can simply run a 2-3-CDH solver on input of $(g, g^a, g^b, \hat{g}^a, \hat{g}^b)$. If the solver outputs a non-trivial pair of the form $(R_1, R_2) = (g^r, g^{r \cdot ab})$, the D3DH2 distinguisher decides that $z = abc$ if and only if $e(R_1, \hat{g}^z) = e(R_2, \hat{g}^c)$.

In the construction of lossy algebraic function in chapter 3, we actually rely on a weaker variant of D3HD1, called wD3HD1, where $\hat{g}^a$ is not given. In our tightly secure construction (which requires asymmetric pairings), we need to rely on the following variant of wD3HD1.

**Definition 2.10.** *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be bilinear groups of order $p$. The **R**andomized weak Decision 3-Party Diffie-Hellman (R-wD3DH1) assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ if no PPT distinguisher can distinguish the distribution*

$$\begin{aligned} D_1 &:= \Big\{ \{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{a_i bc})\}_{i=1}^{Q} \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, \\ &\qquad\qquad a_1, \ldots, a_Q, b, c \xleftarrow{R} \mathbb{Z}_p \} \Big\} \\ D_0 &:= \Big\{ \{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{z_i})\}_{i=1}^{Q} \mid g \xleftarrow{R} \mathbb{G}, \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, \\ &\qquad\qquad a_1, \ldots, a_Q, z_1, \ldots, z_Q, b, c \xleftarrow{R} \mathbb{Z}_p \} \Big\}. \end{aligned}$$

We do not know if D3DH1 or wD3DH1 can be tightly reduced to R-wD3DH1 (the only reduction we are aware of proceeds via a hybrid argument). In asymmetric pairings, however, we can give a tight reduction between R-wD3DH1 and a combination of wD3DH1 and SXDH.

**Lemma 2.1.** *There is a tight reduction from the wD3DH1 assumption and the DDH assumption in $\mathbb{G}$ to the R-wD3DH1 assumption. More precisely, for any R-wD3DH1 adversary $\mathcal{B}$, there exist distinguishers $\mathcal{B}_1$ and $\mathcal{B}_2$ that run in about the same time as $\mathcal{B}$ and such that*

$$\mathbf{Adv}_{\mathcal{B}}^{\text{R-wD3DH1}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_1}^{\text{wD3DH1}}(\lambda) + \mathbf{Adv}_{\mathcal{B}_2}^{\text{DDH}_1}(\lambda),$$

*where the second term denotes $\mathcal{B}_2$'s advantage as a DDH distinguisher in $\mathbb{G}$.*

*Proof.* To prove the result, we consider the following distribution:

$$\begin{aligned} D_{int} &:= \Big\{ \{(g, \hat{g}, g^{a \cdot \alpha_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, g^{z \cdot \alpha_i})\}_{i=1}^{Q} \mid g \xleftarrow{R} \mathbb{G},\ \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, \\ &\qquad\qquad \alpha_1, \ldots, \alpha_Q, b, c, z \xleftarrow{R} \mathbb{Z}_p,\ a \xleftarrow{R} \mathbb{Z}_p^{\star} \} \Big\} \end{aligned}$$

A straightforward reduction shows that, under the wD3DH1 assumption, $D_1$ is computationally indistinguishable from $D_{int}$. We show that, under the DDH assumption in $\mathbb{G}$, $D_{int}$

is computationally indistinguishable from $D_0$. Moreover, the reduction is tight in that the two distinguishers have the same advantage.

First, we show that, under the wD3DH1 assumption, $D_{int}$ is computationally indistinguishable from $D_1$.

We can build a wD3DH1 distinguisher $\mathcal{B}_1$ from any distinguisher for $D_1$ and $D_{int}$. With $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T)$ as input where $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$ and $a, b, c \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{B}_1$ uniformly draws $\alpha_i, \ldots, \alpha_Q \xleftarrow{R} \mathbb{Z}_p$ and computes

$$D_{\mathcal{B}_1} := \left\{ \{(g, \hat{g}, g^{a\cdot\alpha_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, T^{\alpha_i})\}_{i=1}^{Q} \mid \alpha_1, \ldots, \alpha_Q \xleftarrow{R} \mathbb{Z}_p \right\}.$$

It is easy to see that if $T = g^{abc}$, then $D_{B_1}$ is identical to $D_1$. If $T \in_R \mathbb{G}$, then $D_{B_1}$ is distributed as $D_{int}$. Hence, any distinguisher between $D_1$ and $D_{int}$ with $D_{\mathcal{B}_1}$ implies a distinguisher $\mathcal{B}_1$ for the wD3DH1 problem.

Next, we show that, under the DDH assumption in $\mathbb{G}$, $D_{int}$ is computationally indistinguishable from $D_0$. In order to build a DDH distinguisher $\mathcal{B}_2$ out of a distinguisher between $D_{int}$ and $D_0$, we use the random self-reducibility of the DDH assumption.

**Lemma 2.2** (Random Self-Reducibility [NR97]). *Letting $\mathbb{G}$ be a group of prime order $p$, there exists a PPT algorithm $R$ that takes as input $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, for any $a, b, c \in \mathbb{Z}_p$, and returns a triple $(g^a, g^{b'}, g^{c'}) \in \mathbb{G}^3$ such that:*

- *If $c = ab \mod q$, then $b'$ is uniformly random in $\mathbb{Z}_p$ and $c' = ab'$.*
- *If $c \neq ab \mod q$, then $b', c' \in_R \mathbb{Z}_p$ are independent and uniformly random.*

On input of $(g, g^z, g^\alpha, T) \in \mathbb{G}^4$, where $g \xleftarrow{R} \mathbb{G}$ and $z, \alpha \xleftarrow{R} \mathbb{Z}_p$, $\mathcal{B}_2$ uses algorithm $R$ to generate $Q$ instances $\{(g^z, g^{\alpha_i}, T_i)\}_{i=1}^{Q}$. Next, $\mathcal{B}_2$ draws $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$, $a, b, c \xleftarrow{R} \mathbb{Z}_p$ and defines the following distribution:

$$D_{\mathcal{B}_2} := \left\{ \{(g, \hat{g}, (g^{\alpha_i})^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T_i)\}_{i=1}^{Q} \mid \hat{g} \xleftarrow{R} \hat{\mathbb{G}}, a, b, c \xleftarrow{R} \mathbb{Z}_p \right\}.$$

We observe that, if $T = g^{z\cdot\alpha}$, we have $T_i = g^{z\cdot\alpha_i}$ for all $i \in [Q]$. In this case, $D_{\mathcal{B}_2}$ is identical to $D_{int}$. In contrast, if $T \in_R \mathbb{G}$, the random self-reducibility ensures that $T_1, \ldots, T_Q \in_R \mathbb{G}$ are i.i.d, meaning that $D_{\mathcal{B}_2}$ is identical to $D_0$. Using a distinguisher between $D_{int}$ and $D_0$ and feeding it with $D_{\mathcal{B}_2}$, we obtain a distinguisher $\mathcal{B}_2$ for the DDH problem in $\mathbb{G}$. □ □

# Part I.

# Lossy trapdoor functions and their applications

Lossy Trapdoor Functions (LTF), which is introduced by Peikert and Waters [PW08], are function families in which injective functions are computationally indistinguishable from many-to-one functions. Since their introduction, they drew a lot of attention [FGK+10; HO12; Hof12; Wee12; Zha16] and have been found very useful in constructions of many more advanced cryptographic primitives, such as chosen-ciphertext (IND-CCA) secure encryption schemes [PW08], as well as deterministic public-key encryption schemes [BFO08; BS11; RSV13].

In order to construct different cryptographic primitives, many variants of LTF have been introduced. In this part, we will study a different variant of lossy trapdoor functions (lossy algebraic filters) and provide more efficient constructions of this variant of LTF.

We provide a more efficient construction of LAF on DDH and corresponds to the following article published in PKC2019 by Benoît Libert, Chen Qian: Lossy Algebraic Filters with Short Tags.

# Table of Contents

# Lossy Algebraic Filters With Short Tags 3

## 3.1. Introduction



Lossy Algebraic Filters. Lossy algebraic filters (LAFs) are a variant LTFs introduced by Hofheinz [Hof13] as a tool enabling the design of chosen-ciphertext-secure encryption schemes with key-dependent message (KDM-CCA) security [BRS02]. Recently, they were also used by Wen and Liu [WL18] in the construction of robustly reusable fuzzy extractors. In LAF families, each function takes as arguments an input $x$ and a tag $t$, which determines if the function behaves as a lossy or an injective function. More specifically, each tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is comprised of an auxiliary component $t_{\mathsf{a}}$, which may consist of any public data, and a core component $t_{\mathsf{c}}$. For any auxiliary component $t_{\mathsf{a}}$, there should exist at least one $t_{\mathsf{c}}$ such that $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ induces a lossy function $f_{\mathsf{LAF}}(t, \cdot)$. LAFs strengthen the requirements of lossy trapdoor functions in that, for any lossy tag $t$, the function $f_{\mathsf{LAF}}(t, x)$ always reveals the same information about the input $x$, regardless of which tag is used. In particular, for a given evaluation key $ek$, multiple evaluations $f_{\mathsf{LAF}}(t_1, x), \ldots, f_{\mathsf{LAF}}(t_n, x)$ for distinct lossy tags do not reveal any more information about $x$ than a single evaluation. On the other hand, LAFs depart from lossy trapdoor functions in that they need not be efficiently invertible using a trapdoor. For their applications to KDM-CCA security [Hof13] and fuzzy extractors [WL18], lossy algebraic filters are expected to satisfy two security properties. The first one, called *indistinguishability*, requires that lossy tags be indistinguishable from random tags. The second one, named *evasiveness*, captures that lossy tags should be hard to come by without a trapdoor.

So far, the only known LAF realization is a candidate, suggested by Hofheinz [Hof13], which relies on the Decision Linear assumption (DLIN) [BBS04] in groups with a bilinear map. While efficient and based on a standard assumption, it incurs relatively large tags comprised of a quadratic number of group elements in the number of input symbols. More precisely, for functions admitting inputs $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$, where $p$ is the order of a pairing-friendly $\mathbb{G}$, the core components $t_{\mathsf{c}}$ contain $\Theta(n^2)$ elements of $\mathbb{G}$. For the application to KDM-CCA security [Hof13] (where $t_{\mathsf{c}}$ should be part of ciphertexts), quadratic-size tags are not prohibitively expensive as the encryption scheme of [Hof13, Section 4] can make do with a constant $n$ (typically, $n = 6$). In the application to fuzzy extractors [WL18], however, it is desirable to reduce the tag length. In the robustly reusable fuzzy extractor of [WL18], the core tag component $t_{\mathsf{c}}$ is included in the public helper string $P$ that allows reconstructing a secret key from a noisy biometric reading $w$. The latter lives in a metric space that should be small enough to fit in the input space $\mathbb{Z}_p^n$ of the underlying LAF family. Even if $p$ is exponentially large in the security parameter $\lambda$, a constant $n$ would restrict

biometric readings to have linear length in $\lambda$. Handling biometric readings of polynomial length thus incurs $n = \omega(1)$, which results in large tags and longer public helper strings. This motivates the design of new LAF candidates with smaller tags.

Our Results. The contribution of this chapter is two-fold. We first construct a new LAF with linear-size tags and prove it secure under simple, constant-size assumptions (as opposed to $q$-type assumptions, which are described using a linear number of elements in the number of adversarial queries) in bilinear groups. The indistinguishability and evasiveness properties of our scheme are implied by the Decision 3-party Diffie-Hellman assumption (more precisely, its natural analogue in asymmetric bilinear maps), which posits the pseudorandomness of tuples $(g, g^a, g^b, g^c, g^{abc})$, for random $a, b, c \in_R \mathbb{Z}_p$. For inputs in $\mathbb{Z}_p^n$, where $p$ is the group order, our core tag components only consist of $O(n)$ group elements. These shorter tags are obtained without inflating evaluation keys, which remain of length $O(n)$ (as in [Hof13]).

As a second contribution, we provide a second LAF realization with $O(n)$-size tags where the indistinguishability and evasiveness properties are both *a*lmost tightly related to the underlying hardness assumption. Namely, our security proofs are tight – or almost tight in the terminology of Chen and Wee [CW13] – in that the gap between the advantages of the adversary and the reduction only depends on the security parameter, and not on the number of adversarial queries. In the LAF suggested by Hofheinz [Hof13], the proof of evasiveness relies on the unforgeability of Waters signatures [Wat05]. As a result, the reduction loses a linear factor in the number of lossy tags obtained by the adversary. In our second construction, we obtain tight reductions by replacing Waters signatures with (a variant of) a message authentication code (MAC) due to Blazy, Kiltz and Pan [BKP14]. As a result, our proof of evasiveness only loses a factor $O(\lambda)$ with respect to the Symmetric eXternal Diffie-Hellman assumption (SXDH). If our scheme is plugged into the robustly reusable fuzzy extractor of Wen and Liu [WL18], it immediately translates into a tight proof of robustness in the sense of the definition of [WL18]. While directly using our second LAF in the KDM-CCA-secure scheme of [Hof13] does not seem sufficient to achieve tight key-dependent message security, it may still provide a building block for future constructions of tightly KDM-CCA-secure encryption schemes with short ciphertexts.

Techniques. Like the DLIN-based solution given by Hofheinz [Hof13], our evaluation algorithms proceed by computing a matrix-vector product in the exponent, where the matrix is obtained by pairing group elements taken from the core tag $t_c$ with elements of the evaluation key. Here, we reduce the size of $t_c$ from $O(n^2)$ to $O(n)$ group elements using a technique suggested by Boyen and Waters [BW10] in order to compress the evaluation keys of DDH-based lossy trapdoor functions.

In the pairing-based LTF of [BW10], the evaluation key contains group elements $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$, $\{(V_j = g^{v_j}, H_j = (h^j \cdot u)^{v_j})\}_{j=1}^n$. Using a symmetric bilinear maps $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, these make it possible to compute the off-diagonal elements of a matrix

$$M_{i,j} = e(g,h)^{r_i \cdot v_j} = \left(\frac{e(R_i, H_j)}{e(S_i, V_j)}\right)^{1/(j-i)} \qquad \forall (i,j) \in [n] \times [n] \setminus \{(i,i)\}_{i=1}^n \tag{3.1}$$

via a "two equation" technique borrowed from the revocation system of Lewko, Sahai and

Waters [LSW10]. By including $\{D_i = e(g,g)^{r_i \cdot v_i} \cdot e(g,g)\}_{i=1}^n$ in the evaluation key, the LTF of [BW10] allows the evaluator to compute a matrix $(M_{i,j})_{i,j\in[n]}$ such that $M_{i,j} = e(g,g)^{r_i \cdot v_j}$ if $i \neq j$ and $M_{i,i} = e(g,g)^{r_i \cdot v_i} \cdot e(g,g)^{m_i}$ and for which $m_i = 1$ (resp. $m_i = 0$), for all $i \in [n]$, in injective (resp. lossy) functions. The indistinguishability of lossy and injective evaluation keys relies on the fact that (3.1) is only computable when $i \neq j$, making it infeasible to distinguish $\{D_i = e(g,h)^{r_i \cdot v_i} \cdot e(g,g)\}_{i=1}^n$ from $\{D_i = e(g,h)^{r_i \cdot v_i}\}_{i=1}^n$.

Our first LAF construction relies on the "two equation" technique of [LSW10] in a similar way with the difference that we include $\{(V_j = g^{v_j}, H_j = (h^j \cdot u)^{v_j}\}_{j=1}^n$ in the evaluation key $ek$, but $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$ is now part of the core tag components $t_c$. This makes it possible to compute off-diagonal elements of $(M_{i,j})_{i,j\in[n]}$ by pairing elements of $ek$ with those of $t_c$. To enable the computation of diagonal elements $\{M_{i,i}\}_{i=1}^n$, we augment core tag components by introducing pairs $(D_i, E_i) \in \mathbb{G}^2$, which play the same role as $\{D_i = e(g,g)^{r_i \cdot v_i} \cdot e(g,g)\}_{i=1}^n$ in the LTF of [BW10]. In lossy tags, $\{(D_i, E_i)\}_{i=1}^n$ are of the form

$$(D_i, E_i) = (h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, g^{\rho_i}), \tag{3.2}$$

for a random $\rho_i \in_R \mathbb{Z}_p$, where $\tau$ is a chameleon hashing of all tag components. Such pairs $\{(D_i, E_i)\}_{i=1}^n$ allow the evaluator to compute

$$M_{i,i} = \frac{e(D_i, g)}{e(H_{\mathbb{G}}(\tau), E_i)} = e(g,h)^{r_i \cdot v_i} \qquad \forall i \in [n],$$

which results in a rank-one matrix $(M_{i,j})_{i,j\in[n]}$, where $M_{i,j} = e(g,h)^{r_i \cdot v_j}$. When computed as per (3.2), $\{(D_i, E_i)\}_{i=1}^n$ can be seen as "blinded" Waters signatures [Wat05]. Namely, $(g, h, V_i = g^{v_i})$ can be seen as a verification key; $h^{v_i}$ is the corresponding secret key; and $r_i \in \mathbb{Z}_p$ serves as a blinding factor that ensures the indistinguishability of $(D_i, E_i)$ from random pairs. Indeed, the Decision 3-party Diffie-Hellman (D3DH) assumption allows proving that $h^{r_i \cdot v_i}$ is computationally indistinguishable from random given $(g, h, g^{v_i}, g^{r_i})$. In our proof of indistinguishability, however, we need to rely on the proof technique of the Boneh-Boyen IBE [BB04] in order to apply a hybrid argument that allows gradually replacing pairs $\{(D_i, E_i)\}_{i=1}^n$ by random group elements.

In our proof of evasiveness, we rely on the fact that forging a pair of the form $(D_i, E_i) = (h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, g^{\rho_i})$ on input of $(g, h, g^{v_i})$ is as hard as solving the 2-3-Diffie-Hellman problem [KP06], which consist in finding a non-trivial pair $(g^r, g^{r \cdot ab}) \in \mathbb{G}^* \times \mathbb{G}^*$ on input of $(g, g^a, g^b)$. In turn, this problem is known to be at least as hard as breaking the Decision 3-party Diffie-Hellman assumption.

The above techniques allow us to construct a LAF with $O(n)$-size tags and evaluation keys made of $O(n + \lambda)$ group elements under a standard assumption. Our first LAF is actually described in terms of asymmetric pairings, but it can be instantiated in all types (i.e., symmetric or asymmetric) of bilinear groups. Our second LAF construction requires asymmetric pairing configurations and the Symmetric eXternal Diffie-Hellman (SXDH) assumption. It is very similar to our first construction with the difference that we obtain a tight proof of evasiveness by replacing Waters signatures with a variant of a MAC proposed by Blazy, Kiltz and Pan [BKP14]. In order for the proofs to go through, we need to include

$n$ MAC instances (each with its own keys) in lossy tags, which incurs evaluation keys made of $O(n \cdot \lambda)$ group elements. We leave it is an interesting open problem to achieve tight security using shorter evaluation keys.

Related Work. All-but-one lossy trapdoor functions (ABO-LTFs) [PW08] are similar to LAFs in that they are lossy function families where each function is parametrized by a tag that determines if the function is injective or lossy. They differ from LAFs in two aspects: (i) They should be efficiently invertible using a trapdoor; (ii) For a given evaluation key $ek$, there exists only one tag for which the function is lossy. The main motivation of ABO-LTFs is the construction of chosen-ciphertext [RS91] encryption schemes. *A*ll-but-many lossy trapdoor functions (ABM-LTFs) are an extension of ABO-LTFs introduced by Hofheinz [Hof12]. They are very similar to LAFs in that a trapdoor makes it possible to dynamically create arbitrarily many lossy tags using. In particular, each tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ consists of an auxiliary component $t_{\mathsf{a}}$ and a core component $t_{\mathsf{c}}$ so that, by computing $t_{\mathsf{c}}$ as a suitable function of $t_{\mathsf{a}}$, the pair $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ can be made lossy, but still random-looking. The motivation of ABM-LTFs is the construction chosen-ciphertext-secure public-key encryption schemes in scenarios, such as the selective-opening setting [DNRS99; BHY09], which involve multiple challenge ciphertexts [Hof12]. They also found applications in the design of universally composable commitments [Fuj14]. Lossy algebraic filters differ from ABM-LTFs in that they may not have a trapdoor enabling efficient inversion but, for any lossy tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, the information revealed by $f_{\mathsf{LAF}}(t, x)$ is always the same (i.e., it is completely determined by $x$ and the evaluation key $ek$).

LAFs were first introduced by Hofheinz [Hof13] as a building block for KDM-CCA-secure encryption schemes, where they enable some form of "plaintext awareness". In the security proofs of KDM-secure encryption schemes (e.g., [BHHO08]), the reduction must be able to simulate encryptions of (functions of) the secret key. When the adversary is equipped with a decryption oracle, the ability to publicly compute encryptions of the decryption key may be a problem as decryption queries could end up revealing that key. LAFs provide a way reconcile the conflicting requirements of KDM and CCA2-security by introducing in each ciphertext a LAF-evaluation of the secret key. By having the simulator encrypt a lossy function of the secret key, one can keep encryption queries from leaking too much secret information. At the same time, adversarially-generated ciphertexts always contain an injective function of the key, which prevents the adversary from learning the secret key by publicly generating encryptions of that key.

Recently, Wen and Liu [WL18] appealed to LAFs in the design of robustly reusable fuzzy extractors. As defined by Dodis *e*t al. [DRS04], fuzzy extractors allow one to generate a random cryptographic key $R$ – together with some public helper string $P$ – out of a noisy biometric reading $w$. The key $R$ need not be stored as it can be reproduced from the public helper string $P$ and a biometric reading $w'$ which is sufficiently close to $w$. Reusable fuzzy extractors [Boy04] make it possible to safely generate multiple keys $R_1, \ldots, R_t$ (each with its own public helper string $P_i$) from correlated readings $w_1, \ldots, w_t$ of the same biometric source. Wen and Liu [WL18] considered the problem of achieving robustness in reusable fuzzy extractors. In short, robustness prevents adversaries from covertly tampering with the public helper string $P_i$ in order to affect the reproducibility of $R_i$. The Wen-Liu [WL18] fuzzy

extractor relies on LAFs to simultaneously achieve reusability and robustness assuming a common reference string. Their solution requires the LAF to be homomorphic, meaning that function outputs should live in a group and, for any tag $t$ and inputs $x_1$, $x_2$, we have $f_{\mathsf{LAF}}(t, x_1 + x_2) = f_{\mathsf{LAF}}(t, x_1) \cdot f_{\mathsf{LAF}}(t, x_2)$. The candidate proposed by Hofheinz [Hof13] and ours are both usable in robustly reusable fuzzy extractors as they both satisfy this homomorphic property. Our scheme offers the benefit of shorter public helper strings $P$ since these have to contain a LAF tag in the fuzzy extractor of [WL18].

The tightness of cryptographic security proofs was first considered by Bellare and Rogaway [BR96] in the random oracle model [BR93]. In the standard model, it drew a lot of attention in digital signatures and public-key encryption the recent years (see, e.g., [HJ12; CW13; BKP14; LJYP14; LPJY15; Hof16; GHKW16; Hof17; GHK17]). In the context of all-but-many lossy trapdoor functions, a construction with tight evasiveness was put forth by Hofheinz [Hof12]. A tightly secure lattice-based ABM-LTF was described by Libert *et* al. [LSSS17] as a tool enabling tight chosen-ciphertext security from lattice assumptions. To our knowledge, the only other prior work considering tight reductions for lossy trapdoor functions is a recent result of Hofheinz and Nguyen [HN19]. In particular, tight security has never been obtained in the context of LAFs, nor in fuzzy extractors based on public-key techniques.



## 3.2. Background

### 3.2.1. Lossy Algebraic Filters

We recall the definition of Lossy Algebraic Filter (LAF) from [Hof13], in which the distribution over the function domain may not be the uniform one.

**Definition 3.1.** *For integers $\ell_{\mathsf{LAF}}(\lambda), n(\lambda) > 0$, an $(\ell_{\mathsf{LAF}}, n)$-lossy algebraic filter (*$\mathsf{LAF}$*) with security parameter $\lambda$ consists of the following* ppt *algorithms:*

***Key generation.*** $\mathsf{LAF.Gen}(1^\lambda)$ *outputs an evaluation key $ek$ and a trapdoor key $tk$. The evaluation key $ek$ specifies an $\ell_{\mathsf{LAF}}$-bit prime $p$ as well as the description of a tag space $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \mathcal{T}_{\mathsf{a}}$, where $\mathcal{T}_{\mathsf{c}}$ is efficiently samplable. The disjoint sets of injective and non-injective tags tags are called $\mathcal{T}_{\mathsf{inj}}$ and $\mathcal{T}_{\mathsf{non\text{-}inj}} = \mathcal{T} \setminus \mathcal{T}_{\mathsf{inj}}$, respectively. We also define the subset of lossy tags $\mathcal{T}_{\mathsf{loss}}$ to be a subset of $\mathcal{T}_{\mathsf{non\text{-}inj}}$, which induce very lossy functions. A tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is described by a core part $t_{\mathsf{c}} \in \mathcal{T}_{\mathsf{c}}$ and an auxiliary part $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$. A tag may be injective, or lossy, or neither. The trapdoor $tk$ allows sampling lossy tags.*

***Evaluation.*** $\mathsf{LAF.Eval}(ek, t, X)$ *takes as inputs an evaluation key $ek$, a tag $t \in \mathcal{T}$ and a function input $X \in \mathbb{Z}_p^n$. It outputs an image $Y = f_{ek,t}(X)$.*

***Lossy tag generation.*** $\mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$ *takes as input the trapdoor key $tk$, an auxiliary part $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$ and outputs a core part $t_{\mathsf{c}}$ such that $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) \in \mathcal{T}_{\mathsf{loss}}$ forms a lossy tag.*

*In addition,* $\mathsf{LAF}$ *has to meet the following requirements:*

***Lossiness.*** *For any $(ek, tk) \xleftarrow{R} \mathsf{LAF.Gen}(1^\lambda)$, the following conditions should be satisfied.*

a. *For any $t \in \mathcal{T}_{\mathsf{inj}}$, $f_{ek,t}(.)$ should behave as an injective function (note that $f^{-1}_{ek,t}(.)$ is not required to be efficiently computable given $tk$).*

b. *For any auxiliary tag $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$ and any $t_{\mathsf{c}} \xleftarrow{R} \mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$, we have $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) \in \mathcal{T}_{\mathsf{loss}}$, meaning that $f_{ek,t}(.)$ is a lossy function. Moreover, for any input $X = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ and any $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) \in \mathcal{T}_{\mathsf{loss}}$, $f_{ek,t}(X)$ is completely determined by $\sum_{i=1}^{n} v_i \cdot x_i \bmod p$ for coefficients $\{v_i\}_{i=1}^{n}$ that only depend on $ek$.*

***Indistinguishability.*** *Multiple lossy tags are computationally indistinguishable from random tags, namely:*

$$\mathbf{Adv}^{\mathcal{A},\mathsf{ind}}_{Q}(\lambda) := \left| \Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot)} = 1] - \Pr[\mathcal{A}(1^\lambda, ek)^{\mathcal{O}_{\mathcal{T}_{\mathsf{c}}}(\cdot)} = 1] \right|$$

*is negligible for any PPT algorithm $\mathcal{A}$, where $(ek, tk) \xleftarrow{R} \mathsf{LAF.Gen}(1^\lambda)$ and $\mathcal{O}_{\mathcal{T}_{\mathsf{c}}}(\cdot)$ is an oracle that assigns a random core tag $t_{\mathsf{c}} \xleftarrow{R} \mathcal{T}_{\mathsf{c}}$ to each auxiliary tag $t_{\mathsf{a}} \in \mathcal{T}_{\mathsf{a}}$ (rather than a core tag that makes $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ lossy). Here $Q$ denotes the number of oracle queries made by $\mathcal{A}$.*

***Evasiveness.*** *Non-injective tags are computationally hard to find, even with access to an oracle outputting multiple lossy tags, namely:*

$$\mathbf{Adv}^{\mathcal{A},\mathsf{eva}}_{Q_1,Q_2}(\lambda) := \Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot),\ \mathsf{LAF.IsLossy}(tk,\cdot)} \in \mathcal{T}_{\mathsf{non\text{-}inj}}]$$

*is negligible for legitimate adversary $\mathcal{A}$, where $(ek, ik, tk) \xleftarrow{R} \mathsf{LAF.Gen}(1^\lambda)$ and $\mathcal{A}$ is given access to the following oracle:*

- $\mathsf{LAF.LTag}(tk, \cdot)$ *which acts exactly as the lossy tag generation algorithm.*
- $\mathsf{LAF.IsLossy}(tk, \cdot)$ *that takes as input a tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$. It outputs $0$ if $t \in \mathcal{T}_{\mathsf{non\text{-}inj}} = \mathcal{T} \backslash \mathcal{T}_{\mathsf{inj}}$ and $1$ if $t \in \mathcal{T}_{\mathsf{inj}}$. If $t \notin \mathcal{T}$, the oracle outputs $\perp$.*

*We denote by $Q_1$ and $Q_2$ the number of queries to $\mathsf{LAF.LTag}(tk, \cdot)$ and $\mathsf{LAF.IsLossy}(tk, \cdot)$, respectively. By "legitimate adversary", we mean that $\mathcal{A}$ is PPT and never outputs a tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ such that $t_{\mathsf{c}}$ was obtained by invoking the $\mathsf{LAF.LTag}$ oracle on $t_{\mathsf{a}}$.*

In our construction, the tag space $\mathcal{T}$ will not be dense (i.e., not all elements of the ambient algebraic structure are potential tags). However, elements of the tag space $\mathcal{T}$ will be efficiently recognizable given $ek$.

We note that the above definition of evasiveness departs from the one used by Hofheinz [Hof13] in that it uses an additional $\mathsf{LAF.IsLossy}(tk, \cdot)$ oracle that uses the trapdoor $tk$ to decide whether a given tag is injective or not. However, this oracle will only be used in our tightly secure LAF (and not in our first construction). Its only purpose is to enable a modular use of our tightly evasive LAF in applications to KDM security [Hof13] or robustly reusable fuzzy extractors [WL18]. Specifically, by invoking the $\mathsf{LAF.IsLossy}(tk, \cdot)$ oracle, the reduction from the security of a primitive to the underlying LAF's evasiveness does not have to guess which adversarial query involves a non-lossy tag.

## 3.3. A Lossy Algebraic Filter With Linear-Size Tags

We present a LAF based on DDH-like assumptions with tags of size $O(n)$, where $n$ is the number of input symbols when the input is viewed as a vector over $\mathbb{Z}_p$. Our tags are comprised of $4n$ elements of $\mathbb{G}$, which outperforms the construction of [Hof13] for $n > 4$. In his application to KDM-CCA security [Hof13], Hofheinz uses a LAF scheme with $n = 6$, in which case we decrease the tag size from 43 to 24 group elements[1] and thus shorten ciphertexts by 19 group elements.

The construction is inspired by the lossy TDF of [BW10] and relies on the revocation technique of Lewko, Sahai and Waters [LSW10] (LSW) in the same way. In asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, the evaluation key contains a set of LSW ciphertexts $\{(\hat{V}_j = \hat{g}^{v_j}, \hat{H}_j = (\hat{h}^j \cdot \hat{u})^{v_j})\}_{j=1}^n$, while each core tag component $t_\mathsf{c}$ can be seen as containing a set of LSW secret keys $\{(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})\}_{i=1}^n$, allowing the evaluator compute $M_{i,j} = e(g, \hat{h})^{r_i \cdot v_j}$ for any pairwise distinct indices $i \neq j$. In lossy tags $(t_\mathsf{c}, t_\mathsf{a})$, diagonal elements $\{M_{i,i}\}_{i=1}^n$ are handled by having $t_\mathsf{c}$ contain Waters signatures $(D_i, E_i) = (h^{r_i \cdot v_i} \cdot H_\mathbb{G}(\tau)^{\rho_i}, g^{\rho_i})$, where $\rho_i \in_R \mathbb{Z}_p$ and $H_\mathbb{G} : \{0,1\}^L \to \mathbb{G}$ is an algebraic hash function mapping the output $\tau$ of a chameleon hash function to the group $\mathbb{G}$. For indistinguishability purposes, pairs $\{(D_i, E_i)\}_{i=1}^n$ are not immediately recognizable as Waters signatures because the underlying secret key $h^{v_i}$ is blinded by a random exponent $r_i = \log_g(R_i)$. Still, running the verification algorithm of Waters signatures on $(D_i, E_i)$ allows the evaluation algorithm to derive $M_{i,i} = e(g, \hat{h})^{r_i \cdot v_i}$, so that $(M_{i,j})_{i,j \in [n]}$ forms a rank-1 matrix. In injective tags, $\{(D_i, E_i)\}_{i=1}^n$ are uniformly distributed in $\mathbb{G}$, so that $(M_{i,j})_{i,j \in [n]}$ is the sum of a rank-1 matrix and a diagonal matrix.

### 3.3.1. Description

**Key generation.** $\mathsf{LAF.Gen}(1^\lambda)$ conducts the following steps.

1. Choose bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with random generators $g, h, u \xleftarrow{R} \mathbb{G}$ and $\hat{g}, \hat{h}, \hat{u} \xleftarrow{R} \hat{\mathbb{G}}$ subject to the constraints $\log_g(h) = \log_{\hat{g}}(\hat{h})$ and $\log_g(u) = \log_{\hat{g}}(\hat{u})$.
2. Choose a chameleon hash function $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$, where the hashing algorithm $\mathsf{CMhash} : \{0,1\}^* \times \mathcal{R}_{hash} \to \{0,1\}^L$ has output length $L \in \mathsf{poly}(\lambda)$. Generate a pair $(hk_\mathsf{CMH}, td_\mathsf{CMH}) \leftarrow \mathsf{CMKg}(1^\lambda)$ made of a hashing key $hk_\mathsf{CMH}$ and a trapdoor $td_\mathsf{CMH}$.
3. Choose random exponents $w_0, \ldots, w_L \xleftarrow{R} \mathbb{Z}_p$ and define

$$W_k = g^{w_k}, \qquad \hat{W}_k = \hat{g}^{w_k} \qquad \forall k \in [0, L]$$

that will be used to instantiate two hash functions $H_\mathbb{G} : \{0,1\}^L \to \mathbb{G}$, $H_{\hat{\mathbb{G}}} :$

[1] The LAF of [Hof13] was described in terms of symmetric pairings but it extends to asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ where tags are comprised of elements in $\mathbb{G}$.

$\{0,1\}^L \to \hat{\mathbb{G}}$ which map any string $\mathsf{m} \in \{0,1\}^L$ to

$$H_{\mathbb{G}}(\mathsf{m}) = W_0 \cdot \prod_{k=1}^{L} W_k^{\mathsf{m}[k]}, \qquad H_{\hat{\mathbb{G}}}(\mathsf{m}) = \hat{W}_0 \cdot \prod_{k=1}^{L} \hat{W}_k^{\mathsf{m}[k]},$$

respectively. Note that $e(g, H_{\hat{\mathbb{G}}}(\mathsf{m})) = e(H_{\mathbb{G}}(\mathsf{m}), \hat{g})$ for any $\mathsf{m} \in \{0,1\}^L$.

4. Let $n \in \mathsf{poly}(n)$ be the desired input length. For each $j \in [n]$, choose $v_j \xleftarrow{R} \mathbb{Z}_p$ and define

$$\hat{V}_j \;=\; \hat{g}^{v_j}, \qquad \hat{H}_j = (\hat{h}^j \cdot \hat{u})^{v_j} \qquad \forall j \in [n].$$

5. Output the evaluation key $ek$ and the lossy tag generation key $tk$, which consist of

$$\begin{aligned} ek &:= \Big(hk_{\mathsf{CMH}},\ g,\ h,\ u,\ \hat{g},\ \hat{h},\ \hat{u},\ \{W_k, \hat{W}_k\}_{k=0}^{L},\ \ \{\hat{V}_j, \hat{H}_j\}_{j=1}^{n}\Big), \\ tk &:= \big(td_{\mathsf{CMH}},\ \{v_j\}_{j=1}^{n}\big). \end{aligned}$$

The tag space $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \mathcal{T}_{aux}$ is defined as a product of $\mathcal{T}_{\mathsf{a}} = \{0,1\}^*$ and

$$\begin{aligned} \mathcal{T}_{\mathsf{c}} \;:=\; \{\big(\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash}\big) \mid r_{hash} \in \mathcal{R}_{\mathsf{CMH}} \;\wedge \\ \forall i \in [n] : (R_i, S_i, D_i, E_i) \in \mathbb{G}^{*4} \;\wedge\; e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})\}, \end{aligned}$$

where $\mathbb{G}^* := \mathbb{G} \setminus \{1_{\mathbb{G}}\}$. The range of the function family is $\mathsf{Rng}_\lambda = \mathbb{G}_T^{n+1}$ and its domain is $\mathbb{Z}_p^n$.

**Lossy tag generation.** $\mathsf{LAF.LTag}(tk, t_{\mathsf{a}})$ takes in an auxiliary tag component $t_{\mathsf{a}} \in \{0,1\}^*$ and uses $tk = \big(td_{\mathsf{CMH}}, \{v_j\}_{j=1}^{n}, \{w_k\}_{k=0}^{L}\big)$ to generate a lossy tag as follows.

1. For each $i \in [n]$, choose $r_i \xleftarrow{R} \mathbb{Z}_p^*$ and compute

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n]. \tag{3.3}$$

2. For each $i \in [n]$, choose $\rho_i \xleftarrow{R} \mathbb{Z}_p$ and compute

$$D_i = h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, \qquad E_i = g^{\rho_i} \qquad \forall i \in [n],$$

where $\tau \in \{0,1\}^L$ is chosen uniformly in the range of CMhash.

3. Use the trapdoor $td_{\mathsf{CMH}}$ to find $r_{hash} \in \mathcal{R}_{hash}$ such that

$$\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^{n}), r_{hash}\big) \in \{0,1\}^L$$

and output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash})$.

Each lossy tag is associated with a matrix $\big(M_{i,j}\big)_{i,j\in[n]} = \big(e(g,\hat{h})^{r_i\cdot v_j}\big)_{i,j}$, which is a rank-1 matrix in the exponent. Its diagonal entries consist of

$$M_{i,i} = \frac{e(D_i,\hat{g})}{e(E_i, H_{\hat{\mathbb{G}}}(\tau))} = e(g,\hat{h})^{r_i\cdot v_i} \qquad \forall i \in [n], \tag{3.4}$$

while its non-diagonal entries

$$M_{i,j} = \Big(\frac{e(R_i,\hat{H}_j)}{e(S_i,\hat{V}_j)}\Big)^{1/(j-i)} = \; e(g,\hat{h})^{r_i\cdot v_j} \qquad \forall (i,j) \in [n]\times[n] \setminus \{(i,i)\}_{i=1}^n, \tag{3.5}$$

are obtained by pairing tag component $(R_i, S_i)$ with evaluation key components $(\hat{V}_j, \hat{H}_j)$.

**Random Tags.** A random tag can be publicly sampled as follows.

1. For each $i \in [n]$, choose $r_i \xleftarrow{R} \mathbb{Z}_p^*$ and compute $\{R_i, S_i\}_{i=1}^n$ as in (3.3).
2. For each $i \in [n]$, choose $(D_i, E_i) \xleftarrow{R} \mathbb{G}^* \times \mathbb{G}^*$ uniformly at random.
3. Choose $r_{hash} \xleftarrow{R} \mathcal{R}_{hash}$.

Finally, output the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, where $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$.

We note that, in both random and lossy tags, we have $e(R_i, \hat{u}^i \cdot \hat{h}) = e(S_i, \hat{g})$ for all $i \in [n]$, so that elements of $\mathcal{T}$ are publicly recognizable.

**Evaluation.** $\mathsf{LAF.Eval}(ek, t, \mathbf{x})$ takes in the input $\mathbf{x} \in \mathbb{Z}_p^n$ and the tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$. It parses $t_{\mathsf{c}}$ as $(\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ and proceeds as follows.

1. Return $\perp$ if there exists $i \in [n]$ such that $e(R_i, \hat{h}^i \cdot \hat{u}) \neq e(S_i, \hat{g})$.
2. Compute the matrix $\big(M_{i,j}\big)_{i,j\in[n]} \in \mathbb{G}_T^{n\times n}$ as

$$M_{i,i} = \frac{e(D_i,\hat{g})}{e(E_i, H_{\hat{\mathbb{G}}}(\tau))} \qquad \forall i \in [n] \qquad , \tag{3.6}$$

where $\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^n), r_{hash}\big)$, and

$$M_{i,j} = \Big(\frac{e(R_i,\hat{H}_j)}{e(S_i,\hat{V}_j)}\Big)^{1/(j-i)} \qquad \forall (i,j) \in [n]\times[n] \setminus \{(i,i)\}_{i=1}^n, \tag{3.7}$$

Note that, since $R_i = g^{r_i}$ and $S_i = (h^i \cdot u)^{r_i}$ for some $r_i \in \mathbb{Z}_q$, we have

$$\begin{aligned} M_{i,i} &= e(g,\hat{h})^{r_i\cdot v_i+\omega_i}, \qquad \forall i \in [n] \\ M_{i,j} &= e(g,\hat{h})^{r_i\cdot v_j}, \qquad \forall i \neq j, \end{aligned} \tag{3.8}$$

for some vector $(\omega_1, \ldots, \omega_n)^\top \in \mathbb{Z}_p^n$ that only contains non-zero entries if $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ is injective.

3. Compute $\left(V_{T,j}\right)_{j\in[n]}$ as $V_{T,j} = e(h, \hat{V}_j) = e(g, \hat{h})^{v_j}$ for each $j \in [n]$.
4. Use the input $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$ to compute

$$\begin{aligned} Y_0 &= \prod_{j=1}^{n} V_{T,j}^{x_j} \\ Y_i &= \prod_{j=1}^{n} M_{i,j}^{x_j} \qquad \forall i \in [n] \end{aligned} \tag{3.9}$$

and output $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n)^\top \in \mathbb{G}_T^{n+1}$.

While the above construction inherits the $\Theta(\lambda)$-size public keys of Waters signatures [Wat05], we believe that it can be adapted to other signature schemes in the standard model (e.g., [BHJ+13; JR13]) so as to obtain shorter evaluation keys.

Injectivity and lossiness. For any injective tag, all entries of the vector $(\omega_1, \ldots, \omega_n)^\top$ are non-zero in (3.8). We can use $Y_0$ to ensure that the function is injective. As long as $\omega_i \neq 0$ for all $i \in [n]$, the evaluation algorithm (3.9) yields a vector $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$ of the form

$$\begin{aligned} Y_0 &= e(g, \hat{h})^{\sum_{j=1}^{n} v_j \cdot x_j} \\ Y_i &= e(g, \hat{h})^{\omega_i \cdot x_i + r_i \cdot \sum_{j=1}^{n} v_j \cdot x_j} \qquad \forall i \in [n], \end{aligned}$$

meaning that $x_i \in \mathbb{Z}_p$ is uniquely determined by $(Y_0, Y_i)$ and $(R_i, D_i, E_i)$ (note that the triple $(R_i, D_i, E_i)$ uniquely defines $\omega_i$).

For any lossy tag, the evaluation outputs $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$ such that

$$\begin{aligned} Y_0 &= e(g, \hat{h})^{\sum_{j=1}^{n} v_j \cdot x_j} \\ Y_i &= e(g, \hat{h})^{r_i \cdot \sum_{j=1}^{n} v_j \cdot x_j} \qquad \forall i \in [n], \end{aligned}$$

which always reveals the same information $\sum_{j=1}^{n} v_j \cdot x_j \bmod p$ about the input vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$, no matter which tag is used.

### 3.3.2. Security

The proof of indistinguishability relies on the wD3DH1 assumption via a hybrid argument over the queries to the $\mathsf{LAF.LTag}(tk, \cdot)$ oracle and over the pairs $\{(D_i, E_i)\}_{i=1}^n$ produced by $\mathsf{LAF.LTag}(tk, \cdot)$ at each query. Using the R-wD3DH1 assumption, it is possible to modify the proof so as to use a hybrid argument over the pairs $\{(D_i, E_i)\}_{i=1}^n$ only (meaning that all queries to $\mathsf{LAF.LTag}(tk, \cdot)$ are processed in parallel at each game transition). However, this proof would require the SXDH assumption – which only holds in asymmetric pairings – to apply the result of Lemma 2.1. In contrast, the proof of Theorem 3.1 allows instantiations in all bilinear group configurations, even in symmetric pairings.

The proof of Theorem 3.1 uses a hybrid argument to gradually replace pairs $\{(D_i, E_i)\}_{i=1}^n$ by truly random group elements in outputs of the lossy tag generation oracle. To this end, it relies on the proof technique of the Boneh-Boyen IBE [BB04] in the proof of Lemma 3.2. Namely, in order to embed a D3DH1 instance $(g, h, g^{v_k}, g^{r_k}, T \stackrel{?}{=} h^{r_k \cdot v_k})$ in the $k$-th pair $(D_k, E_k)$, for indexes $i > k$, the reduction has to simulate $h^{r_i \cdot v_k}$ for a known $r_i \in \mathbb{Z}_p$ and an unknown $h^{v_k}$.

**Theorem 3.1.** *The above LAF provides indistinguishability under the wD3DH1 assumption in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

*Proof.* We first recall that, for any injective or non-injective tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$, the core component $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ imply a matrix $\left(M_{i,j}\right)_{i,j\in[n]}$ where the off-diagonal entries are $M_{i,j} = e(g, \hat{h})^{r_i \cdot v_j}$ and the diagonal entries are of the form (3.8). In injective tags, the vector $(\omega_1, \ldots, \omega_n)^\top \in \mathbb{Z}_p^n$ only contains non-zero entries. In lossy tags, we have $(\omega_1, \ldots, \omega_n)^\top = \mathbf{0}^n$. We define a sequence of hybrid games. In $\mathsf{Game}_{(0,0)}$, the adversary has access to the real oracle $\mathsf{LAF.LTag}(tk, .)$ oracle that always outputs lossy tags. In $\mathsf{Game}_{(Q,n)}$, the adversary is given access to an oracle $\mathcal{O}_{\mathcal{T}_{\mathsf{c}}}(.)$ that always outputs random tags.

**$\mathsf{Game}_{(\ell,k)}$** $(1 \leq \ell \leq Q, 1 \leq k \leq n)$**:** In this game, the adversary interacts with a hybrid oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$. At the $\mu$-th query, this oracle outputs tags $t^{(\mu)} = (t_{\mathsf{c}}^{(\mu)}, t_{\mathsf{a}}^{(\mu)})$ such that

- If $\mu < \ell$, the tag $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ implies a matrix $\left(M_{i,j}^{(\mu)}\right)_{i,j\in[n]}$ of the form (3.8) where $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top$ is uniform over $\mathbb{Z}_p^n$
- If $\mu = \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ implies a matrix $\left(M_{i,j}^{(\mu)}\right)_{i,j\in[n]}$ of the form (3.8) where the first $k$ entries of $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top$ are uniform over $\mathbb{Z}_p$ and its last $n - k$ entries are zeroes.
- If $\mu > \ell$, the matrix $\left(M_{i,j}^{(\mu)}\right)_{i,j\in[n]}$ implied by the core tag component

$$t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$$

is a rank-1 matrix in the exponent since $(\omega_1^{(\mu)}, \ldots, \omega_n^{(\mu)})^\top = \mathbf{0}^n$.

Lemma 3.2 shows that, for all pairs $(\ell, k) \in [Q] \times [n]$, these games are computationally indistinguishable from one another, which yields the stated result. □ □

**Lemma 3.2.** *For each $k \in [n]$ and $\ell \in [Q]$,* $\mathsf{Game}_{(\ell,k)}$ *is computationally indistinguishable from* $\mathsf{Game}_{(\ell,k-1)}$ *if the wD3DH1 assumption holds. Under the same assumption,* $\mathsf{Game}_{(\ell,1)}$ *is computationally indistinguishable from* $\mathsf{Game}_{(\ell-1,n)}$.

*Proof.* For the sake of contradiction, assume that there exists $\ell \in [Q]$, $k \in [n]$ such that the adversary $\mathcal{A}$ can distinguish $\mathsf{Game}_{(\ell,k)}$ from $\mathsf{Game}_{(\ell,k-1)}$ with noticeable advantage (the indistinguishability of $\mathsf{Game}_{(\ell-1,n)}$ and $\mathsf{Game}_{(\ell,1)}$ can be proved in a completely similar

way). We build a wD3DH1 distinguisher $\mathcal{B}$ that inputs $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^b, \hat{g}^c, T)$ with the goal of deciding if $T = g^{abc}$ or $T \in_R \mathbb{G}$.

To this end, $\mathcal{B}$ defines $h = g^b$, $\hat{h} = \hat{g}^b$ and $\hat{V}_k = \hat{g}^c$. It picks $\alpha \xleftarrow{R} \mathbb{Z}_p$ and defines $\hat{u} = \hat{h}^{-k} \cdot \hat{g}^{\alpha}$ as well as $u = h^{-k} \cdot g^{\alpha}$, which implicitly sets $v_k = c$. This allows defining

$$\hat{H}_k = (\hat{h}^k \cdot \hat{u})^c = (\hat{g}^c)^{\alpha},$$

In addition, $\mathcal{B}$ defines $(W_0, W_1, \ldots, W_L) \in \mathbb{G}^{L+1}$ and $(\hat{W}_0, \hat{W}_1, \ldots, \hat{W}_L) \in \hat{\mathbb{G}}^{L+1}$ by setting

$$W_i = (g^b)^{\alpha_i} \cdot g^{\beta_i}, \qquad \hat{W}_i = (\hat{g}^b)^{\alpha_i} \cdot \hat{g}^{\beta_i} \quad \forall i \in \{0, \ldots, L\}$$

for randomly chosen $\alpha_0, \ldots, \alpha_L \xleftarrow{R} \mathbb{Z}_p$, $\beta_0, \ldots, \beta_L \xleftarrow{R} \mathbb{Z}_p$. Then, $\mathcal{B}$ chooses $v_i \xleftarrow{R} \mathbb{Z}_p$ for each $i \in [n] \setminus \{k\}$ and defines the rest of the evaluation key $ek$ by setting

$$\hat{V}_i = \hat{g}^{v_i}, \qquad \hat{H}_i = (\hat{h}^i \cdot \hat{u})^{v_i}, \qquad \forall i \in [n] \setminus \{k\}$$

Then, at each invocation of the $\mathsf{LAF.LTag}(tk, .)$ oracle, $\mathcal{B}$ responds as follows. At the $\mu$-th query $t_{\mathsf{a}}^{(\mu)}$, it generates a core tag $t_{\mathsf{c}}^{(\mu)}$ such that

- If $\mu < \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ contains $\{\hat{D}_i, \hat{E}_i\}_{i=1}^n$ uniformly random pairs whereas $\{R_i, \hat{S}_i\}_{i=1}^n$ are chosen as in the real algorithm sampling random tags.
- If $\mu = \ell$, $t_{\mathsf{c}}^{(\mu)} = (\{R_i, S_i, D_i, E_i\}_{i=1}^n, r_{hash})$ is generated as follows. It sets

  $$R_k = g^a, \qquad S_k = (g^a)^{\alpha}.$$

  As for indexes $i \neq k$, it chooses $r_1, \ldots, r_{k-1}, r_{k+1}, \ldots, r_n \xleftarrow{R} \mathbb{Z}_p$ and sets

  $$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n] \setminus \{k\}.$$

  It generates the pairs $\{D_i, E_i\}_{i=1}^n$ by choosing $(D_i, E_i) \xleftarrow{R} \mathbb{G}^2$ at random for each $i \in [k-1]$. The $k$-th pair $(D_k, E_k)$ is defined as

  $$D_k = T \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad E_k = g^{\rho_k}. \tag{3.10}$$

  for a randomly chosen $\rho_k \xleftarrow{R} \mathbb{Z}_p$. As for $\{D_i, E_i\}_{i=k+1}^n$, they are obtained by choosing a random $\tau = \tau[1] \ldots \tau[L] \in \{0,1\}^L$ in the range of $\mathsf{CMhash}$ and choosing $\rho_i \xleftarrow{R} \mathbb{Z}_p$ before setting

  $$\begin{aligned} D_i &= H_{\mathbb{G}}(\tau)^{\rho_i} \cdot (g^c)^{-r_i \cdot \frac{\beta_0 + \sum_{i=1}^L \beta_i \cdot \tau[i]}{\alpha_0 + \sum_{i=1}^L \alpha_i \cdot \tau[i]}} \\ E_i &= g^{\rho_i} \cdot (g^c)^{-\frac{r_i}{\alpha_0 + \sum_{i=1}^L \alpha_i \cdot \tau[i]}} \end{aligned} \tag{3.11}$$

  which can be written

  $$\begin{aligned} D_i &= g^{bc \cdot r_i} \cdot H_{\mathbb{G}}(\tau)^{\tilde{\rho}_i} = h^{v_k \cdot r_i} \cdot H_{\mathbb{G}}(\tau)^{\tilde{\rho}_i} \\ E_i &= g^{\tilde{\rho}_i} \end{aligned}$$

if we define $\tilde{\rho}_i = \rho_i - \frac{c \cdot r_i}{\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i]}$. Note that the reduction $\mathcal{B}$ fails if $\alpha_0 + \sum_{i=1}^{L} \alpha_i \cdot \tau[i] = 0$ but this only occurs with negligible chance since the coordinates $(\alpha_0, \ldots, \alpha_L) \in \mathbb{Z}_p^L$ are independent of $\mathcal{A}$'s view. Finally, $\mathcal{B}$ uses the trapdoor $td_{\mathsf{CMH}}$ of the chameleon hash function to find coins $r_{hash} \in \mathcal{R}_{\mathsf{CMH}}$ such that

$$\tau = \mathsf{CMhash}\big(hk_{hash}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i\}_{i=1}^{n}), r_{hash}\big).$$

- If $\mu > \ell$, the tags are generated as lossy tags. To this end, $\mathcal{B}$ proceeds as in the previous case, except that all elements $\{D_i, E_i\}_{i=1}^{n}$ (and not only the last $n-k$ ones) are generated as per (3.11).

It is easy to see that, if $T = g^{abc}$, the pair $(D_k, E_k)$ of (3.10) can be written

$$D_k = h^{v_k \cdot r_k} \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad E_k = g^{\rho_k},$$

meaning that $\mathcal{A}$'s view is the same as in $\mathsf{Game}_{(\ell,k-1)}$. In contrast, if $T \in_R \mathbb{G}$, then $(D_k, E_k)$ can be written

$$D_k = h^{\omega_k + v_k \cdot r_k} \cdot H_{\mathbb{G}}(\tau)^{\rho_k}, \qquad E_k = g^{\rho_k},$$

for some uniformly random $\omega_k \in_R \mathbb{Z}_p$. In this case, $\mathcal{A}$'s view corresponds to $\mathsf{Game}_{(\ell,k)}$. □
□

The evasiveness property is established by Theorem 3.3.

**Theorem 3.3.** *The above LAF provides evasiveness assuming that: (i)* CMH *is a collision-resistant chameleon hash function; (ii) The wD3DH1 and* 2-3-*CDH assumptions both hold in* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

*Proof.* Let us assume that a PPT adversary $\mathcal{A}$ can break the evasiveness property with noticeable advantage. We show that it contradicts either: (i) The indistinguishability of the scheme; (ii) The collision-resistance of the chameleon hash function; (iii) The 2-3-CDH assumption. We will prove this claim via a sequence of hybrid games.

In $\mathsf{Game}_0$, the adversary $\mathcal{A}$ proceeds as in the real evasiveness security experiment. In the final game, we show that, if the adversary can output a lossy tag, we can easily construct an algorithm breaking the 2-3-CDH assumption with non-negligible advantage.

For each $i$, we denote by $\mathsf{bad}_i$ the event that $\mathcal{A}$ manages to output a non-trivial lossy tag in $\mathsf{Game}_i$.

**$\mathsf{Game}_0$:** In this game, the adversary $\mathcal{A}$ has access to the lossy tag generation oracle $\mathsf{LAF.LTag}(tk, \cdot)$ that always outputs lossy tags. By definition,

$$\Pr[\mathsf{bad}_0] \quad = \Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot)}]. \tag{3.12}$$

**$\mathsf{Game}_1$:** In this game, we define $\mathsf{bad}_{hash}$ to be the event that the adversary $\mathcal{A}$ outputs a tag $t = ((\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash}))$ for which the corresponding chameleon hash collides with that of some tag produced by the oracle $\mathsf{LAF.LTag}(tk, \cdot)$. The only

difference between $\mathsf{Game}_1$ and $\mathsf{Game}_0$ is that $\mathsf{Game}_1$ aborts when $\mathsf{bad}_{hash}$ occurs. It is straightforward that

$$|\Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_0]| \quad = \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1]. \tag{3.13}$$

We want to use the collision resistance property of the underlying chameleon hash function to bound the probability $\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1]$. However, the lossy key generation oracle uses the trapdoor $td_{\mathsf{CMH}}$ to create lossy tags. To avoid a circularity, we consider $\mathsf{Game}'_1$, where the lossy key generation oracle always outputs injective tags instead of lossy ones. Using the indistinguishability between lossy and injective tags (established by Theorem 3.1), we have

$$|\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1] - \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}'_1]| = nQ \cdot \mathbf{Adv}^{\mathrm{wD3DH1}}(\lambda). \tag{3.14}$$

Since $\mathsf{Game}_0$ and $\mathsf{Game}_1$ only differ when $\mathsf{bad}_{hash}$ occurs in $\mathsf{Game}_1$, we can bound the probability (3.13) as

$$\begin{aligned} |\Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_0]| &= |\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1]| \\ &\leq |\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}'_1]| + nQ \cdot \mathbf{Adv}^{\mathrm{wD3DH1}}(\lambda) \end{aligned}$$

In $\mathsf{Game}'_1$, we clearly have $\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}'_1] \leq \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{CMH}}(\lambda)$: since the trapdoor of CMH is not used, we can readily build a reduction that breaks the collision-resistance of CMH out of an adversary for which $\mathsf{bad}_{hash}$ occurs with noticeable probability. This immediately implies

$$|\Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_0]| \leq \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{CMH}}(\lambda) + nQ \cdot \mathbf{Adv}^{\mathrm{wD3DH1}}(\lambda)$$

We now proceed to bound $\Pr[\mathsf{bad}_1]$ by showing that, using the adversary $\mathcal{A}$ in $\mathsf{Game}_1$, we can build an algorithm $\mathcal{B}$ breaking the 2-3-CDH assumption.

Algorithm $\mathcal{B}$ takes as input $(g^a, g^b, \hat{g}^a, \hat{g}^b)$ with the goal of computing $g^r, g^{r \cdot ab}$. To this end, $\mathcal{B}$ defines $h = g^a$. It randomly chooses a $J \xleftarrow{R} [n]$ and sets $V_J = g^b$, which implicitly sets $v_J = b$. In addition, $\mathcal{B}$ defines $(W_0, W_1, \ldots, W_L) \in \mathbb{G}^{L+1}$ and $(\hat{W}_0, \hat{W}_i, \ldots, \hat{W}_L) \in \hat{\mathbb{G}}^{L+1}$ as

$$W_i = (g^a)^{\alpha_i} \cdot g^{\beta_i} \qquad\qquad \hat{W}_i = (\hat{g}^a)^{\alpha_i} \cdot \hat{g}^{\beta_i}$$

where $\alpha_0 = -1$ and $\alpha_1, \ldots, \alpha_L \xleftarrow{R} \{-1, 0, 1\}$ and $\beta_0, \ldots, \beta_L \xleftarrow{R} \mathbb{Z}_p$.

In order to simulate the $\mathsf{LAF.LTag}$ oracle on input of $t_a$, $\mathcal{B}$ proceeds as follows:

1. For each $i \in [n]$, $\mathcal{B}$ uniformly samples $r_i \xleftarrow{R} \mathbb{Z}_p^\star$ and computes $\{R_i, S_i\}_{i=1}^n$ as in (3.3).
2. $\mathcal{B}$ samples a random $\tau$ in the range of $\mathsf{CMhash}$. For each $i \in [n] \backslash \{J\}$, it chooses $\rho_i \xleftarrow{R} \mathbb{Z}_p$ and computes

$$D_i = h^{r_i \cdot v_i} \cdot H_{\mathbb{G}}(\tau)^{\rho_i}, \qquad\qquad E_i = g^{\rho_i}.$$

3. For $i = J$, $\mathcal{B}$ aborts if $\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau[k] = 0$. Otherwise, $\mathcal{B}$ chooses $\rho_J \xleftarrow{R} \mathbb{Z}_p$ and computes $(D_J, E_J)$ as in (3.11):

$$D_J = H_{\mathbb{G}}(\tau)^{\rho_J} \cdot (V_J)^{-r_J \cdot \frac{\beta_0 + \sum_{k=1}^{L} \beta_k \cdot \tau[k]}{\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau[k]}}$$
$$E_J = g^{\rho_J} \cdot (V_J)^{-\frac{r_J}{\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau[k]}} \tag{3.15}$$

4. Next, $\mathcal{B}$ uses the trapdoor $td_{\mathsf{CMH}}$ of the chameleon hash function in order to find random coins $r_{hash} \in \mathcal{R}_{\mathsf{CMH}}$ such that

$$\tau = \mathsf{CMhash}(hk_{hash}, (t_a, \{R_i, S_i, D_i, E_i\}_{i=1}^{n}), r_{hash}).$$

5. Finally, $\mathcal{B}$ outputs $(t_{\mathsf{c}}, t_{\mathsf{a}})$ with $t_{\mathsf{c}} = (\{R_i, S_i, D_i, E_i\}_{i=1}^{n}, r_{hash})$.

As in (3.11), if we define $\tilde{\rho}_J = \rho_J - \frac{b \cdot r_J}{\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau[k]}$, we observe that (3.15) can be written as $D_J = h^{b \cdot r_J} \cdot H_{\mathbb{G}}(\tau)^{\tilde{\rho}_J}$ and $E_J = g^{\tilde{\rho}_J}$. Hence, if $\mathcal{B}$ does not abort in any query to LAF.LTag, the output distribution of $\mathcal{B}$ is identical to that of the real LAF.LTag oracle. We denote by $\mathsf{abort}_k$ the event that $\mathcal{B}$ aborts at the $k$-th query to the LAF.LTag oracle for $k \in [Q]$. Letting $t^\star = (t^\star_{\mathsf{c}}, t^\star_{\mathsf{a}})$ denote the lossy tag generated by $\mathcal{A}$, we parse $t^\star_{\mathsf{c}}$ as $t^\star_{\mathsf{c}} = (\{R^\star_i, S^\star_i, D^\star_i, E^\star_i\}_{i=1}^{n}, r^\star_{hash})$ and compute

$$\tau^\star = \mathsf{CMhash}(hk_{hash}, (t^\star_{\mathsf{a}}, \{R^\star_i, S^\star_i, D^\star_i, E^\star_i\}_{i=1}^{n}), r^\star_{hash}).$$

In the event that $\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau^\star[k] \neq 0$, $\mathcal{B}$ aborts. We denote the latter event by $\mathsf{abort}_{ch}$. If $\mathcal{B}$ did not abort (which implies $\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau^\star[k] = 0$), we have

$$\begin{aligned} D^\star_J &= h^{b \cdot r^\star_J} \cdot H_{\mathbb{G}}(\tau^\star)^{\tilde{\rho}^\star_J} \\ &= g^{ab \cdot r^\star_J} \cdot g^{(a \cdot (\alpha_0 + \sum_{k=1}^{L} \alpha_k \cdot \tau^\star[k]) + (\beta_0 + \sum_{k=1}^{L} \beta_k \cdot \tau^\star[k])) \cdot \tilde{\rho}^\star_J} \\ &= g^{ab \cdot r_J} \cdot {E^\star_J}^{\beta_0 + \sum_{k=1}^{L} \beta_k \cdot \tau^\star[k]}, \end{aligned}$$

where $E^\star_J = g^{\tilde{\rho}^\star_J}$. Finally, $\mathcal{B}$ outputs $\left(R^\star_J, \frac{D^\star_J}{{E^\star_J}^{\beta_0 + \sum_{k=1}^{L} \beta_k \cdot \tau^\star[k]}}\right)$.

Clearly, if $\mathcal{B}$ did not abort, its output $(R_J, D_J / E_J^{\beta_0 + \sum_{k=1}^{L} \beta_k \cdot \tau[k]})$ is a valid 2-3-CDH challenge. We are left with evaluating the probability that $\mathcal{B}$ aborts.

If we define the function $\alpha : \{0,1\}^L \to \mathbb{Z}$ that maps the string $m = m[1] \ldots m[L] \in \{0,1\}^L$ to $\alpha(m) = \alpha_0 + \sum_{k=1}^{L} \alpha_i \cdot m[i]$, the probability that $\mathcal{B}$ does not abort is given by

$$\begin{aligned} &\Pr[\neg\mathsf{abort}_{ch} \wedge \neg\mathsf{abort}_1 \wedge \ldots \wedge \neg\mathsf{abort}_Q] \\ &\qquad = \Pr[\alpha(\tau^\star) = 0 \wedge \alpha(\tau_1), \ldots, \alpha(\tau_Q) \neq 0], \end{aligned} \tag{3.16}$$

where $Q$ is the number of queries to LAF.LTag. and $\tau_i$ denotes the output of the chameleon hash function produced at the $i$-th LAF.LTag query. By applying known results on programmable hash functions [HK08], our choice of $\alpha_0, \alpha_1, \ldots, \alpha_L$ ensures that

$$\Pr[\neg\mathsf{abort}_{ch} \wedge \neg\mathsf{abort}_1 \wedge \ldots \wedge \neg\mathsf{abort}_Q] \;\geq\; \delta, \tag{3.17}$$

where $\delta = \Omega(Q \cdot \sqrt{L})$. Putting the above altogether, we can conclude that

$$\mathbf{Adv}^{\mathrm{eva}}(1^\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{CR}}_{\mathsf{CMH}}(\lambda) + nQ \cdot \mathbf{Adv}^{\mathrm{wD3DH1}}(\lambda) + \mathcal{O}(Q \cdot \sqrt{L}) \cdot \mathbf{Adv}^{\text{2-3-CDH}}(\lambda),$$

which yields the statement of the theorem. □ □

Recall that the wD3DH1 and 2-3-CDH assumptions are implied by the D3DH1 and D3DH2 assumptions, respectively. Theorem 3.1 and Theorem 3.3 thus guarantee the D3DH1 and D3DH2 assumptions suffice to ensure the indistinguishability and evasiveness properties of our LAF construction (indeed, chameleon hash functions also exist under these assumptions).

### 3.3.3. Towards All-But-Many Lossy Trapdoor Functions

Our LAF construction can be modified to construct an all-but-many lossy trapdoor function [Hof12]. Recall that ABM-LTFs do not require evaluations on lossy tags to always output the same information about the input: on any lossy tag, the image size is only required to be much smaller. On the other hand, ABM-LTFs require that, for any injective tag, the function be efficiently invertible using a trapdoor.

Our construction can be turned into an ABM-LTF in the following way. In the evaluation algorithm, a binary input vector $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \{0,1\}^n$ is mapped to the output $(Y_0, \ldots, Y_n) \in \mathbb{G}_T^{n+1}$, where

$$\begin{aligned} Y_0 &= \prod_{i=1}^{n} e(R_i, \hat{h})^{x_i} \\ Y_j &= \prod_{i=1}^{n} M_{i,j}^{x_i} \qquad\qquad \forall j \in [n], \end{aligned}$$

which can be written

$$\begin{aligned} Y_0 &= e(g, \hat{h})^{\sum_{i=1}^{n} r_i \cdot x_i} \\ Y_j &= e(g, \hat{h})^{\omega_j \cdot x_j + v_j \cdot \sum_{i=1}^{n} r_i \cdot x_i} \qquad\qquad \forall j \in [n]. \end{aligned}$$

Using $ik = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ as an inversion key, one can decode the $j$-th input bit as $x_j = 0$ (resp. $x_j = 1$) if $Y_j / Y_0^{v_j} = 1_{\mathbb{G}_T}$ (resp. $Y_j / Y_0^{v_j} \neq 1_{\mathbb{G}_T}$).

Unfortunately, the above ABM-LTF does not seem immediately usable in the application to selective-opening chosen-ciphertext security, which was suggested in [Hof12]. The reason is that our tags have a special and publicly recognizable structure, where $(R_i, S_i)$ both depend on the same exponent $r_i \in \mathbb{Z}_p$. In the selective-opening setting, the problem arises when the adversary chooses to corrupt some senders, at which point the reduction should reveal the random coins used to create lossy/injective tags. In our construction, this would entail to reveal $r_i \in \mathbb{Z}_p$, which is incompatible with our proofs of indistinguishability and evasiveness. In the ABM-LTF constructions of [Hof12; LSSS17], lossy tags are explainable because they are pseudorandom, which allows the reduction to pretend that they have been

randomly sampled in their ambient space. Here, the special structure of lossy/injective tags prevents us from explaining the generation of lossy tags in the same way for corrupted senders. The only apparent way to sample a pair $(R_i, S_i)$ satisfying $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$ is to choose $r_i \in \mathbb{Z}_p$ and compute $(R_i, S_i) = (g^{r_i}, (h^i \cdot u)^{r_i})$.

We thus leave it as an open problem to build an ABM-LTF with explainable linear-size tags under DDH-like assumptions.

## 3.4. A Lossy Algebraic Filter With Tight Security

In this section, we modify our first LAF construction in such a way that we can prove it tightly secure under constant-size assumptions.[2] To this end, we replace Waters signatures by a variant of the MAC described by Blazy, Kiltz and Pan [BKP14], which is itself inspired by the Naor-Reingold PRF [NR97].



### 3.4.1. A Variant of the BKP MAC

The MAC construction below is identical to the signature scheme implied by the IBE scheme of [BKP14, Appendix D] with two differences which prevent public verification in order to obtain a pseudo-random MAC instead of a digital signature. The signature scheme of [BKP14] was actually designed by transposing a pseudo-random MAC from standard DDH-hard groups to bilinear groups in order to enable public verification. Here, we cannot immediately use the MAC of [BKP14] because we need bilinear maps in the evaluation algorithm of our LAF.

In order to obtain a pseudo-random MAC, we thus modify the signature scheme of [BKP14] by introducing an additional randomizer $r \in \mathbb{Z}_p$ and an extra group element $h$, of which the discrete logarithm $\log_g(h)$ serves as a private verification key.

**Keygen**$(1^\lambda, 1^L)$: Given a security parameter $\lambda$ and a message length $L \in \mathsf{poly}(\lambda)$, choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $g, h \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.

1. Choose $\theta, \alpha, \beta \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{g}^\theta \in \hat{\mathbb{G}}$. For each $\mu \in \{0,1\}$, choose vectors $\vec{x}_\mu = (x_{1,\mu}, \ldots, x_{L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$, $\vec{y}_\mu = (y_{1,\mu}, \ldots, y_{L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$.
2. Set $v = \alpha + \theta \cdot \beta$ and $\vec{z}_\mu = \vec{x}_\mu + \theta \cdot \vec{y}_\mu \in \mathbb{Z}_p^L$. Compute $\hat{V} = \hat{g}^v$ and, for each $\mu \in \{0,1\}$, define $\vec{\hat{Z}}_\mu = (\hat{Z}_{1,\mu}, \ldots, \hat{Z}_{L,\mu}) = \hat{g}^{\vec{z}_\mu}$.

Output a secret key $\mathsf{sk}_{mac} = (\alpha, \beta, \vec{x}_0, \vec{x}_1, \vec{y}_0, \vec{y}_1, \eta)$, where $\eta = \log_g(h)$, and public parameters consisting of $\mathsf{pp} = \big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, h, \hat{g}^\theta, (\hat{V}, \vec{\hat{Z}}_0, \vec{\hat{Z}}_1)\big)$.

[2] While the assumption of Definition 2.10 is described using $O(Q)$ group elements, it tightly reduces to wD3DH1 and DDH which both take a constant number of group elements to describe.

**Mac.Sig**$(\mathsf{pp}, \mathsf{sk}_{mac}, M)$**:** To generate a MAC for $M = m[1]\dots m[L] \in \{0,1\}^L$ using $\mathsf{sk}_{mac} = (x, y, \vec{x}_0, \vec{x}_1, \vec{y}_0, \vec{y}_1, \eta)$, choose $r, \rho \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\begin{aligned}\sigma_1 &= h^{\alpha\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} x_{k,m[k]})}\\ \sigma_2 &= h^{\beta\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} y_{k,m[k]})}\\ \sigma_3 &= g^{\rho}\\ \sigma_4 &= g^{r}\end{aligned}$$

**Mac.Ver**$(\mathsf{pp}, \mathsf{sk}_{mac}, M, \sigma)$**:** Given $\mathsf{sk}_{mac} = (\alpha, \beta, \vec{x}_0, \vec{x}_1, \vec{y}_0, \vec{y}_1, \eta)$ and an $L$-bit message $M = m[1]\dots m[L]$, a purported MAC $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is accepted if and only if

$$e(\sigma_1, \hat{g}) \cdot e(\sigma_2, \hat{g}^{\theta}) = e(\sigma_4, \hat{V})^{\eta} \cdot e(\sigma_3, \prod_{k=1}^{L} \hat{Z}_{k,m[k]}). \tag{3.18}$$

We note that the verification algorithm can be modified in such a way that it does not require any pairing evaluation. The above description is just meant to simplify the presentation of the security proof of our LAF construction.

The proof is essentially identical to that of [BKP14] but we give it for completeness. We note that, in the security definitions of MACs, the adversary is generally allowed to make verification queries. Here, for simplicity, we prove unforgeability in a game where the adversary knows $\eta = \log_g(h)$, which allows it to run the verification oracle itself. This dispenses us with the need for a verification oracle.

**Lemma 3.4.** *The above construction is an unforgeable MAC assuming that the SXDH assumption holds in $(\mathbb{G}, \hat{\mathbb{G}})$. Namely, any forger $\mathcal{A}$ making $Q$ MAC queries and $Q_V$ verification queries within running time $t_{\mathcal{A}}$ has advantage at most*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{uf-mac}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_1}^{\text{DDH}_2}(\lambda) + 2L \cdot \mathbf{Adv}_{\mathcal{B}_2}^{\text{DDH}_1}(\lambda),$$

*where $\mathcal{B}_1$ and $\mathcal{B}_2$ are PPT distinguishers against the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, which run in time $t_{\mathcal{A}} + (Q + Q_V) \cdot \mathsf{poly}(\lambda)$.*

*Proof.* To prove the result, we consider a sequence of games. For each index $i$, we call $W_i$ the event that the challenger outputs 1 in $\text{Game}_i$.

**Game**$_0$**:** This is the real game MAC security game, where the adversary $\mathcal{A}$ is additionally given $\eta = \log_g(h)$ in such a way that it can run the verification algorithm (and test whether equation (3.18) holds) by itself. The challenger outputs 1 if and only if $\mathcal{A}$ eventually outputs a pair $(M^{\star}, \sigma^{\star} = (\sigma_1^{\star}, \sigma_2^{\star}, \sigma_3^{\star}, \sigma_4^{\star}))$ satisfying

$$e(\sigma_1^{\star}, \hat{g}) \cdot e(\sigma_2^{\star}, \hat{g}^{\theta}) = e(\sigma_4^{\star}, \hat{V})^{\eta} \cdot e(\sigma_3^{\star}, \prod_{k=1}^{L} \hat{Z}_{k,m^{\star}[k]}), \tag{3.19}$$

where $M^{\star} = m^{\star}[1]\dots m^{\star}[L] \in \{0,1\}^L$, although $M^{\star}$ was not previously queried to the MAC oracle. By definition, $\Pr[W_0] = \mathbf{Adv}_{\mathcal{A}}^{\text{uf-mac}}(\lambda)$.

**Game$_1$:** In this game, we modify again the verification oracle as follows. When $\mathcal{A}$ outputs a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ such that $M^\star$ was not queried to the MAC oracle but $(M^\star, \sigma^\star)$ still satisfies (3.19), the challenger checks if

$$\begin{aligned} \sigma_1^\star &= \sigma_4^{\star\eta\cdot\alpha} \cdot \sigma_3^{\star\sum_{k=1}^{L} x_{k,m^\star[k]}} \\ \sigma_2^\star &= \sigma_4^{\star\eta\cdot\beta} \cdot \sigma_3^{\star\sum_{k=1}^{L} y_{k,m^\star[k]}}. \end{aligned} \tag{3.20}$$

We call $E_1$ the event that equalities (3.20) are satisfied. If they are not satisfied, the challenger outputs 0. Otherwise, it outputs 1 as it did in Game$_0$. If we denote by $E_0$ the analogue of event $E_1$ in Game$_0$, we have

$$\begin{aligned} \Pr[W_0] &= \Pr[W_0 \wedge E_0] + \Pr[W_0 \wedge \neg E_0] \\ &= \Pr[W_1 \wedge E_1] + \Pr[W_0 \wedge \neg E_0] = \Pr[W_1] + \Pr[W_0 \wedge \neg E_0] \end{aligned}$$

since $\Pr[W_1 \wedge \neg E_1] = 0$. Lemma 3.5 shows that event $W_0 \wedge \neg E_0$ would contradict the DDH assumption in $\hat{\mathbb{G}}$: namely, we have $\Pr[W_0 \wedge \neg E_0] \leq \mathbf{Adv}^{\mathrm{DDH}_2}(\lambda)$, which implies $|\Pr[W_1] - \Pr[W_0]| \leq \mathbf{Adv}^{\mathrm{DDH}_2}(\lambda)$.

We now use a sub-sequence of $L$ hybrid games over the input bits of queried messages. For convenience, we define Game$_{2.0}$ to be identical to Game$_1$.

**Game$_{2.i}$ $(1 \leq i \leq L)$:** In this sub-sequence of games, we modify the key generation phase and the MAC oracle in the following way.

- At the beginning of the game, the challenge defines $\hat{V} = \hat{g}^v$ for a random $v \xleftarrow{R} \mathbb{Z}_p$.
- MAC queries are handled as follows. Let $R : \{0,1\}^i \to \mathbb{Z}_p$ be a truly random function mapping $i$-bit input to $\mathbb{Z}_p$. At each message $M$ queried by $\mathcal{A}$, the challenger computes $(\sigma_3, \sigma_4) = (g^\rho, g^r)$ for random $\rho, r \xleftarrow{R} \mathbb{Z}_p$. Then, it outputs $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, where

$$\begin{aligned} \sigma_1 &= h^{(v-\theta\cdot R(m[1]\ldots m[i]))\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} x_{k,m[k]})} \\ \sigma_2 &= h^{R(m[1]\ldots m[i])\cdot r} \cdot g^{\rho\cdot(\sum_{k=1}^{L} y_{k,m[k]})} \end{aligned}$$

When the adversary outputs $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (3.19) for a new message $M^\star$, the challenger checks if the following equalities are satisfied:

$$\begin{aligned} \sigma_1^\star &= \sigma_4^{\star\eta\cdot(v-\theta\cdot R(m^\star[1]\ldots m^\star[i]))} \cdot \sigma_3^{\star\sum_{k=1}^{L} x_{k,m^\star[k]}} \\ \sigma_2^\star &= \sigma_4^{\star\eta\cdot R(m^\star[1]\ldots m^\star[i])} \cdot \sigma_3^{\star\sum_{k=1}^{L} y_{k,m^\star[k]}}. \end{aligned} \tag{3.21}$$

If so, the challenger outputs 1. Otherwise, it outputs 0. Lemma 3.6 shows that Game$_{2.i}$ is indistinguishable from Game$_{2.(i-1)}$ under the DDH assumption in $\mathbb{G}$. Namely, $|\Pr[W_{2.i}] - \Pr[W_{2.(i-1)}]| \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda)$.

In $\mathsf{Game}_{2.L}$, we claim that $\Pr[W_{2.L}] = 1/p$. Indeed, the equalities (3.21) can only hold by pure chance when $i = L$ because $m^\star[1]\ldots m^\star[L]$ was never involved in an output of the MAC oracle. Hence, the random function output $R(m^\star[1]\ldots m^\star[L])$ is perfectly independent of $\mathcal{A}$'s view. Since $\Pr[W_{2.0}] = \Pr[W_1]$, we obtain the claimed upper bound for $\Pr[W_0]$. □ □

**Lemma 3.5.** *In* $\mathsf{Game}_0$*, we have* $\Pr[W_0 \wedge \neg E_0] \leq \mathbf{Adv}^{\mathrm{DDH}_2}(\lambda)$.

*Proof.* Towards a contradiction, let us assume that, in $\mathsf{Game}_1$, the adversary $\mathcal{A}$ can output a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (3.19) but not (3.20). We construct a distinguisher $\mathcal{B}$ for the DDH assumption in $\hat{\mathbb{G}}$. Our distinguisher $\mathcal{B}$ takes as input $(\hat{g}, \hat{g}^\theta, \hat{g}^\omega, \hat{T}) \in \hat{\mathbb{G}}^4$ and decides if $\hat{T} = \hat{g}^{\alpha\cdot\omega}$ or $\hat{T} \in_R \hat{\mathbb{G}}$. To this end, $\mathcal{B}$ will compute a pair of the form $(w, w^\theta) \in \mathbb{G}^2$ with $w \neq 1_{\mathbb{G}}$, which allows solving the given DDH instance in $\hat{\mathbb{G}}$ by testing if $e(w, \hat{T}) = e(w^\theta, \hat{g}^\omega)$. Indeed, the latter equality holds if and only if $\hat{T} = \hat{g}^{\alpha\cdot\omega}$.

The reduction $\mathcal{B}$ runs the real key generation algorithm and answers all MAC and verification queries exactly as in $\mathsf{Game}_1$. By hypothesis, $\mathcal{B}$ has non-negligible probability of outputting a pair $(M^\star, \sigma^\star = (\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star))$ satisfying (3.19) although

$$\sigma_1^\star \neq \sigma_4^{\star\eta\cdot\alpha} \cdot \sigma_3^{\star\sum_{k=1}^{L} x_{k,m^\star[k]}}, \qquad \sigma_2^\star \neq \sigma_4^{\star\eta\cdot\beta} \cdot \sigma_3^{\star\sum_{k=1}^{L} y_{k,m^\star[k]}}.$$

At this point, $\mathcal{B}$ uses $\mathsf{sk}_{mac}$ to construct a different valid MAC $(\sigma_1', \sigma_2', \sigma_3^\star, \sigma_4^\star)$ satisfying (3.19) and such that $(\sigma_1', \sigma_2') \neq (\sigma_1^\star, \sigma_2^\star)$. Namely, $\mathcal{B}$ computes

$$\sigma_1' = \sigma_4^{\star\eta\cdot\alpha} \cdot \sigma_3^{\star\sum_{k=1}^{L} x_{k,m^\star[k]}}, \qquad \sigma_2' = \sigma_4^{\star\eta\cdot\beta} \cdot \sigma_3^{\star\sum_{k=1}^{L} y_{k,m^\star[k]}}.$$

By dividing the two verification equations for $(\sigma_1', \sigma_2', \sigma_3^\star, \sigma_4^\star)$ and $(\sigma_1^\star, \sigma_2^\star, \sigma_3^\star, \sigma_4^\star)$, we get

$$e(\sigma_1^\star/\sigma_1', \hat{g}) \cdot e(\sigma_2^\star/\sigma_2', \hat{g}^\theta) = 1_{\mathbb{G}_T}$$

meaning that $\sigma_1^\star/\sigma_1' = (\sigma_2'/\sigma_2^\star)^\theta$. Since $\sigma_1^\star \neq \sigma_1'$, this provides $\mathcal{B}$ with a non-trivial pair $(w, w^\theta) = \left(\sigma_2'/\sigma_2^\star, \sigma_1^\star/\sigma_1'\right)$, which is sufficient to solve DDH in $\hat{\mathbb{G}}$. □ □

**Lemma 3.6.** *Under the DDH assumption in* $\mathbb{G}$*, the challenger outputs* 1 *with about the same probabilities in* $\mathsf{Game}_{3.(i-1)}$ *and* $\mathsf{Game}_{3.i}$*. We have*

$$|\Pr[W_{2.i}] - \Pr[W_{2.(i-1)}]| \leq 2 \cdot \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda).$$

*Proof.* Assuming the existence of an adversary $\mathcal{A}$ that can distinguish between $\mathsf{Game}'_{2.(i-1)}$ and $\mathsf{Game}'_{2.i}$, we will build a DDH distinguisher $\mathcal{B}$. Our distinguisher $\mathcal{B}$ inputs a DDH instance $(g, g^a, g^b, T) \in \mathbb{G}^4$ and decides whether $T = g^{ab}$ or $T \in_R \mathbb{G}$. To do this, $\mathcal{B}$ flips a random coin $\gamma \xleftarrow{R} \{0,1\}$ and uses a random function $R' : \{0,1\}^{i-1} \to \mathbb{Z}_p$, which is lazily defined as the adversary makes queries. Using $R'$, $\mathcal{B}$ defines another random function $R : \{0,1\}^i \to \mathbb{Z}_p$ as

$$R(m[1]\ldots m[i]) = \begin{cases} R(m[1]\ldots m[i-1]) & m[i] = \gamma \\ R(m[1]\ldots m[i-1]) + R'(m[1]\ldots m[i-1]) & m[i] = 1-\gamma \end{cases}.$$

We now consider the output of MAC queries. Implicitly, $\mathcal{B}$ defines $x_{i,1-\gamma}$ and $y_{i,1-\gamma}$ as $x_{i,1-\gamma} = x'_{i,1-\gamma} + \theta(1-a) \cdot y'_{i,1-\gamma}$ and $y_{i,1-\gamma} = a \cdot y'_{i,1-\gamma}$, where $x'_{i,1_\gamma}, y'_{i,1-\gamma} \xleftarrow{R} \mathbb{Z}_p$. Note that the only value in public parameter that depends on $x_{i,1-\gamma}$ and $y_{i,1-\gamma}$ is

$$\hat{Z}_{i,1-\gamma} = \hat{g}^{x_{i,1-\gamma}+\theta \cdot y_{i,1-\gamma}} = \hat{g}^{x'_{i,1-\gamma}+\theta \cdot y'_{i,1-\gamma}},$$

so that $\hat{Z}_{i,1-\gamma}$ is computable from $(x'_{i,1-\gamma}, y'_{i,1-\gamma}) \in \mathbb{Z}_p^2$. The remaining secret key components are chosen as in the real key generation algorithm, by sampling $\eta, \alpha, \beta \xleftarrow{R} \mathbb{Z}_p$, $x_{i,\gamma}, y_{i,\gamma} \xleftarrow{R} \mathbb{Z}_p$ and $x_{k,b}, y_{k,b} \xleftarrow{R} \mathbb{Z}_p$ for each $k \in [L] \setminus \{i\}$, $b \in \{0,1\}$.

Then, $\mathcal{B}$ simulates the responses to MAC queries in the following way.

1. From $(A = g^a, B = g^b, T)$, $\mathcal{B}$ uses the random self-reducibili1ty of DDH assumption to generate a fresh pair $(B_{m|_{i-1}}, T_{m|_{i-1}})$ for each value of $m|_{i-1} = m[1] \ldots m[i-1] \in \{0,1\}^{i-1}$ in such a way that, if $(A, B, T)$ is a DDH tuple, so is $(A, B_{m|_{i-1}}, T_{m|_{i-1}})$. Otherwise, $B_{m|_{i-1}} \in_R \mathbb{G}$ and $T_{m|_{i-1}} \in_R \mathbb{G}$ are i.i.d. For convenience, we may associate each string $m|_{i-1} \in \{0,1\}^i$ with a tuple

$$(A, B_{m|_{i-1}}, T_{m|_{i-1}}) = \left(g^a, g^{b_{m|_{i-1}}}, g^{a \cdot b_{m|_{i-1}} + e_{m|_{i-1}}}\right)$$

where either $e_{m|_{i-1}} = 0$ or $e_{m|_{i-1}} \in_R \mathbb{Z}_p$. Note that the pairs $(B_{m|_{i-1}}, T_{m|_{i-1}})$ can be sampled lazily by having $\mathcal{B}$ initially generate $Q$ pairs since at most $Q$ distinct prefixes $m|_{i-1}$ can occur in all MAC queries.

2. For each message $M$ queried by $\mathcal{A}$, $\mathcal{B}$ randomly chooses $r, d \xleftarrow{R} \mathbb{Z}_p$ and computes $\sigma$ in the following way.

$$\begin{aligned}
\sigma_1 &= h^{(v-\theta R(m[1]\ldots m[i-1])) \cdot r} \cdot (B_{m|_i}^r \cdot g^d)^{x'_{i,m[i]} + \theta y'_{i,m[i]}} \\
&\qquad \cdot (T^r \cdot A^d)^{-\theta y'_{i,m[i]}} \cdot (B_{m|_{i-1}}^r \cdot g^d)^{\sum_{k=1 \wedge k \neq i}^{L} x_{k,m[k]}}, \\
\sigma_2 &= h^{R(m[1]\ldots m[i-1]) \cdot r} \cdot (T^r \cdot A^d)^{y'_{i,m[i]}} \cdot (B_{m|_{i-1}}^r \cdot g^d)^{\sum_{k=1 \wedge k \neq i}^{L} y_{k,m[k]}}, \\
\sigma_3 &= B_{m|_{i-1}}^r \cdot g^d, \\
\sigma_4 &= g^r.
\end{aligned}$$

We observe that, if we set $\rho = b_{m|_{i-1}} \cdot r + d$, the above equations can be written as

$$\begin{aligned}
\sigma_1 &= h^{(v-\theta R(m[1]\ldots m[i-1])) \cdot r} \cdot g^{\rho \cdot (x'_{i,m[i]} + \theta(1-a) \cdot y'_{i,m[i]})} \\
&\qquad \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} x_{k,m[k]}} \cdot g^{-e_{m|_{i-1}} \cdot y_{i,m[i]} \cdot r \cdot \theta} \\
&= h^{(v-\theta \cdot R(m[1]\ldots m[i-1])) \cdot r} \cdot g^{\rho \cdot x_{i,m[i]}} \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} x_{k,m[k]}} \cdot g^{-e_{m|_{i-1}} \cdot y'_{i,m[i]} \cdot r \cdot \theta} \\
\sigma_2 &= h^{R(m[1]\ldots m[i-1]) \cdot r} \cdot g^{\rho \cdot a \cdot y'_{i,m[i]}} \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} y_{k,m[k]}} \cdot g^{e_{m|_{i-1}} \cdot y_{i,m[i]} \cdot r} \\
&= h^{R(m[1]\ldots m[i-1]) \cdot r} \cdot g^{\rho \cdot y_{i,m[i]}} \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} y_{k,m[k]}} \cdot g^{e_{m|_{i-1}} \cdot y_{i,m[i]} \cdot r}
\end{aligned}$$

If $(A, B_{m|_{i-1}}, T_{m|_{i-1}})$, is a Diffie-Hellman tuple (i.e., if $e_{m|_{i-1}} = 0$), the output distribution is the same as in $\mathsf{Game}_{2.(i-1)}$. In contrast, if $e_{m|_{i-1}} \in_R \mathbb{Z}_p$, we have

$$\begin{aligned}
\sigma_1 &= h^{(v-\theta \cdot R(m[1]\ldots m[i])) \cdot r} \cdot g^{\rho \cdot x_{i,m[i]}} \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} x_{k,m[k]}} \\
\sigma_2 &= h^{R(m[1]\ldots m[i]) \cdot r} \cdot g^{\rho \cdot y_{i,m[i]}} \cdot g^{\rho \cdot \sum_{k=1 \wedge k \neq i}^{L} y_{k,m[k]}}
\end{aligned}$$

where the random function $R : \{0,1\}^i \to \mathbb{Z}_p$ is defined using

$$R'(m[1]\dots m[i-1]) = g^{\frac{e_{m|_{i-1}} \cdot y_{i,m[i]}}{\eta}}.$$

In this case, the output distribution of the MAC oracle is identical to that of $\mathsf{Game}_{2.i}$.

If the adversary chooses to forge on a message $m^\star[1]\dots m^\star[L]$ such that $m^\star[i] = 1-\gamma$ (which occurs with probability $1/2$), then $\mathcal{B}$ aborts and outputs a random bit. If $m^\star[i] = \gamma$, we have

$$R(m^\star[1]\dots m^\star[i]) = R(m^\star[1]\dots m^\star[i-1])$$

by the definition of $R$. Since $\mathcal{B}$ knows $y_{i,m^\star[i]} = y_{i,\gamma}$, it can check if

$$\sigma_2^\star = \sigma_4^{\star\eta \cdot R(m^\star[1]\dots m^\star[i-1])} \cdot \sigma_3^{\star \cdot \sum_{k=1}^{L} y_{k,m^\star[k]}}$$

and return 1 if and only if this equality is satisfied. We thus conclude that $|\Pr[W_{2.i}] - \Pr[W_{2.(i-1)}]| \leq 2 \cdot \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda)$, as claimed. □ □

### 3.4.2. The LAF Construction

In order to apply a hybrid argument in our proof of indistinguishability, we need to use $n$ instances of the MAC of Section 3.4.1, each of which has its own secret key $\mathsf{sk}_{mac,j}$ and its own set of public parameters $\mathsf{pp}_j = \big(g, \hat{g}, h, \hat{g}^{\theta_j}, (\hat{V}_j, \vec{\hat{Z}}_{j,0}, \vec{\hat{Z}}_{j,1})\big)$. As a result, we need an evaluation key containing $\Theta(n \cdot L)$ group elements. We leave it as an open problem to shorter the evaluation while retaining tight security and short tags.

**Key generation.** $\mathsf{LAF.Gen}(1^\lambda)$ conducts the following steps.

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with generators $g, h \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$ and let $\eta = \log_g(h)$.
2. Choose a chameleon hash function $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$, where the hashing algorithm $\mathsf{CMhash} : \{0,1\}^* \times \mathcal{R}_{hash} \to \{0,1\}^L$ has output length $L \in \mathsf{poly}(\lambda)$. Generate a pair $(hk_{\mathsf{CMH}}, td_{\mathsf{CMH}}) \leftarrow \mathsf{CMKg}(1^\lambda)$ made of a hashing key $hk_{\mathsf{CMH}}$ and a trapdoor $td_{\mathsf{CMH}}$.
3. Generate $n$ keys for the MAC of Section 3.4.1 which all share the same parameters $g, h \in \mathbb{G}$, $\hat{g} \in \hat{\mathbb{G}}$. Namely, for each $j \in [n]$, conduct the following steps.
   a. Choose $\theta_j \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{g}^{\theta_j} \in \hat{\mathbb{G}}$.
   b. For each $\mu \in \{0,1\}$, choose vectors $\vec{x}_{j,\mu} = (x_{j,1,\mu}, \dots, x_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$ and $\vec{y}_{j,\mu} = (y_{j,1,\mu}, \dots, y_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$.
   c. Compute $\vec{z}_{j,\mu} = \vec{x}_{j,\mu} + \theta_j \cdot \vec{y}_{j,\mu}$ and $\vec{\hat{Z}}_{j,\mu} = \hat{g}^{\vec{z}_{j,\mu}} = (\hat{g}^{z_{j,1,\mu}}, \dots, g^{z_{j,L,\mu}})$ for each $\mu \in \{0,1\}$.
   d. Choose $\alpha_j, \beta_j \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$.

e. Define $\mathsf{sk}_{mac,j} = (\alpha_j, \beta_j, \vec{x}_{j,0}, \vec{x}_{j,1}, \vec{y}_{j,0}, \vec{y}_{j,1})$.

4. Choose $u \xleftarrow{R} \mathbb{G}$ and $\hat{h}, \hat{u} \xleftarrow{R} \hat{\mathbb{G}}$ subject to the constraints $\log_g(h) = \log_{\hat{g}}(\hat{h})$ and $\log_g(u) = \log_{\hat{g}}(\hat{u})$.
5. Define

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j} \qquad \forall j \in [n].$$

6. Output the evaluation key $ek$ and the lossy tag generation key $tk$, which consist of

$$\begin{aligned} ek &:= \Big(g,\ h,\ u,\ \hat{g},\ \hat{h},\ \hat{u},\ \{\hat{g}^{\theta_j}\}_{j=1}^n, \{\hat{\vec{Z}}_{j,\mu}\}_{j\in[n],\mu\in\{0,1\}},\ \{\hat{V}_j, \hat{H}_j\}_{j=1}^n, hk_{\mathsf{CMH}}\Big), \\ tk &:= (\{\mathsf{sk}_{mac,j}\}_{j=1}^n, \eta, td_{\mathsf{CMH}}). \end{aligned}$$

The tag space $\mathcal{T} = \mathcal{T}_\mathsf{c} \times \mathcal{T}_{aux}$ is defined as a product of $\mathcal{T}_\mathsf{a} = \{0,1\}^*$ and

$$\begin{aligned} \mathcal{T}_\mathsf{c} := \{(\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash}) \mid r_{hash} \in \mathcal{R}_{hash} \ \wedge \\ \forall i \in [n] : (R_i, S_i, D_i, E_i, F_i) \in \mathbb{G}^5 \ \wedge \ e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})\}. \end{aligned}$$

The range of the function family is $\mathsf{Rng}_\lambda = \mathbb{G}_T^{n+1}$ and its domain is $\mathbb{Z}_p^n$.

**Lossy tag generation.** $\mathsf{LAF.LTag}(tk, t_\mathsf{a})$ takes in an auxiliary tag component $t_\mathsf{a} \in \{0,1\}^*$ and uses $tk = (\{\mathsf{sk}_{mac,j}\}_{j=1}^n, \eta)$ to generate a lossy tag as follows.

1. For each $i \in [n]$, choose $r_i \xleftarrow{R} \mathbb{Z}_p$ and compute

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n]. \tag{3.22}$$

2. Choose a random string $\tau \in \{0,1\}^L$ in the range of CMhash. Then, for each $i \in [n]$, choose $\rho_i \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\begin{aligned} D_i &= h^{\alpha_i \cdot r_i} \cdot g^{\rho_i \cdot (\sum_{k=1}^L x_{i,k,\tau[k]})}, \\ E_i &= h^{\beta_i \cdot r_i} \cdot g^{\rho_i \cdot (\sum_{k=1}^L y_{i,k,\tau[k]})}, \qquad \forall i \in [n] \\ F_i &= g^{\rho_i}. \end{aligned} \tag{3.23}$$

3. Use the trapdoor $td_{\mathsf{CMH}}$ of the chameleon hash function to find random coins $r_{hash} \in \mathcal{R}_{hash}$ such that

$$\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_\mathsf{a}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash}) \in \{0,1\}^L.$$

4. Output the tag $t = (t_\mathsf{c}, t_\mathsf{a})$, where $t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash})$.

Each lossy tag corresponds to a matrix $\left(M_{i,j}\right)_{i,j\in[n]} = \left(e(g,\hat{h})^{r_i\cdot(\alpha_j+\theta_j\cdot\beta_j)}\right)_{i,j}$, which forms a rank-1 matrix in the exponent. Its diagonal entries consist of

$$M_{i,i} = \frac{e(D_i,\hat{g})\cdot e(E_i,\hat{g}^{\theta_i})}{e(F_i,\prod_{k=1}^{L}\hat{Z}_{i,k,\tau[k]})} = e(g,\hat{h})^{r_i\cdot(\alpha_i+\theta_i\cdot\beta_i)} \qquad \forall i\in[n], \tag{3.24}$$

while its non-diagonal entries

$$\begin{aligned} M_{i,j} &= \left(\frac{e(R_i,\hat{H}_j)}{e(S_i,\hat{V}_j)}\right)^{1/(j-i)} \\ &= e(g,\hat{h})^{r_i\cdot(\alpha_j+\theta_j\cdot\beta_j)} \qquad \forall (i,j)\in[n]\times[n]\setminus\{(i,i)\}_{i=1}^{n}, \end{aligned} \tag{3.25}$$

are obtained by pairing tag component $(R_i, S_i)$ with evaluation key components $(\hat{V}_j, \hat{H}_j)$.

**Random Tags.** A random tag can be publicly sampled as follows.

1. For each $i\in[n]$, choose $r_i \xleftarrow{R} \mathbb{Z}_p$ and compute $\{R_i,S_i\}_{i=1}^{n}$ as in (3.22).
2. For each $i\in[n]$, choose $(D_i,E_i,F_i)\xleftarrow{R}\mathbb{G}^3$ uniformly at random.
3. Choose $r_{hash}\xleftarrow{R}\mathcal{R}_{hash}$.

Output the tag $t=(t_{\mathsf{c}},t_{\mathsf{a}})$, where $t_{\mathsf{c}}=(\{R_i,S_i,D_i,E_i,F_i\}_{i=1}^{n},r_{hash})$.

We note that, in both random and lossy tags, we have $e(R_i,\hat{u}^i\cdot\hat{h})=e(S_i,\hat{g})$ for all $i\in[n]$, so that elements of $\mathcal{T}$ are publicly recognizable.

**Evaluation.** $\mathsf{LAF.Eval}(ek,t,\mathbf{x})$ takes in the input $\mathbf{x}\in\mathbb{Z}_p^n$ and the tag $t=(t_{\mathsf{c}},t_{\mathsf{a}})$. It parses $t_{\mathsf{c}}$ as $(\{R_i,S_i,D_i,E_i,F_i\}_{i=1}^{n},r_{hash})$ and does the following.

1. Return $\perp$ if there exists $i\in[n]$ such that $e(R_i,\hat{h}^i\cdot\hat{u})\neq e(S_i,\hat{g})$.
2. Compute the matrix $\left(M_{i,j}\right)_{i,j\in[n]}\in\mathbb{G}_T^{n\times n}$ as

$$M_{i,i} = \frac{e(D_i,\hat{g})\cdot e(E_i,\hat{g}^{\theta_i})}{e(F_i,\prod_{k=1}^{L}\hat{Z}_{i,k,\tau[k]})} \qquad \forall i\in[n] \quad , \tag{3.26}$$

where $\tau=\mathsf{CMhash}(hk_{\mathsf{CMH}},(t_{\mathsf{a}},\{R_i,S_i,D_i,E_i,F_i\}_{i=1}^{n}),r_{hash})\in\{0,1\}^L$, and

$$M_{i,j} = \left(\frac{e(R_i,\hat{H}_j)}{e(S_i,\hat{V}_j)}\right)^{1/(j-i)} \qquad \forall (i,j)\in[n]\times[n]\setminus\{(i,i)\}_{i=1}^{n}, \tag{3.27}$$

Since $R_i=g^{r_i}$ and $S_i=(h^i\cdot u)^{r_i}$ for some $r_i\in\mathbb{Z}_q$, we have

$$\begin{aligned} M_{i,i} &= e(g,\hat{h})^{r_i\cdot(\alpha_i+\theta_i\cdot\beta_i)+\omega_i}, \qquad \forall i\in[n] \\ M_{i,j} &= e(g,\hat{h})^{r_i\cdot(\alpha_j+\theta_j\cdot\beta_j)}, \qquad \forall i\neq j, \end{aligned} \tag{3.28}$$

for some vector $(\omega_1,\ldots,\omega_n)^\top\in\mathbb{Z}_p^n$ that only contains non-zero entries if $t=(t_{\mathsf{c}},t_{\mathsf{a}})$ is injective.

3. Compute the vector $\left(V_{T,j}\right)_{j\in[n]}$ as $V_{T,j} = e(h, \hat{V}_j) = e(g, \hat{h})^{\alpha_j + \theta_j \cdot \beta_j}$ for each $j \in [n]$.
4. Use the input $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{Z}_p^n$ to compute

$$\begin{aligned} Y_0 &= \prod_{j=1}^{n} V_{T,j}^{x_j} \\ Y_i &= \prod_{j=1}^{n} M_{i,j}^{x_j} \qquad \forall i \in [n] \end{aligned} \tag{3.29}$$

and output $\mathbf{Y} = (Y_0, Y_1, \ldots, Y_n)^\top \in \mathbb{G}_T^{n+1}$.

The lossiness/injectivity properties can be analyzed exactly in the same way as in the construction of Section 3.3. Indeed, by defining $v_j = \alpha_j + \theta_j \cdot \beta_j$ for each $j \in [n]$, we find that $\{\hat{V}\}_{j=1}^n$ and $(M_{ij})_{i,j\in[n]}$ are distributed as in Section 3.3.



### 3.4.3. Security

**Theorem 3.7.** *The above LAF provides indistinguishability assuming that the wD3DH1 assumption holds in $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ and that the DDH assumptions holds in $\mathbb{G}$. The advantage of any PPT distinguisher $\mathcal{A}$ making $Q$ queries within time $t_\mathcal{A}$ is bounded by*

$$\mathbf{Adv}^{\text{indist}}(\lambda) \leq n \cdot (\mathbf{Adv}_{\mathcal{B}_1}^{\text{wD3DH1}}(\lambda) + \mathbf{Adv}_{\mathcal{B}_2}^{\text{DDH}_1}(\lambda))$$

*for PPT algorithm $\mathcal{B}_1$, $\mathcal{B}_2$ running in time $t_\mathcal{A} + Q \cdot \mathsf{poly}(\lambda)$.*

*Proof.* We define a sequence of hybrid games. In $\text{Game}_0$, the adversary has access to the real oracle $\mathsf{LAF.LTag}(tk, .)$ oracle that always outputs lossy tags. In $\text{Game}_n$, the adversary is given access to an oracle $\mathcal{O}_{\mathcal{T}_\mathsf{c}}(.)$ that always outputs random tags in the tag space $\mathcal{T}$.

**$\text{Game}_\xi$'** $(1 \leq \xi \leq n)$**:** The adversary interacts with an oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$ that outputs tags $t = (t_\mathsf{c}, t_\mathsf{a})$ with the following hybrid distribution. In the core component $t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash})$, the first $\xi - 1$ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=1}^{\xi}$ of $t_\mathsf{c}$ are random group elements satisfying the equality $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$. The last $n - \xi$ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=\xi+1}^n$ are generated exactly as in lossy tags. The $\xi$-th tuple $(R_\xi, S_\xi, D_\xi, E_\xi, F_\xi)$ has a special distribution where $e(R_\xi, \hat{h}^\xi \cdot \hat{u}) = e(S_\xi, \hat{g})$, $D_\xi$ is completely random in $\mathbb{G}$ and

$$\begin{aligned} E_\xi &= h^{\beta_\xi \cdot \log_g(R_\xi)} \cdot g^{\rho_\xi \cdot \sum_{k=1}^{L} y_{\xi,k,\tau[k]}}, \\ F_\xi &= g^{\rho_\xi} \end{aligned}$$

**$\text{Game}_\xi$** $(1 \leq \xi \leq n)$**:** The adversary interacts with an oracle $\mathsf{LAF.LTag}^{(\ell,k)}(tk, .)$ that outputs $t = (t_\mathsf{c}, t_\mathsf{a})$ such that the first $\xi$ tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=1}^{\xi}$ of $t_\mathsf{c}$ are random subject to the constraint $e(R_i, \hat{h}^i \cdot \hat{u}) = e(S_i, \hat{g})$ while $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=\xi+1}^n$ are generated as in lossy tags.

For each index $\xi \in [n]$, Lemma 3.8 shows that $\mathsf{Game}'_\xi$ is computationally indistinguishable from $\mathsf{Game}_{\xi-1}$ if the R-wD3DH1 assumption holds. In a second step, Lemma 3.9 shows that $\mathsf{Game}'_\xi$ is indistinguishable from $\mathsf{Game}_\xi$ under the DDH assumption in $\mathbb{G}$. By applying Lemma 2.1, we obtain that the scheme provides indistinguishability under tight reductions from the hardness of wD3DH1 and that of the DDH problem in $\mathbb{G}$. □ □

**Lemma 3.8.** $\mathsf{Game}'_\xi$ *is computationally indistinguishable from* $\mathsf{Game}_{\xi-1}$ *under the R-wD3DH1 assumption. The advantage of any PPT distinguisher between the two games can be bounded by* $\mathbf{Adv}^{\xi'\text{-}(\xi-1)}(\lambda) \leq \mathbf{Adv}^{\text{R-wD3DH1}}(\lambda)$.

*Proof.* Let us assume that there exists $\xi \in [n]$ such that the adversary $\mathcal{A}$ can distinguish $\mathsf{Game}'_\xi$ from $\mathsf{Game}_{\xi-1}$ with non-negligible advantage. We build a R-wD3DH1 distinguisher $\mathcal{B}$ that takes as input $\{(g, \hat{g}, g^{a_i}, g^b, g^c, \hat{g}^b, \hat{g}^c, T_i)\}_{i=1}^Q$ with the goal of deciding if $T_i = g^{a_i bc}$ for each $i \in [Q]$ or if $\{T_i\}_{i=1}^Q$ are all independent and uniformly distributed over $\mathbb{G}$.

To this end, $\mathcal{B}$ defines $h = g^b$, $\hat{h} = \hat{g}^b$. It also picks $\theta'_\xi, \beta'_\xi \xleftarrow{R} \mathbb{Z}_p$ uniformly and sets

$$\hat{g}^{\theta_\xi} = (\hat{g}^b)^{\theta'_\xi}, \qquad \hat{V}_\xi = (\hat{g})^c \cdot \hat{g}^{\theta'_\xi \cdot \beta'_\xi},$$

which implicitly defines

$$\alpha_\xi = c, \qquad \beta_\xi = \beta'_\xi / b, \qquad \theta_\xi = b \cdot \theta'_\xi.$$

It chooses $\nu \xleftarrow{R} \mathbb{Z}_p$ and defines $\hat{u} = \hat{h}^{-\xi} \cdot \hat{g}^\nu$ as well as $u = h^{-\xi} \cdot g^\nu$. This allows defining

$$\hat{H}_\xi = (\hat{h}^\xi \cdot \hat{u})^{c+\theta'_\xi \cdot \beta'_\xi} = (\hat{V}_\xi)^\nu,$$

For all indexes $j \in [n] \setminus \{\xi\}$, it chooses $\alpha_j, \beta_j, \theta_j \xleftarrow{R} \mathbb{Z}_p$ and faithfully computes $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$ and

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j}.$$

Then, it constructs the MAC secret keys $\{\vec{x}_{j,\mu}, \vec{y}_{j,\mu}\}_{j=1}^n$ for randomly chosen vectors $\vec{x}_{j,\mu} = (x_{j,1,\mu}, \ldots, x_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$, $\vec{y}_{j,\mu} = (y_{j,1,\mu}, \ldots, y_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$. For each $j \in [n]$, it defines

$$\begin{aligned}
\hat{\vec{Y}}_{j,\mu} &= (\hat{Y}_{j,1,\mu}, \ldots, \hat{Y}_{j,L,\mu}) = \hat{g}^{\vec{y}_{j,\mu}}, & \vec{Y}_{j,\mu} &= (Y_{j,1,\mu}, \ldots, Y_{j,L,\mu}) = g^{\vec{y}_{j,\mu}} \\
\hat{\vec{X}}_{j,\mu} &= (\hat{X}_{j,1,\mu}, \ldots, \hat{X}_{j,L,\mu}) = \hat{g}^{\vec{x}_{j,\mu}}, & \vec{X}_{j,\mu} &= (X_{j,1,\mu}, \ldots, X_{j,L,\mu}) = g^{\vec{x}_{j,\mu}}.
\end{aligned}$$

Then, it computes

$$\begin{aligned}
\hat{\vec{Z}}_{j,\mu} &= \hat{\vec{X}}_{j,\mu} \cdot \hat{\vec{Y}}_{j,\mu}^{\theta_j} && \forall j \in [n] \setminus \{\xi\} \\
\hat{\vec{Z}}_{\xi,\mu} &= \hat{\vec{X}}_{\xi,\mu} \cdot (\hat{g}^b)^{\vec{y}_{\xi,\mu} \cdot \theta'_\xi}
\end{aligned}$$

At the $t$-th invocation of the $\mathsf{LAF.LTag}(tk, .)$ oracle, $\mathcal{B}$ sets

$$R_\xi = g^{a_t}, \qquad S_\xi = (g^{a_t})^\nu = (h^\xi \cdot u)^{a_t},$$

where $g^{a_t}$ is fetched from the $t$-th input tuple $(g, \hat{g}, g^{a_t}, g^b, g^c, \hat{g}^b, \hat{g}^c, T_t)$. For all indexes $i \neq \xi$, it chooses $r_1, \ldots, r_{\xi-1}, r_{\xi+1}, \ldots, r_n \xleftarrow{R} \mathbb{Z}_p$ and sets

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n] \setminus \{\xi\}.$$

It generates the triples $\{D_i, E_i, F_i\}_{i=1}^n$ by choosing $(D_i, E_i, F_i) \xleftarrow{R} \mathbb{G}^3$ at random for each $i \in [\xi - 1]$. The $\xi$-th triple $(D_k, E_k, F_k)$ is defined as

$$\begin{aligned} D_\xi &= T_t \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ E_\xi &= (g^{a_t})^{\beta'_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}. \end{aligned}$$

for a randomly chosen $\rho_\xi \xleftarrow{R} \mathbb{Z}_p$ and $\tau \xleftarrow{R} \{0,1\}^L$. As for $\{D_i, E_i, F_i\}_{i=\xi+1}^n$, they are obtained by choosing choosing $\rho_i, r_i \xleftarrow{R} \mathbb{Z}_p$ before setting

$$D_i = (g^b)^{\alpha_i \cdot r_i} \big(\prod_{k=1}^{L} X_{\xi,k,\tau[k]}\big)^{\rho_i}, \qquad E_i = (g^b)^{\beta_i \cdot r_i} \big(\prod_{k=1}^{L} Y_{\xi,k,\tau[k]}\big)^{\rho_i}, \qquad F_i = g^{\rho_i}.$$

Then, it uses the trapdoor $td_{\mathsf{CMH}}$ of the chameleon hash function to find coins $r_{hash} \in \mathcal{R}_{hash}$ such that $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash})$.

It is easy to see that, if $T_t = g^{a_t bc}$, the triple $(D_\xi, E_\xi, F_\xi)$ can be written

$$\begin{aligned} D_\xi &= h^{\alpha_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{X}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ E_\xi &= h^{\beta_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi} \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

meaning that $\mathcal{A}$'s view is the same as in $\mathsf{Game}_{\xi-1}$. In contrast, if $T_t \in_R \mathbb{G}$, it can be written $T_t = g^{a_t bc + z_t}$ for some uniformly random $z_t \in_R \mathbb{Z}_p$. In this case, $(D_\xi, E_\xi, F_\xi)$ can be written

$$\begin{aligned} D_\xi &= h^{z_t + \alpha_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{X}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ E_\xi &= h^{\beta_\xi \cdot r_\xi} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

for some random $z_t \in_R \mathbb{Z}_p$ that does not appear anywhere else. In this case, $\mathcal{A}$'s view corresponds to $\mathsf{Game}'_\xi$. □ □

**Lemma 3.9.** $\mathsf{Game}_\xi$ *is computationally indistinguishable from* $\mathsf{Game}'_\xi$ *under the DDH assumption in* $\mathbb{G}$*. The advantage of any PPT distinguisher between the two games can be bounded by* $\mathbf{Adv}^{\xi\text{-}\xi'}(\lambda) \leq \mathbf{Adv}^{\mathrm{DDH}_1}(\lambda)$.

*Proof.* We assume that there exists $\xi \in [n]$ such that $\mathcal{A}$ can tell apart $\mathsf{Game}'_\xi$ from $\mathsf{Game}_\xi$ with noticeable advantage. We build a distinguisher $\mathcal{B}$ that takes as input $Q$ tuples $\{(g, g^{a_i}, g^{a_i \cdot b}, g^b, T_i)\}_{i=1}^Q$ in $\mathbb{G}^5$ with the goal of deciding if $T_i = g^{a_i b}$ for each $i \in [Q]$ or if $\{T_i\}_{i=1}^Q$ are independent and uniformly distributed over $\mathbb{G}$. This assumption is known (see, e.g., [NR97, Lemma 4.4]) to have a tight reduction from the DDH assumption.

To this end, $\mathcal{B}$ defines $h = g^\eta$, $\hat{h} = \hat{g}^\eta$ for a random $\eta \xleftarrow{R} \mathbb{Z}_p$. It also computes $\hat{g}^{\theta_\xi}$ for a randomly chosen $\theta_\xi \xleftarrow{R} \mathbb{Z}_p$. Then, it picks $v_\xi \xleftarrow{R} \mathbb{Z}_p$ uniformly and sets

$$\hat{V}_\xi = \hat{g}^{v_\xi}.$$

Implicitly, $\mathcal{B}$ will define

$$\beta_\xi = b, \qquad \alpha_\xi = v_\xi - b \cdot \theta_\xi$$

although it does not know $(\alpha_\xi, \beta_\xi)$. It chooses $\hat{u} \in \hat{\mathbb{G}}$ and $u \in \mathbb{G}$ by setting $u = g^\nu$ and $\hat{u} = \hat{g}^\nu$ for a random $\nu \xleftarrow{R} \mathbb{Z}_p$. Then, $\mathcal{B}$ defines

$$\hat{H}_\xi = (\hat{h}^\xi \cdot \hat{u})^{v_\xi}.$$

For all indexes $j \in [n] \setminus \{\xi\}$, it chooses $\alpha_j, \beta_j, \theta_j \xleftarrow{R} \mathbb{Z}_p$ and faithfully computes $\hat{V}_j = \hat{g}^{\alpha_j + \theta_j \cdot \beta_j}$ and

$$\hat{H}_j = (\hat{h}^j \cdot \hat{u})^{\alpha_j + \theta_j \cdot \beta_j}.$$

Then, it constructs the MAC secret keys $\{\vec{x}_{j,\mu}, \vec{y}_{j,\mu}\}_{j=1}^n$ by for randomly chosen vectors $\vec{x}_{j,\mu} = (x_{j,1,\mu}, \ldots, x_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$, $\vec{y}_{j,\mu} = (y_{j,1,\mu}, \ldots, y_{j,L,\mu}) \xleftarrow{R} \mathbb{Z}_p^L$. For each $j \in [n]$, it defines

$$\begin{aligned} \hat{\vec{Y}}_{j,\mu} &= (\hat{Y}_{j,1,\mu}, \ldots, \hat{Y}_{j,L,\mu}) = \hat{g}^{\vec{y}_{j,\mu}}, & \vec{Y}_{j,\mu} &= (Y_{j,1,\mu}, \ldots, Y_{j,L,\mu}) = g^{\vec{y}_{j,\mu}} \\ \hat{\vec{X}}_{j,\mu} &= (\hat{X}_{j,1,\mu}, \ldots, \hat{X}_{j,L,\mu}) = \hat{g}^{\vec{x}_{j,\mu}}, & \vec{X}_{j,\mu} &= (X_{j,1,\mu}, \ldots, X_{j,L,\mu}) = g^{\vec{x}_{j,\mu}}. \end{aligned}$$

Then, it computes

$$\hat{\vec{Z}}_{j,\mu} = \hat{\vec{X}}_{j,\mu} \cdot \hat{\vec{Y}}_{j,\mu}^{\theta_j} \qquad \forall j \in [n].$$

For each $t \in [Q]$, the $t$-th invocation of the $\mathsf{LAF.LTag}(tk, .)$ oracle is handled by setting

$$R_\xi = g^{a_t}, \qquad S_\xi = (g^{a_t})^{\eta \cdot \xi + \nu} = (h^\xi \cdot u)^{a_t},$$

where $g^{a_t}$ is fetched from the $t$-th input tuple $(g, g^{a_t}, g^{a_t \cdot b}, g^b, T_t)$. For all indexes $i \neq \xi$, it chooses $r_1, \ldots, r_{\xi-1}, r_{\xi+1}, \ldots, r_n \xleftarrow{R} \mathbb{Z}_p$ and sets

$$R_i = g^{r_i}, \qquad S_i = (h^i \cdot u)^{r_i} \qquad \forall i \in [n] \setminus \{\xi\}.$$

It generates the triples $\{D_i, E_i, F_i\}_{i=1}^n$ by choosing $(D_i, E_i, F_i) \xleftarrow{R} \mathbb{G}^3$ at random for each $i \in [\xi - 1]$. The $\xi$-th triple $(D_k, E_k, F_k)$ is defined by sampling $D_\xi \xleftarrow{R} \mathbb{G}$ uniformly and setting

$$\begin{aligned} E_\xi &= T_t^\eta \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi}, \\ F_\xi &= g^{\rho_\xi}. \end{aligned}$$

for randomly chosen $\rho_\xi \xleftarrow{R} \mathbb{Z}_p$ and $\tau \xleftarrow{R} \{0,1\}^L$. As for $\{D_i, E_i, F_i\}_{i=\xi+1}^n$, they are obtained by choosing choosing $\rho_i, r_i \xleftarrow{R} \mathbb{Z}_p$ before setting

$$D_i = h^{\alpha_i \cdot r_i}\big(\prod_{k=1}^{L} X_{\xi,k,\tau[k]}\big)^{\rho_i}, \qquad E_i = h^{\beta_i \cdot r_i}\big(\prod_{k=1}^{L} Y_{\xi,k,\tau[k]}\big)^{\rho_i}, \qquad F_i = g^{\rho_i}.$$

Then, it uses the trapdoor $td_{\mathsf{CMH}}$ of the chameleon hash function to obtain coins $r_{hash} \in \mathcal{R}_{hash}$ such that $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_{\mathsf{a}}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash})$.

We observe that, if $T_t = g^{a_t \cdot b}$ for each $t \in [Q]$, the triples $(D_\xi, E_\xi, F_\xi)$ are distributed as

$$\begin{aligned} D_\xi &\in_R \mathbb{G}, \\ E_\xi &= h^{\beta_\xi \cdot \log_g(R_\xi)} \cdot \big(\prod_{k=1}^{L} \hat{Y}_{\xi,k,\tau[k]}\big)^{\rho_\xi} \\ F_\xi &= g^{\rho_\xi}, \end{aligned}$$

so that $\mathcal{A}$'s view is the same as in $\mathsf{Game}'_\xi$. In contrast, if $T_t \in_R \mathbb{G}$, it can be written $T_t = g^{a_t b + z_t}$ for some uniformly random $z_t \in_R \mathbb{Z}_p$ that does not appear anywhere else. In this case, $(D_\xi, E_\xi, F_\xi)$ is just a triple of uniformly random group elements, meaning that $\mathcal{A}$'s view is the same as in $\mathsf{Game}_\xi$. □ □

**Theorem 3.10.** *The above LAF provides evasiveness under the SXDH and wD3DH1 assumptions, assuming that* CMH *is a collision-resistant chameleon hash function. Namely, for any PPT evasiveness adversary, there exist efficient algorithms $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ with comparable running time and such that*

$$\begin{aligned} \mathbf{Adv}_Q^{\mathcal{A},\mathsf{eva}} \leq\ & \mathbf{Adv}_{\mathcal{B}_0}^{\mathsf{CMH\text{-}CR}}(\lambda) + n \cdot \mathbf{Adv}_{\mathcal{B}_1}^{\mathrm{wD3DH1}}(\lambda) \\ & + n \cdot \mathbf{Adv}_{\mathcal{B}_2}^{\mathrm{DDH}_2}(\lambda) + 2n \cdot (1 + L) \cdot \mathbf{Adv}_{\mathcal{B}_3}^{\mathrm{DDH}_1}(\lambda), \end{aligned}$$

*Proof.* Let us assume that a PPT adversary $\mathcal{A}$ can break the evasiveness property with noticeable advantage. We show that this contradicts either: (i) The indistinguishability of the scheme; (ii) The collision-resistance of the chameleon hash function; (iii) The SXDH assumption. We will prove this claim via a sequence of hybrid games:

In $\mathsf{Game}_0$, the challenger interacts with the adversary $\mathcal{A}$ as in the real evasiveness experiment. In the final game, we show that, if the adversary can output lossy tag with non-negligible probability, we can create an PPT algorithm breaks SXDH assumption with noticeable advantage.

For each $i$, we denote by $\mathsf{bad}_i$, the event $\mathcal{A}$ manages to output a non-trivial lossy tag in $\mathsf{Game}_i$.

**Game$_0$:** In this game, the adversary $\mathcal{A}$ has access to two oracles: (i) the lossy tag generation oracle $\mathsf{LAF.LTag}(tk, \cdot)$ that always outputs lossy tags; (ii) the lossy tag verification oracle $\mathsf{LAF.IsLossy}(\cdot)$ that uses a trapdoor to decide if a tag is lossy or injective. By definition.

$$\Pr[\mathsf{bad}_0] = \Pr[\mathcal{A}(1^\lambda, ek)^{\mathsf{LAF.LTag}(tk,\cdot),\mathsf{LAF.IsLossy}(\cdot)}]. \tag{3.30}$$

**Game$_1$:** In this game, we define $\mathsf{bad}_{hash}$ to be the event that the adversary $\mathcal{A}$ manages to output a tag $t = (t_a, t_c = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash}))$ for which the corresponding chameleon hash collides with that of some tags produced by the oracle $\mathsf{LAF.LTag}(tk, \cdot)$. The only difference between $\mathsf{Game}_0$ and $\mathsf{Game}_1$ is that the latter aborts when $\mathsf{bad}_{hash}$ occurs. It is straightforward that

$$|\Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_0]| = \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1]. \tag{3.31}$$

As in the proof of Theorem 3.3, we need to use the collision resistance property of the CMH to bound the probability $\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1]$. Since the $\mathsf{LAF.LTag}(tk, \cdot)$ oracle uses the trapdoor $td_{\mathsf{CMH}}$ to create lossy tags, we cannot immediately rely on the collision-resistance of CMH. Instead, we need to consider $\mathsf{Game}'_1$, where the $\mathsf{LAF.LTag}(tk, \cdot)$ oracle always outputs injective tags instead of lossy ones. Using the result of Theorem 3.7, we have

$$\begin{aligned}|\Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1] - \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}'_1]| \\ \leq n \cdot (\mathbf{Adv}^{\text{R-wD3DH1}}_{\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_2}(\lambda)).\end{aligned}$$

Since $\mathsf{Game}_0$ and $\mathsf{Game}_1$ only differ when $\mathsf{bad}_{hash}$ occurs in $\mathsf{Game}_1$, we can bound to probability (3.31) as

$$\begin{aligned}|\Pr[\mathsf{bad}_1] - \Pr[\mathsf{bad}_0]| &= \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}_1] \\ &\leq \Pr[\mathsf{bad}_{hash} \text{ in } \mathsf{Game}'_1] \\ &\quad + n \cdot (\mathbf{Adv}^{\text{R-wD3DH1}}_{\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_2}(\lambda)).\end{aligned} \tag{3.32}$$

Since the trapdoor $td_{\mathsf{CMH}}$ is never been used in $\mathsf{Game}'_1$, a straightforward reduction shows that $\Pr[\mathsf{bad}_{has} \text{ in } \mathsf{Game}_1\text{'}] \leq \mathbf{Adv}^{\mathsf{CR}}_{\mathsf{CMH}}(\lambda)$.

We now prove that, if event $\mathsf{bad}_1$ occurs with noticeable probability, we can construct a PPT adversary $\mathcal{B}$ that breaks the unforgeability of the MAC of Section 3.4.1. As this property is proven by Theorem 3.4 under SXDH assumption, this will conclude the proof.

Our MAC adversary $\mathcal{B}$ will simulate $\mathsf{Game}_1$ using the access to MAC oracle and MAC verification oracle. Algorithm $\mathcal{B}$ receives as input MAC public parameters

$$\mathsf{pp} = ((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, h, \hat{g}^\theta, (\hat{V}, \hat{\vec{Z}}_0, \hat{\vec{Z}}_1)).$$

As mentioned in $\mathsf{Game}_0$ of the proof for Lemma 3.4, $\mathcal{B}$ additionally obtain the discrete logarithm $\eta = \log_g(h)$ from the MAC challenger, which will allow it to simulate the

LAF.IsLossy oracle in the evasiveness experiment. Then, $\mathcal{B}$ extends these public parameters into public key of a LAF public key. To this end, $\mathcal{B}$ randomly guesses the position $i^\star \xleftarrow{R} [n]$ of the MAC forgery in the non-injective tag it is expected to produce and sets

$$\mathsf{pp}_{i^\star} = ((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, h, \hat{g}^\theta, (\hat{V}, \hat{\vec{Z}}_0, \hat{\vec{Z}}_1))$$

using the public parameters pp obtained from its challenger. Next, for all $i \in [n]\backslash\{i^\star\}$, $\mathcal{B}$ sets $\hat{h} = \hat{g}^\eta$ and generates $n-1$ keys $(\{\mathsf{pp}_i, \mathsf{sk}_{mac,i})\}_{i\in[n]\backslash\{i^\star\}}$ for the MAC of Section 3.4.1 which all share the same $g, h \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. For each $i \in [n] \setminus \{i^\star\}$, the $i$-th set of MAC public parameters thus consist of

$$\mathsf{pp}_i = ((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, h, \hat{g}^{\theta_i}, (\hat{V}_i, \hat{\vec{Z}}_{i,0}, \hat{\vec{Z}}_{i,1})).$$

To complete the generation the LAF evaluation key, $\mathcal{B}$ chooses $a \xleftarrow{R} \mathbb{Z}_p$, which is used to set $u = g^a$ and $\hat{u} = \hat{g}^a$. In addition, $\mathcal{B}$ computes $\hat{H}_i = (\hat{V}_i^{i\cdot\eta} \cdot \hat{V}_i^a)$ for all $i \in [n]$. It also chooses a key pair $(hk_{\mathsf{CMH}}, td_{\mathsf{CMH}})$ for the chameleon hash function CMH and includes $hk_{\mathsf{CMH}}$ in the LAF evaluation key.

To simulate the generation of lossy tags at each LAF.LTag-query $t_\mathsf{a}$ made by $\mathcal{A}$, $\mathcal{B}$ forwards $t_\mathsf{a}$ to its challenger and obtains a MAC $(R_{i^\star}, D_{i^\star}, E_{i^\star}, F_{i^\star})$ of the message $t_\mathsf{a}$. Then, $\mathcal{B}$ computes on its own $n-1$ MACs $\{(R_i, D_i, E_i, F_i)\}_{i\in[n]\backslash\{i^\star\}}$ using the secret MAC keys $\{\mathsf{sk}_{mac,i}\}_{i\in[n]\backslash\{i^\star\}}$. For all $i \in [n]$, $\mathcal{B}$ also computes $S_i = R_i^{\eta\cdot i+a}$. It finally uses the trapdoor of the chameleon hash function to generate $r_{hash}$ such that

$$\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_\mathsf{a}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash}) \in \{0,1\}^L$$

and returns $t = (t_\mathsf{a}, t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash}))$ to $\mathcal{A}$.

In order to simulate the LAF.IsInjective oracle for an input tag $t = (t_\mathsf{a}, t_\mathsf{c})$ containing $t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash})$, $\mathcal{B}$ computes the chameleon hash

$$\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_\mathsf{a}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash}).$$

It then uses $\eta$ to run the MAC verification oracle and check that $(R_{i^\star}, D_{i^\star}, E_{i^\star}, F_{i^\star})$ is a valid MAC. Since $\mathcal{B}$ also knows $\mathsf{sk}_{mac,i}$ for all $i \in [n]\backslash\{i^\star\}$, it can efficiently check that $\{(R_i, D_i, E_i, F_i)\}_{i\in[n]\backslash\{i^\star\}}$ are all valid MACs of $t_\mathsf{a}$. If all the MACs are valid, $\mathcal{B}$ also checks that for all $i \in [n]$, $S_i = R_i^{\eta\cdot i+a}$. If there exists $i \in [n]$ such that $S_i \neq R_i^{\eta\cdot i+a}$, $\mathcal{B}$ returns $\perp$ to indicate that $t$ does not belong to the space $\mathcal{T}$ of valid tags. Finally, $\mathcal{B}$ outputs 0 (meaning that the tag $t$ is non-injective) if it contains at least one valid MAC. If the $n$ MACs contained in $t_\mathsf{c}$ are all invalid and $S_i = R_i^{\eta\cdot i+a}$ for all $i \in [n]$, it outputs 1 meaning that the tag is injective.

It remains to show how $\mathcal{B}$ can extract a MAC forgery when $\mathsf{bad}_1$ occurs in $\mathsf{Game}_1$. Namely, we assume that $\mathcal{B}$ outputs a non-injective tag

$$t = (t_\mathsf{a}, t_\mathsf{c} = (\{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n, r_{hash}))$$

for which $\tau = \mathsf{CMhash}(hk_{\mathsf{CMH}}, (t_\mathsf{a}, \{R_i, S_i, D_i, E_i, F_i\}_{i=1}^n), r_{hash}) \in \{0,1\}^L$ has never been queried to LAF.LTag. Since $\mathcal{B}$ only queries the MAC oracle in the simulation of

$\mathsf{LAF.LTag}$, $\tau$ has never been queried to MAC oracle. Since $t$ is non-injective, the tuples $\{(R_i, S_i, D_i, E_i, F_i)\}_{i=1}^{n}$ must contain at least one valid MAC. If $(R_{i^\star}, D_{i^\star}, E_{i^\star}, F_{i^\star})$ is a valid MAC (which occurs with probability $1/n$ since $i^\star$ was chosen uniformly and independently of the adversary's view), $\mathcal{B}$ can successfully break the unforgeability of MAC by outputting $(R_{i^\star}, D_{i^\star}, E_{i^\star}, F_{i^\star})$. We thus have

$$\Pr[\mathsf{bad}_1] \leq n \cdot \mathbf{Adv}^{\text{uf-mac}}_{\mathcal{A}}(\lambda) \leq n \cdot (\mathbf{Adv}^{\text{DDH}_2}_{\mathcal{B}_1}(\lambda) + 2L \cdot \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_2}(\lambda)).$$

Putting the above arguments altogether, we obtain

$$\begin{aligned} \mathbf{Adv}^{\mathcal{A},\text{eva}}_{Q} &\leq \mathbf{Adv}^{\mathsf{CMH\text{-}CR}}_{\mathcal{B}_0}(\lambda) + n \cdot (\mathbf{Adv}^{\text{R-wD3DH1}}_{\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_3}(\lambda) \\ &\quad + \mathbf{Adv}^{\text{DDH}_2}_{\mathcal{B}_2}(\lambda) + 2L \cdot \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_3}(\lambda)) \\ &= \mathbf{Adv}^{\mathsf{CMH\text{-}CR}}_{\mathcal{B}_0}(\lambda) + n \cdot \mathbf{Adv}^{\text{R-wD3DH1}}_{\mathcal{B}_1}(\lambda) \\ &\quad + n \cdot \mathbf{Adv}^{\text{DDH}_2}_{\mathcal{B}_2}(\lambda) + n \cdot (1 + 2L) \cdot \mathbf{Adv}^{\text{DDH}_1}_{\mathcal{B}_3}(\lambda). \end{aligned} \tag{3.33}$$

By applying Lemma 2.1, we obtained the stated upper bound. □ □

# Part II.

# Homomorphic encryptions and zero-knowledge arguments

Non-interactive zero-knowledge (NIZK) proof systems have been introduced by Blum, Feldman and Micali [BFM88] and allow a prover to prove membership of an NP language without interactions. They can be used to convince anybody that a statement belongs to the language and their zero-knowledge property ensures that a proof reveals nothing beyond the membership of the language. NIZK proof systems are essential building blocks for many more complex cryptographic protocols, such as ring-signatures [RST01], group signatures [CH91], voting schemes [DJ01].

- The first chapter gives the first ring signature construction under DDH assumption achieving both tightly security and logarithmic-size property using a zero-knowledge proof as building blocks, corresponds to the following paper published at ESORICS2018 by Benoît Libert, Thomas Peters, Chen Qian [LPQ18]: Logarithmic-Size Ring Signatures With Tight Security from the DDH Assumption.
- The second chapter provides a new construction of lattice-based designated-verifier zero-knowledge argument systems. As application, we also constructed a lattice-based voting scheme without random oracle. This work is in submission by Pierre-Alain Fouque, Chen Qian, and Adeline Roux-Langlois.

# Table of Contents

# Logarithmic-Size Ring Signatures With Tight Security from the DDH Assumption

4

## 4.1. Introduction

As introduced by Rivest, Shamir and Tauman [RST01], ring signatures make it possible for a signer to sign messages while hiding his identity within an *a*d hoc set of users, called a *r*ing, that includes himself. To this end, the signer only needs to know the public keys of all ring members (besides his own secret key) in order to generate an anonymous signature on behalf of the entire ring. Unlike group signatures [CH91], ring signatures do not require any setup, coordination or registration phase and neither do they involve a tracing authority to de-anonymize signatures. Whoever has a public key can be appointed as a ring member without being asked for his agreement or even being aware of it. Moreover, signatures should ideally provide everlasting anonymity and carry no information as to which ring member created them. The main motivation of ring signatures is to enable the anonymous leakage of secrets, by concealing the identity of a source (e.g., a whistleblower in a political scandal) while simultaneously providing guarantees of its reliability.

In this paper, we consider the exact security of ring signatures in the random oracle model [BR93]. So far, the only known solutions with logarithmic signature length [GK15; LLNW16] suffered from loose reductions: the underlying hard problem could only be solved with a probability smaller than the adversary's advantage by a linear factor in the number of hash queries. Our main result is to give the first construction that simultaneously provides tight security – meaning that there is essentially no gap between the adversary's probability of success and the reduction's advantage in solving a hard problem – *a*nd logarithmic signature size in the number of ring members. In particular, the advantage of our reduction is not multiplicatively affected by the number $Q_H$ of random oracle queries nor the number of $Q_V$ of public verification keys in a ring.

Our Contribution. We describe the first logarithmic-size ring signatures with tight security proofs in the random oracle model. The unforgeability of our construction is proved under the standard Decision Diffie-Hellman (DDH) assumption in groups without a bilinear map while anonymity is achieved against unbounded adversaries. Our security proof eliminates *b*oth the linear gap in the number of random oracle queries *a*nd the $\Theta(Q_V)$ security loss. It thus features a *f*ully tight reduction, meaning that – up to statistically negligible terms – the reduction's advantage as a DDH distinguisher is only smaller than the adversary's forging probability by a factor 2. To our knowledge, our scheme is the first ring signature for which such a fully tight reduction is reported. It is obtained by tweaking a construction due to Groth and Kohlweiss [GK15] and achieves tight security

at the expense of increasing the number of scalars and group elements per signature by a small constant factor. For the same exact security, our reduction allows smaller key sizes which essentially decrease the signature length of [GK15] by a logarithmic factor $n$ in the cardinality $N$ of the ring and the time complexity by a factor $\omega(n^2)$. For rings of cardinality $N = 2^6$, for example, our signatures can be 36 times faster to compute and 6 times shorter than [GK15].

Our Techniques. Our scheme builds on the Groth-Kohlweiss proof system [GK15] that allows proving that one-out-of-$N$ commitments opens to 0 with a communication complexity $O(\log N)$. This proof system was shown to imply logarithmic-size ring signatures with perfect anonymity assuming that the underlying commitment scheme is perfectly hiding. At the heart of the protocol of [GK15] is a clever use of a $\Sigma$-protocol showing that a committed value $\ell$ is 0 or 1, which proceeds in the following way. In order to prove that a commitment $\vec{C}_\ell \in \{\vec{C}_i\}_{i=0}^{N-1}$ opens to 0 without revealing the index $\ell \in \{0, \ldots, N-1\}$, the $n$-bit indexes $\ell_j$ of the binary representation $\ell_1 \ldots \ell_n \in \{0,1\}^n$ of $\ell \in \{0, \ldots, N-1\}$ are committed to and, for each of them, the prover uses the aforementioned $\Sigma$-protocol to prove that $\ell_j \in \{0,1\}$. The response $f_j = a_j + \ell_j x$ of the $\Sigma$-protocol is then viewed as a degree-one polynomial in the challenge $x \in \mathbb{Z}_q$ and used to define polynomials

$$P_i[Z] = \prod_{j=1}^{n} f_{j,i_j} = \delta_{i,\ell} \cdot Z^n + \sum_{k=0}^{n-1} p_{i,k} \cdot Z^k \qquad \forall i \in [N],$$

where $f_{j,0} = f_j$ and $f_{f,1} = x - f_j$, which have degree $n = \log N$ if $i = \ell$ and degree $n-1$ otherwise. In order to prove that one of the polynomials $\{P_i[Z]\}_{i=0}^{N-1}$ has degree $n$ without revealing which one, Groth and Kohlweiss [GK15] homomorphically compute the commitment $\prod_{i=0}^{N-1} \vec{C}_i^{P_i(x)}$ and multiply it with $\prod_{k=0}^{n-1} \vec{C}_{d_k}^{-x^k}$, for auxiliary homomorphic commitments $\{\vec{C}_{d_k} = \prod_{i=0}^{N-1} \vec{C}_i^{p_{i,k}}\}_{k=0}^{n-1}$, in order to cancel out the terms of degree 0 to $n-1$ in the exponent. Then, they prove that the product $\prod_{i=0}^{N-1} \vec{C}_i^{P_i(x)} \cdot \prod_{k=0}^{n-1} \vec{C}_{d_k}^{-x^k}$ is indeed a commitment of 0. The soundness of the proof relies on the Schwartz-Zippel lemma, which ensures that $\prod_{i=0}^{N-1} \vec{C}_i^{P_i(x)} \cdot \prod_{k=0}^{n-1} \vec{C}_{d_k}^{-x^k}$ is unlikely to be a commitment to 0 if $\vec{C}_\ell$ is not.

As an application of their proof system, Groth and Kohlweiss [GK15] obtained logarithmic-size ring signatures from the discrete logarithm assumption in the random oracle model. While efficient and based on a standard assumption, their scheme suffers from a loose security reduction incurred by the use of the Forking Lemma [PS96]. In order to extract a discrete logarithm from a ring signature forger, the adversary has to be run $n = \log N$ times with the same random tape (where $N$ is the ring cardinality), leading to a reduction with advantage $\varepsilon' \approx \frac{\varepsilon^n}{Q_V \cdot Q_\mathcal{H}}$, where $Q_\mathcal{H}$ is the number of hash queries and $Q_V$ is the number of public keys. This means that, if we want to increase the key size so as to compensate for the concrete security gap, we need to multiply the security parameter by a factor $n = \log N$, even without taking into account the factors $Q_\mathcal{H}$ and $Q_V$.

In our pursuit of a tight reduction, a first idea is to apply the lossy identification paradigm [KW03; AFLT12] where the security proofs proceed by replacing a well-formed public key by a so-called lossy public key, with respect to which forging a signature becomes statistically impossible. In particular, the DDH-based instantiation of Katz and Wang [KW03]

appears as an ideal candidate since, somewhat analogously to [GK15], well-formed public keys can be seen as homomorphic Elgamal encryptions of $0$. However, several difficulties arise when we try to adapt the techniques of [KW03; AFLT12] to the ring signature setting.

The first one is that the Groth-Kohlweiss ring signatures [GK15] rely on perfectly *h*iding commitments in order to achieve unconditional anonymity whereas the Elgamal encryption scheme is a perfectly binding commitment. This fortunately leaves us the hope for computational anonymity if we trade the perfectly hiding commitments for Elgamal encryptions. A second difficulty is to determine which public keys should be replaced by lossy keys in the reduction. At each public key generation query, the reduction has to decide if the newly generated key will be lossy or injective. Replacing all public keys by lossy keys is not possible because of corruptions (indeed, lossy public keys have no underlying secret key) and the reduction does not know in advance which public keys will end up in the target ring $\mathcal{R}^\star$ of the forgery. Only replacing a randomly chosen key by a lossy key does not work either: indeed, in the ring signature setting, having one lossy public key $PK^\dagger$ in the target ring $\mathcal{R}^\star$ does not prevent an unbounded adversary from using the secret key of a well-formed key $PK^\star \in \mathcal{R}^\star \setminus \{PK^\dagger\}$ to create a forgery. Moreover, as long as the reduction can only embed the challenge (injective or lossy) key in one output of the key generation oracle, it remains stuck with an advantage $\Theta(\varepsilon/Q_V)$ if the forger has advantage $\varepsilon$. Arguably, this bound is the best we can hope for by directly applying the lossy identification technique.

To obtain a fully tight reduction, we depart from the lossy identification paradigm [AFLT12] in that, instead of tampering with one user's public keys at some step, our security proof embeds a DDH instance in the public parameters pp of the scheme. This allows the reduction to have all users' private keys at disposal and reveal them to the adversary upon request. In the real system, the set pp contains uniformly random group elements $(g, h, \tilde{g}, \tilde{h}, U, V) \in \mathbb{G}^6$ and each user's public key consists of a pair $(X, Y) = (g^\alpha \cdot h^\beta, \tilde{g}^\alpha \cdot \tilde{h}^\beta)$, where $(\alpha, \beta) \in \mathbb{Z}_q^2$ is the secret key. The idea of the security proof is that, if $(g, h, \tilde{g}, \tilde{h}) \in \mathbb{G}^4$ is not a Diffie-Hellman tuple, the public key $PK = (X, Y)$ uniquely determines $(\alpha, \beta) \in \mathbb{Z}_q^2$. In the case $\tilde{h} = \tilde{g}^{\log_g(h)}$, the public key $(X, Y)$ is compatible with $q$ equally likely pairs $(\alpha, \beta)$ since it only reveals the information $\log_g(X) = \alpha + \log_g(h) \cdot \beta$.

The reduction thus builds a DDH distinguisher by forcing the adversary's forgery to contain a committed encoding $\Gamma = U^\alpha \cdot V^\beta$ of the signer's secret key $(\alpha, \beta) \in \mathbb{Z}_q^2$, which can be extracted using some trapdoor information. So long as $(U, V)$ is linearly independent of $(g, h)$, the encoding $\Gamma = U^\alpha \cdot V^\beta$ is independent of the adversary's view if $(g, h, \tilde{g}, \tilde{h})$ is a Diffie-Hellman tuple. In contrast, this encoding is uniquely determined by the public key if $\tilde{h} \neq \tilde{g}^{\log_g(h)}$. This allows the reduction to infer that $(g, h, \tilde{g}, \tilde{h})$ is a Diffie-Hellman tuple whenever it extracts $\Gamma = U^\alpha \cdot V^\beta$ from the adversary's forgery. To apply this argument, however, we need to make sure that signing queries do not leak any more information about $(\alpha, \beta)$ than the public key $PK = (X, Y)$ does. For this purpose, we resort to *l*ossy encryption schemes [BHY09] (a.k.a. dual-mode encryption/commitments [GS08; PVW08]), which can either behave as perfectly hiding or perfectly binding commitments depending on the distribution of the public key. In each signature, we embed a lossy encryption $(T_0, T_1) = (g^{\theta_1} \cdot h^{\theta_2}, U^\alpha \cdot V^\beta \cdot H_1^{\theta_1} \cdot H_2^{\theta_2})$ of $\Gamma = U^\alpha \cdot V^\beta$, which is computed using the

DDH-based lossy encryption scheme of [BHY09]. If $(H_1, H_2) \in \mathbb{G}^2$ is linearly independent of $(g, h)$, then $(T_0, T_1)$ perfectly hides $\Gamma$. At the same time, the reduction should be able to extract $\Gamma$ from $(T_0, T_1)$ in the forgery. To combine these seemingly conflicting requirements, we derive $(H_1, H_2)$ from a (pseudo-)random oracle which is programmed to have $(H_1, H_2) = (g^\gamma, h^\gamma)$, for some $\gamma \in_R \mathbb{Z}_q$, in the adversary's forgery and maintain the uniformity of all pairs $(H_1, H_2) \in \mathbb{G}^2$ in all signing queries. By doing so, the witness indistinguishability of the Groth-Kohlweiss $\Sigma$-protocol [GK15] implies that the adversary only obtains a limited amount of information from uncorrupted users' private keys. While the above information theoretic argument is reminiscent of the security proof of Okamoto's identification scheme [Oka92], our proof departs from [Oka92] in that we do not rewind the adversary as it would not enable a tight reduction.

Related Work. The concept of ring signatures was coined by Rivest, Shamir and Tauman [RST01] who gave constructions based on trapdoor functions and proved their security in the ideal cipher model. They also mentioned different realizations based on proofs of partial knowledge [CDS94]. The latter approach was extended by Abe *et* al. [AOS02] to support rings containing keys from different underlying signatures and assumptions. Bresson, Stern and Szydlo [BSS02] modified the scheme of Rivest *et* al. [RST01] so as to prove it secure in the random oracle model.

In 2006, Bender, Katz and Morselli [BKM06] provided rigorous security definitions and theoretical constructions without random oracles. In the standard model, the first efficient instantiations were put forth by Shacham and Waters [SW07] in groups with a bilinear map. Brakerski and Tauman-Kalai [BK10] gave alternative constructions based on lattice assumptions. Meanwhile, Boyen [Boy07] suggested a generalization of the primitive with standard-model instantiations.

The early realizations [RST01; BSS02] had linear size in the cardinality of the ring. Dodis *et* al. [DKNS04] mentioned constant-size ring signatures as an application of their anonymous *a*d hoc identification protocols. However, their approach requires a setup phase where an RSA modulus is generated by some trusted entity. Chase and Lysyanskaya [CL06] suggested a similar construction of constant-size ring signatures from cryptographic accumulators [BM93]. However, efficiently instantiating their construction requires setup-free accumulators which are compatible with zero-knowledge proofs. The hash-based accumulators of [BLL00; CHKO08] would not provide efficient solutions as they would incur proofs of knowledge of hash function pre-images. While the lattice-based construction of [LLNW16] relies on hash-based accumulators, its security proof is not tight and its efficiency is not competitive with discrete-logarithm-based techniques. Sander's number-theoretic accumulator [San99] is an alternative candidate to instantiate [CL06] without a setup phase. However, it is not known to provide practical protocols: as observed in [GK15], it would involve much larger composite integers than standard RSA moduli (besides zero-knowledge proofs for double discrete logarithms). Moreover, it is not clear how it would be compatible with tight security proofs.

Chandran, Groth and Sahai [CGS07] gave sub-linear-size signatures in the standard model, which were recently improved in [Gon17]. In the random oracle model, Groth and Kohlweiss [GK15] described an elegant construction of logarithmic-size ring signatures

based on the discrete logarithm assumption. Libert *et* al. [LLNW16] obtained logarithmic-size lattice-based ring signatures in the random oracle model.

The logarithmic-size ring signatures of [GK15; BCC+15; LLNW16] are obtained by applying the Fiat-Shamir heuristic [FS86] to interactive $\Sigma$-protocols. While these solutions admit security proofs under well-established assumptions in the random oracle model, their security reductions are pretty loose. In terms of exact security, they are doomed [PV05] to lose a linear factor in the number $Q_{\mathcal{H}}$ of random oracle queries as long as they rely on the Forking Lemma [PS96].

The exact security of digital signatures was first considered by Bellare and Rogaway [BR96] and drew a lot of attention [Cor00; GJ03; KW03; AFLT12; KK12] since then.

## 4.2. Background

### 4.2.1. Syntax and Security Definitions for Ring Signatures

**Definition 4.1.** *A ring signature scheme consists of a tuple of efficient algorithms*

$$(\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$$

*with the following specifications:*

***Par-Gen***$(1^\lambda)$*: Given a security parameter $\lambda$, outputs the public parameters* pp*.*

***Keygen***$(\mathsf{pp})$*: Given* pp*, outputs a key pair $(PK, SK)$ for the user.*

***Sign***$(\mathsf{pp}, SK, \mathcal{R}, M)$*: Given the user's secret key $SK$, a ring $\mathcal{R}$ and a message $M$, outputs the signature $\sigma$ of the message $M$ on behalf of the ring $\mathcal{R}$.*

***Verify***$(\mathsf{pp}, M, \mathcal{R}, \sigma)$*: Given the message $M$, a ring $\mathcal{R}$ and a candidate signature $\sigma$, the verification algorithm outputs $0$ or $1$.*

These algorithms must also verify the correctness, meaning that for all $\mathsf{pp} \leftarrow \mathsf{Par\text{-}Gen}(1^\lambda)$, $(PK, SK) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, for all $M$, and for all $\mathcal{R}$ such that $PK \in \mathcal{R}$, we have w.h.p

$$\mathsf{Verify}(\mathsf{pp}, M, \mathcal{R}, \mathsf{Sign}(\mathsf{pp}, SK, \mathcal{R}, M)) = 1.$$

From a security point of view, Bender *et* al. [BKM06] suggested the following stringent definitions of anonymity and unforgeability.

**Definition 4.2.** *A ring signature* $(\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ *provides statistical **a**nonymity under **f**ull key exposure if, for any computationally unbounded adversary $\mathcal{A}$, there exists a negligible function $\varepsilon(\lambda)$ such that*

$$\Big|\Pr[\mathsf{pp} \leftarrow \mathsf{Par\text{-}Gen}(1^\lambda); (M^\star, i_0, i_1, \mathcal{R}^\star) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\cdot)}; b \xleftarrow{R} \{0,1\}; \sigma^\star \leftarrow \mathsf{Sign}(pp, SK_{i_b}, \mathcal{R}^\star, M^\star) : \mathcal{A}(\sigma^\star) = b] - \frac{1}{2}\Big| < \varepsilon(\lambda),$$

*where $PK_{i_0}, PK_{i_1} \in \mathcal{R}^\star$ and* Keygen *is an oracle that generates a fresh key pair $(PK, SK) \leftarrow \mathsf{Keygen}(\mathsf{pp})$ at each query and returns both $PK$ and $SK$ to $\mathcal{A}$.*

**Definition 4.3.** *A ring signature* $(\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ *provides* ***u****nforgeability w.r.t* ***i****nsider corruption if, for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\varepsilon(\lambda)$ *such that, for any* $\mathsf{pp} \leftarrow \mathsf{Par\text{-}Gen}(1^\lambda)$*, we have*

$$\Pr[(M, \mathcal{R}, \sigma) \leftarrow \mathcal{A}^{\mathsf{Keygen}(\cdot),\mathsf{Sign}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp}) : \mathsf{Verify}(\mathsf{pp}, M, \mathcal{R}, \sigma) = 1] < \varepsilon(\lambda),$$

- $\mathsf{Keygen}()$*: is an oracle that maintains a counter* $j$ *initialized to* $0$*. At each query, it increments* $j$*, generates* $(PK_j, SK_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and outputs* $PK_j$*.*
- $\mathsf{Sign}(i, M, \mathcal{R})$ *is an oracle that returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, SK_i, \mathcal{R}, M)$ *if* $PK_i \in \mathcal{R}$ *and* $(PK_i, SK_i)$ *has been generated by* $\mathsf{Keygen}$*. Otherwise, it returns* $\perp$*.*
- $\mathsf{Corrupt}(i)$ *returns* $SK_i$ *if* $(PK_i, SK_i)$ *was output by* $\mathsf{Keygen}$ *and* $\perp$ *otherwise.*
- $\mathcal{A}$ *is restricted to output a triple* $(M, \mathcal{R}, \sigma)$ *such that: (i) No query of the form* $(\star, M, \mathcal{R})$ *has been made to* $\mathsf{Sign}(\cdot, \cdot, \cdot)$*; (ii)* $\mathcal{R}$ *only contains public keys* $PK_i$ *produced by* $\mathsf{Keygen}$ *and for which* $i$ *was never queried to* $\mathsf{Corrupt}(\cdot)$*.*

### 4.2.2. $\Sigma$-protocol Showing that a Commitment Opens to $0$ or $1$

We recall the $\Sigma$-protocol used in [GK15] to prove that a commitment opens to $0$ or $1$. Let $\mathcal{R} = \{(ck, c, (m, r)) \mid c = \mathsf{Com}_{ck}(m, r) \wedge (m, r) \in \{0, 1\} \times \mathbb{Z}_q\}$ the binary relation, where $ck$ is the commitment key generated for the underlying commitment scheme, $u = c$ is the public input and $w = (m, r)$ is the private input. Figure 4.1 gives us a $\Sigma$-protocol $(\mathsf{P}, \mathrm{V})$ for $\mathcal{R}$.

**Theorem 4.1** ([GK15, Theorem 2])**.** *Let* $(\mathsf{Setup}, \mathsf{Com})$ *be a perfectly binding, computationally hiding, strongly binding and additively homomorphic commitment scheme. The* $\Sigma$*-protocol presented in figure 4.1 for the commitment to* $0$ *or to* $1$ *is perfectly complete, perfectly* $2$*-special sound and perfectly SHVZK.*

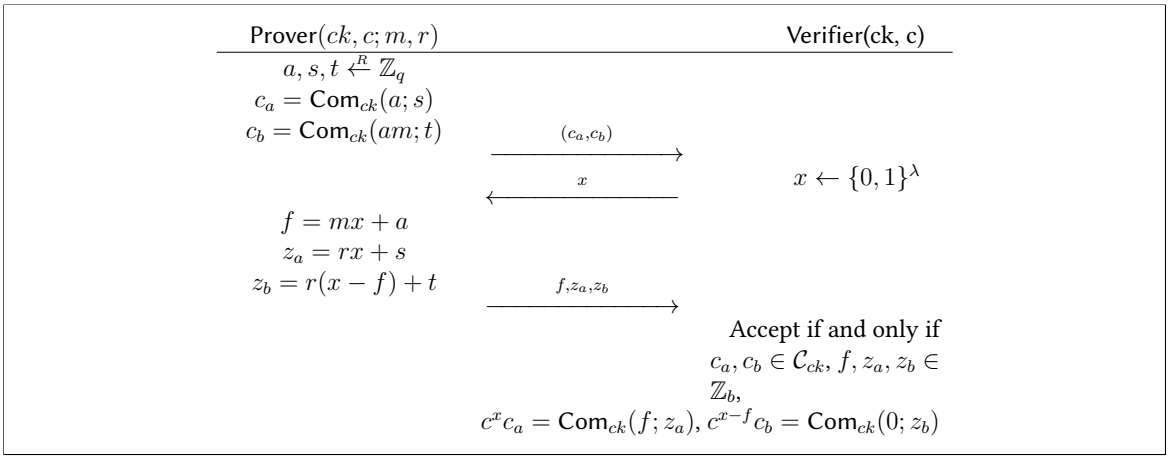


**Figure 4.1.** – $\Sigma$-protocol for commitment to $m \in \{0, 1\}$

### 4.2.3. $\Sigma$-protocol for One-out-of-$N$ Commitments Containing $0$

Groth and Kohlweiss [GK15] used the $\Sigma$-protocol of Section 4.2.2 to build an efficient $\Sigma$-protocol allowing to prove knowledge of an opening of one-out-of-$N$ commitments $\{c_i\}_{i=0}^{N-1}$ to $m = 0$. Their protocol outperforms the standard OR-proof approach [CDS94] in that its communication complexity is only $O(\log N)$, instead of $O(N)$. The idea is to see the responses $f = mx + a$ of the basic $\Sigma$ protocol as degree-1 polynomials in $x \in \mathbb{Z}_q$ and exploit the homomorphism of the commitment.

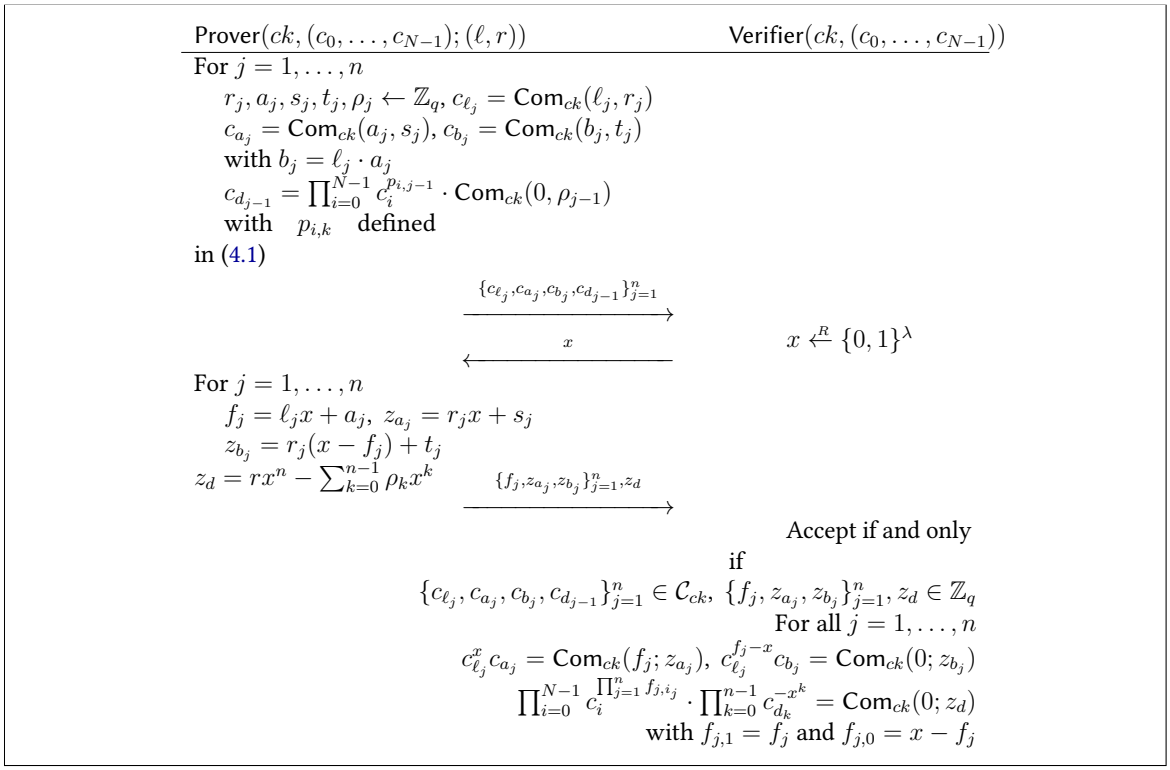


**Figure 4.2.** – $\Sigma$-protocol for one of $(c_0, \ldots, c_{N-1})$ commits to $0$

**Theorem 4.2** ([GK15, Theorem 3]). *The $\Sigma$-protocol of figure 4.2 is perfectly complete. It is (perfectly) $(n+1)$-special sound if the commitment is (perfectly) binding. It is (perfectly) SHVZK if the commitment scheme is (perfectly) hiding.*

In Figure 4.2, for each $i$, $p_{i,0}, \ldots, p_{i,n-1} \in \mathbb{Z}_q$ are the coefficients of the polynomial

$$P_i[Z] = \prod_{j=1}^{n} F_{j,i_j}[Z] = \delta_{i,\ell} \cdot Z^n + \sum_{k=0}^{n-1} p_{i,k} \cdot Z^k \qquad \forall i \in \{0, \ldots, N-1\} \tag{4.1}$$

obtained by defining $F_{j,1}[Z] = \ell_j \cdot Z + a_j$ and $F_{j,0}[Z] = Z - F_{j,1}[Z]$ for all $j \in [n]$. Note that the equality (4.1) stems from the fact that, for each index $i = i_1 \ldots i_n \in \{0, \ldots, N-1\}$, we have $F_{j,i_j}[Z] = \delta_{i_j,\ell_j} \cdot Z + (-1)^{\delta_{0,i_j}} \cdot a_j$ for all $j \in [n]$, so that the coefficient of $Z^n$ in (4.1) is non-zero if and only if $i = \ell$.

### 4.2.4. A Note on the Application to Ring Signatures

In [GK15], Groth and Kohlweiss obtained a ring signature scheme by applying the Fiat-Shamir paradigm [FS86] to the above $\Sigma$-protocol. In short, key pairs are of the form $(c, r)$ such that $c = \mathsf{Com}(0; r)$ and a ring signature associated with $\mathcal{R} = \{c_0, \ldots, c_N\}$ is simply a proof that the signer knows how to open to $0$ one of the $N$ commitments in that ring. In [GK15], the following theorem states about the security of the resulting construction, denoted $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy})$.

**Theorem 4.3** ([GK15, Theorem 4])**.** *The scheme* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy})$ *is a ring signature scheme with perfect correctness. It has perfect anonymity if the commitment scheme is perfectly hiding. It is unforgeable in the random oracle model if the commitment scheme is perfectly hiding and computationally binding.*

As the security of the ring signature relies on that of the $\Sigma$-protocol, it is interesting to take a closer look at the computation of commitments $\{C_{d_{j-1}}\}_{j=1}^{n}$ in Figure 4.2. This part of the $\Sigma$-protocol is the only point where the ring signature generation may involve adversarially-generated values. In the anonymity game, the signer's public key may be one of the only two honestly-generated public keys in the ring $\mathcal{R}$. The security proof of [GK15] argues that, as long as the commitment is perfectly hiding, the fact that each $C_{d_{j-1}}$ contains a (randomizing) factor $\mathsf{Com}(0; \rho_{j-1})$, for some uniformly random $\rho_{j-1}$, is sufficient to guarantee perfect anonymity. We point out an issue that arises when $\mathcal{R} = \{c_0, \ldots, c_N\}$ contains maliciously generated keys outside the space of honestly generated commitments (even if they are perfectly hiding). In short, multiplying a maliciously generated commitment by a fresh commitment may not fully "clean-up" its distribution.

The following example is a perfectly hiding commitment where re-randomizing does not wipe out maliciously generated commitments components: the setup algorithm outputs generators $ck = (g, h)$ cyclic group $\mathbb{G}$ of prime order $q$; committing to $m \in \mathbb{Z}_q$ using randomness $\rho = (r, s) \xleftarrow{R} \mathbb{Z}_q^2$ is achieved by computing $\mathsf{Com}_{ck}(m; \rho) = (c_1, c_2, c_3) = (g^m h^r, g^s, h^s) \in \mathbb{G}^3$, which is a perfectly hiding commitment since $c_1$ is a Pedersen commitment and the Elgamal encryption $(c_2, c_3)$ of $0$ is independent of $c_1$. If we consider the maliciously generated commitment $(c_1^\star, c_2^\star, c_3^\star) = (h^u, g^v, g \cdot h^v)$, multiplying it by any $\mathsf{Com}_{ck}(0; \rho)$ does not bring it back in the range of $\mathsf{Com}$. Therefore, in an instantiation with the above commitment, an unbounded adversary can defeat the anonymity property.

The only missing requirement on behalf of the underlying perfectly hiding commitment is that it should be possible to efficiently recognize elements in the range of the commitment algorithm. This assumption is quite natural and satisfied by schemes like Pedersen's commitment. Hence, this observation does not affect the perfect anonymity of the discrete-log-based instantiation of [GK15].

## 4.3. A Fully Tight Construction from the DDH Assumption

We modify the scheme of [GK15] so as to prove its unforgeability via a fully tight reduction from the DDH assumption. The advantage of the DDH distinguisher is only smaller than

the adversary's advantage by a (small) constant factor.

The price to pay for this fully tight reduction is relatively small since signatures are only longer than in [GK15] by roughly $2n$ group elements. Moreover, as in [GK15], our signing algorithm requires $\Theta(N)$ exponentiations if $N$ is the size of the ring.

### 4.3.1. Description

We exploit the fact that, in the $\Sigma$-protocol of [GK15], not all first-round messages should be computed using the same commitment scheme as the one used to compute the public key. The second step of the signing algorithm computes perfectly hiding commitments $\{\vec{C}_{d_k}\}_{k=0}^{n-1}$ which are vectors of dimension 4. They live in a different space than public keys $(X, Y) = (g^{\alpha} \cdot h^{\beta}, \tilde{g}^{\alpha} \cdot \tilde{h}^{\beta})$, which are DDH-based lossy encryptions of (and thus perfectly hiding commitments to) 0.

The signer generates a commitment $(T_0, T_1) = (g^{\theta_1} \cdot h^{\theta_2}, \Gamma \cdot H_1^{\theta_1} \cdot H_2^{\theta_2})$ to $\Gamma = U^{\alpha_\ell} \cdot V^{\beta_\ell}$, which encodes his secret key $(\alpha_\ell, \beta_\ell) \in \mathbb{Z}_q^2$. This defines a vector $\vec{V}_\ell = (X_\ell, Y_\ell, T_0, T_1) \in \mathbb{G}^4$ in the column space of a matrix $\mathbf{M}_H \in \mathbb{G}^{4\times 4}$, which has full rank in the scheme but not in the proof of unforgeability. Then, for each key $\vec{X}_i = (X_i, Y_i)$ in the ring $\mathcal{R}$, the signer defines $\vec{V}_i = (X_i, Y_i, T_0, T_1)^{\top} \in \mathbb{G}^4$ and, by extending the technique of [GK15], generates a NIZK proof that one of the vectors $\{\vec{V}_i\}_{i=0}^{N-1}$ is in the column span of $\mathbf{M}_H$. To prove this without revealing which $\vec{V}_\ell \in \mathbb{G}^4$ is used, the commitments $\{\vec{C}_{d_{j-1}}\}_{j=1}^{n}$ are re-randomized by multiplying them with a random vector in the column space of $\mathbf{M}_H$.

**Par-Gen**$(1^\lambda)$**:** Given a security parameter $\lambda$, choose a cyclic group $\mathbb{G}$ of prime order $q$ with generators $g, h, \tilde{g}, \tilde{h} \xleftarrow{R} \mathbb{G}$ and $U, V \xleftarrow{R} \mathbb{G}$. Choose hash functions $\mathcal{H}_{\mathsf{FS}} : \{0,1\}^* \to \mathbb{Z}_q$ and $\mathcal{H} : \{0,1\}^* \to \mathbb{G}^2$ which will be modeled as random oracles. Output the common public parameters $\mathsf{pp} = \big(\lambda, \mathbb{G}, g, h, \tilde{g}, \tilde{h}, U, V\big)$.

**Keygen**$(\mathsf{pp})$**:** Given $\mathsf{pp}$, choose a secret key is $SK = (\alpha, \beta) \xleftarrow{R} \mathbb{Z}_q^2$ and compute the public key $PK = \vec{X} = (X, Y) = (g^{\alpha} \cdot h^{\beta}, \tilde{g}^{\alpha} \cdot \tilde{h}^{\beta})$.

**Sign**$(\mathsf{pp}, SK, \mathcal{R}, M)$**:** To sign $M \in \{0,1\}^*$ on behalf of $\mathcal{R} = \{\vec{X}_0, \ldots, \vec{X}_{N-1}\}$ such that $\vec{X}_i = (X_i, Y_i) \in \mathbb{G}^2$ for each $i \in [N]$, the signer uses $SK = (\alpha, \beta)$ and $PK = \vec{X} = (X, Y) = (g^{\alpha} \cdot h^{\beta}, \tilde{g}^{\alpha} \cdot \tilde{h}^{\beta}) \in \mathcal{R}$ as follows. We assume that $N = 2^n$ for some $n$. Let $\ell \in \{0, \ldots, N-1\}$ the index of $PK = \vec{X}$ in $\mathcal{R}$ when $\mathcal{R}$ is arranged in lexicographical order and write it as $\ell = \ell_1 \ldots \ell_n \in \{0,1\}^n$.

1. Choose $\theta_1, \theta_2 \xleftarrow{R} \mathbb{Z}_q$. For all $j \in [n]$, choose $a_j, r_j, s_j, t_j, u_j, v_j, w_j, \rho_{j-1} \xleftarrow{R} \mathbb{Z}_q$ and compute $(T_0, T_1) = \big(g^{\theta_1} \cdot h^{\theta_2}, U^{\alpha} \cdot V^{\beta} \cdot H_1^{\theta_1} \cdot H_2^{\theta_2}\big)$, as well as

$$\begin{aligned} \vec{C}_{\ell_j} &= (C_{\ell_j,0}, C_{\ell_j,1}) = \big(g^{r_j} \cdot h^{s_j},\ g^{\ell_j} \cdot H_1^{r_j} \cdot H_2^{s_j}\big) \\ \vec{C}_{a_j} &= (C_{a_j,0}, C_{a_j,1}) = \big(g^{t_j} \cdot h^{u_j},\ g^{a_j} \cdot H_1^{t_j} \cdot H_2^{u_j}\big) \\ \vec{C}_{b_j} &= (C_{b_j,0}, C_{b_j,1}) = \big(g^{v_j} \cdot h^{w_j},\ g^{\ell_j \cdot a_j} \cdot H_1^{v_j} \cdot H_2^{w_j}\big), \end{aligned} \tag{4.2}$$

where $(H_1, H_2) = \mathcal{H}\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big) \in \mathbb{G}^2$. Define

$$\mathbf{M}_H \;=\; \begin{bmatrix} g & h & 1 & 1 \\ \tilde{g} & \tilde{h} & 1 & 1 \\ 1 & 1 & g & h \\ U & V & H_1 & H_2 \end{bmatrix} \in \mathbb{G}^{4\times 4} \tag{4.3}$$

and its corresponding discrete logarithms $\mathbf{L}_h = \log_g(\mathbf{M}_H)$ matrix

$$\mathbf{L}_h \;=\; \begin{bmatrix} 1 & \log_g(h) & 0 & 0 \\ \log_g(\tilde{g}) & \log_g(\tilde{h}) & 0 & 0 \\ 0 & 0 & 1 & \log_g(h) \\ \log_g(U) & \log_g(V) & \log_g(H_1) & \log_g(H_2) \end{bmatrix} \in \mathbb{Z}_q^{4\times 4}. \tag{4.4}$$

Note that the signer's witnesses $(\alpha, \beta, \theta_1, \theta_2) \in \mathbb{Z}_q^4$ satisfy

$$\log_g \big[X \mid Y \mid T_0 \mid T_1\big]^\top \;=\; \mathbf{L}_h \cdot \big[\alpha \mid \beta \mid \theta_1 \mid \theta_2\big]^\top. \tag{4.5}$$

In the following, we will sometimes re-write relation (4.5) as

$$\begin{bmatrix} X \\ Y \\ T_0 \\ T_1 \end{bmatrix} \;=\; \begin{bmatrix} g & h & 1 & 1 \\ \tilde{g} & \tilde{h} & 1 & 1 \\ 1 & 1 & g & h \\ U & V & H_1 & H_2 \end{bmatrix} \odot \begin{bmatrix} \alpha \\ \beta \\ \theta_1 \\ \theta_2 \end{bmatrix}. \tag{4.6}$$

For each $i \in [N]$, define the vector $\vec{V}_i = (X_i, Y_i, T_0, T_1)^\top \in \mathbb{G}^4$. The next step is to prove knowledge of witnesses $(\alpha_\ell, \beta_\ell, \theta_1, \theta_2) \in \mathbb{Z}_q^4$ such that $\vec{V}_\ell = (X_\ell, Y_\ell, T_0, T_1)^\top = g^{\mathbf{L}_h \cdot (\alpha_\ell, \beta_\ell, \theta_1, \theta_2)^\top}$, for some $\ell \in [N]$.

2. For each $j \in [n]$, pick $\rho_{j-1,\alpha}, \rho_{j-1,\beta}, \rho_{j-1,\theta_1}, \rho_{j-1,\theta_2} \xleftarrow{R} \mathbb{Z}_q$ and compute

$$\vec{C}_{d_{j-1}} = \prod_{i=0}^{N-1} \vec{V}_i^{p_{i,j-1}} \cdot g^{\mathbf{L}_h \cdot (\rho_{j-1,\alpha}, \rho_{j-1,\beta}, \rho_{j-1,\theta_1}, \rho_{j-1,\theta_2})^\top} \;\in \mathbb{G}^4, \tag{4.7}$$

where, for each $i \in \{0, \ldots, N-1\}$, $p_{i,0}, \ldots, p_{i,n-1}$ are the coefficients of

$$P_i[Z] = \prod_{j=1}^{n} F_{j,i_j}[Z] = \delta_{i,\ell} \cdot Z^n + \sum_{k=0}^{n-1} p_{i,k} \cdot Z^k \;\in \mathbb{Z}_q[Z], \tag{4.8}$$

where $F_{j,1}[Z] = \ell_j \cdot Z + a_j$ and $F_{j,0}[Z] = Z - F_{j,1}[Z]$ for all $j \in [n]$. Note that the coefficient of $Z^n$ in (4.8) is non-zero if and only if $i = \ell$.

3. Compute $x = \mathcal{H}_{\mathsf{FS}}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^n) \;\in \mathbb{Z}_q$.

4. For each $j \in [n]$, compute (modulo $q$) $f_j = \ell_j \cdot x + a_j = F_{j,1}(x)$ and

$$\begin{aligned} z_{r_j} &= r_j \cdot x + t_j, & \bar{z}_{r_j} &= r_j \cdot (x - f_j) + v_j \\ z_{s_j} &= s_j \cdot x + u_j, & \bar{z}_{s_j} &= s_j \cdot (x - f_j) + w_j \end{aligned}$$

and

$$\begin{aligned} z_{d,\alpha} &= \alpha \cdot x^n - \sum_{k=0}^{n-1} \rho_{k,\alpha} \cdot x^k, & z_{d,\beta} &= \beta \cdot x^n - \sum_{k=0}^{n-1} \rho_{k,\beta} \cdot x^k \\ z_{d,\theta_1} &= \theta_1 \cdot x^n - \sum_{k=0}^{n-1} \rho_{k,\theta_1} \cdot x^k, & z_{d,\theta_2} &= \theta_2 \cdot x^n - \sum_{k=0}^{n-1} \rho_{k,\theta_2} \cdot x^k \end{aligned}$$

Let $\boldsymbol{\Sigma}_j = \big(\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}, f_j, z_{r_j}, z_{s_j}, \bar{z}_{r_j}, \bar{z}_{s_j}\big)$ for all $j \in [n]$ and output

$$\sigma = \big(\{\boldsymbol{\Sigma}_j\}_{j=1}^n, T_0, T_1, z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2}\big). \tag{4.9}$$

**Verify**$(\mathsf{pp}, M, \mathcal{R}, \sigma)$**:** Given a ring $\mathcal{R} = \{\vec{X}_0, \ldots, \vec{X}_{N-1}\}$ and a pair $(M, \sigma)$, parse $\sigma$ as in (4.9) and define $f_{j,1} = f_j$ and $f_{j,0} = x - f_j$ for each $j \in [n]$.

1. Compute $(H_1, H_2) = \mathcal{H}\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big) \in \mathbb{G}^2$ and, for each public key $\vec{X}_i = (X_i, Y_i) \in \mathbb{G}^2$ in $\mathcal{R}$, set $\vec{V}_i = (X_i, Y_i, T_0, T_1)^\top \in \mathbb{G}^4$.
2. Let $x = \mathcal{H}_{\mathsf{FS}}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^n)$. If the equalities

$$\begin{aligned} \vec{C}_{a_j} \cdot \vec{C}_{\ell_j}^{x} &= \big(g^{z_{r_j}} \cdot h^{z_{s_j}},\ g^{f_j} \cdot H_1^{z_{r_j}} \cdot H_2^{z_{s_j}}\big), \\ \vec{C}_{b_j} \cdot \vec{C}_{\ell_j}^{x-f_j} &= \big(g^{\bar{z}_{r_j}} \cdot h^{\bar{z}_{s_j}}, H_1^{\bar{z}_{r_j}} \cdot H_2^{\bar{z}_{s_j}}\big), \qquad \forall j \in [n] \end{aligned} \tag{4.10}$$

are not satisfied, return 0. Then, return 1 if and only if

$$\prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{j=1}^{n} \vec{C}_{d_{j-1}}^{-(x^{j-1})} = \begin{bmatrix} g & h & 1 & 1 \\ \tilde{g} & \tilde{h} & 1 & 1 \\ 1 & 1 & g & h \\ U & V & H_1 & H_2 \end{bmatrix} \odot \begin{bmatrix} z_{d,\alpha} \\ z_{d,\beta} \\ z_{d,\theta_1} \\ z_{d,\theta_2} \end{bmatrix}. \tag{4.11}$$

Correctness is shown by observing from (4.8) that $\prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^n f_{j,i_j}}$ equals

$$\begin{aligned} \prod_{i=0}^{N-1} \vec{V}_i^{P_i(x)} &= \prod_{i=0}^{N-1} \vec{V}_i^{\delta_{i,\ell} \cdot x^n + \sum_{k=0}^{n-1} p_{i,k} \cdot x^k} = \vec{V}_\ell^{x^n} \cdot \prod_{i=0}^{N-1} \vec{V}_i^{\sum_{k=0}^{n-1} p_{i,k} \cdot x^k} \\ &= \vec{V}_\ell^{x^n} \cdot \prod_{k=0}^{n-1} \cdot \Big(\prod_{i=0}^{N-1} \vec{V}_i^{p_{i,k}}\Big)^{x^k} = \vec{V}_\ell^{x^n} \cdot \prod_{k=0}^{n-1} \cdot \big(\vec{C}_{d_k} \cdot g^{-\mathbf{L}_h \cdot (\rho_{k,\alpha}, \rho_{k,\beta}, \rho_{k,\theta_1}, \rho_{k,\theta_2})^\top}\big)^{x^k}, \end{aligned}$$

where the last equality follows from (4.7). Since $\vec{V}_\ell = g^{\mathbf{L}_h \cdot (\alpha_\ell, \beta_\ell, \theta_1, \theta_2)^\top}$, we obtain

$$\begin{aligned}\prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^{n} f_{j,i_j}} \cdot \prod_{k=0}^{n-1} \vec{C}_{d_k}^{-x^k} &= \vec{V}_\ell^{x^n} \cdot \prod_{k=0}^{n-1} g^{-\mathbf{L}_h \cdot (\rho_{k,\alpha}, \rho_{k,\beta}, \rho_{k,\theta_1}, \rho_{k,\theta_2})^\top (x^k)}, \\ &= g^{\mathbf{L}_h \cdot (z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2})^\top}.\end{aligned}$$

### 4.3.2. Security Proofs

Statistical anonymity is achieved because $\{\vec{C}_{d_{j-1}}\}_{j=1}^n$ are uniformly distributed. The reason is that the matrices (4.4) have full rank in the scheme (but not in the proof of unforgeability), so that computing $\vec{C}_{d_{j-1}}$ as per (4.7) makes its distribution uniform over $\mathbb{G}^4$.

**Theorem 4.4.** *Any unbounded anonymity adversary $\mathcal{A}$ has advantage at most $\mathbf{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) \leq \frac{2}{q} + \frac{Q_{\mathcal{H}_{\mathsf{FS}}}}{q^2}$, where $Q_{\mathcal{H}_{\mathsf{FS}}}$ is the number of hash queries to $\mathcal{H}_{\mathsf{FS}}$.*

*Proof.* We consider a sequence of games and, for each $i$, we call $W_i$ the event that the challenger outputs 1 in Game $i$, meaning that the adversary successfully guesses the challenger's bit and outputs $b' = b$. In each game, we also consider the event $E_i$ by which the tuple $(g, h, \tilde{g}, \tilde{h})$ of the public parameter or the tuple $(g, h, H_1, H_2)$ defined in the challenge signature forms a Diffie-Hellman tuple.

**Game** 0**:** This is the real game where the challenger outputs 1 if and only if $\mathcal{A}$ wins. By definition, $\mathcal{A}$'s advantage is $\mathbf{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) = |\Pr[W_0] - 1/2|$. We assume that, for all public keys generated by the $\mathsf{Keygen}(.)$ oracle, the adversary immediately obtains the secret keys. Since $(\tilde{g}, \tilde{h})$ is uniformly distributed in $\mathsf{pp}$ and since $(H_1, H_2)$ is an independent random output of the random oracle $\mathcal{H}$, we find $\Pr[E_0] = 2/q - 1/q^2$ and then $\Pr[W_0] \leq \Pr[W_0 | \neg E_0] + (2/q - 1/q^2)$. We are left with bounding $\Pr[W_0 | \neg E_0]$.

**Game** 1**:** We modify the generation of the challenge signature. On a challenge query $(M, \mathcal{R}, \ell^{(0)}, \ell^{(1)})$, where $(0 \leq \ell^{(0)}, \ell^{(1)} \leq |\mathcal{R}| - 1)$, the challenger $\mathcal{B}$ parses $\mathcal{R}$ as $\{\vec{X}_0, \ldots, \vec{X}_{N-1}\}$ and returns $\perp$ if $\vec{X}_{\ell^{(0)}}$ and $\vec{X}_{\ell^{(1)}}$ are not public keys produced by the $\mathsf{Keygen}(.)$ oracle. Otherwise, it flips a coin $b \xleftarrow{R} \{0, 1\}$ and sets $(\ell_1, \ldots, \ell_n)$ as the bit representation of $\ell^{(b)}$. Then, it chooses $x \xleftarrow{R} \mathbb{Z}_q$ as well as $z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2} \xleftarrow{R} \mathbb{Z}_q$ and $f_j, z_{r_j}, z_{s_j}, \bar{z}_{r_j}, \bar{z}_{s_j} \xleftarrow{R} \mathbb{Z}_q$ for all $j \in [n]$. Then, it picks $T_0 \xleftarrow{R} \mathbb{G}$ as well $C_{\ell_j,0} \xleftarrow{R} \mathbb{Z}_q$ for all $j \in [n]$. It can now compute

$$C_{a_j,0} = g^{z_{r_j}} \cdot h^{z_{s_j}} \cdot C_{\ell_j,0}^{-x}, \qquad C_{b_j,0} = g^{\bar{z}_{r_j}} \cdot h^{\bar{z}_{s_j}} \cdot C_{\ell_j,0}^{f_j - x} \qquad \forall j \in [n],$$

so as to define $(H_1, H_2) = \mathcal{H}(M, \mathcal{R}, T_0, \{(C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0})\}_{j=1}^n)$. Then, $\mathcal{B}$ completes the computation of the dual-mode commitments as follows. First, it picks $T_1 \xleftarrow{R} \mathbb{G}$ as well as $C_{\ell_j,1} \xleftarrow{R} \mathbb{G}$ for all $j \in [n]$. Then, it computes

$$C_{a_j,1} = \left(g^{f_j} \cdot H_1^{z_{r_j}} \cdot H_2^{z_{s_j}}\right) \cdot C_{\ell_j,1}^{-x}, \qquad C_{b_j,1} = \left(H_1^{\bar{z}_{r_j}} \cdot H_2^{\bar{z}_{s_j}}\right) \cdot C_{\ell_j,1}^{f_j - x}.$$

It draws $\vec{C}_{d_{j-1}} \xleftarrow{R} \mathbb{G}$ for each $j \in \{2, \ldots, n\}$ while, for $j = 1$, it computes

$$\vec{C}_{d_0} = \prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^{n} f_{j,i_j}} \prod_{j=2}^{n} \tilde{\vec{C}}_{d_{j-1}}^{-(x^{j-1})} \Big( \mathbf{M}_{\mathcal{H}} \odot (-z_{d,\alpha}, -z_{d,\beta}, -z_{d,\theta_1}, -z_{d,\theta_2})^{\top} \Big),$$

where $\vec{V}_i = (X_i, Y_i, T_0, T_1)^{\top}$, $f_{j,1} = f_j$ and $f_{j,0} = x - f_j$ for each $j \in [n]$. Finally, the challenger programs the random oracle $\mathcal{H}_{\mathsf{FS}}$ to have the equality

$$x = \mathcal{H}_{\mathsf{FS}}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^{n}).$$

If $\mathcal{H}_{\mathsf{FS}}$ was already defined for this input, the challenger aborts and picks $b'$ as a random bit. If the simulation does not fail, the oracle outputs the challenge signature $\sigma = \big(\{\mathbf{\Sigma}_j\}_{j=1}^{n}, T_0, T_1, z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2}\big)$, which is distributed exactly as in $W_0|\neg E_0$, assuming that $E_1$ does not occur. Indeed, if $(g, h, H_1, H_2)$ is not a Diffie-Hellman tuple in both games, all the dual-mode commitments are perfectly hiding and if $(g, h, \tilde{g}, \tilde{h})$ is not a Diffie-Hellman tuple as well, the matrix $\mathbf{M}_H$ has full rank, meaning that $\{\vec{C}_{d_{j-1}}\}_{j=1}^{n}$ are uniformly distributed over $\mathbb{G}^4$. Therefore, as long as no collision occurs in the simulation of the challenge, $\mathcal{A}$'s view in $W_1|\neg E_1$ is the same as in $W_0|\neg E_0$. If we call $F_1$ the event that a hash collision prevents the correct generation of the challenge signature, we obtain the inequality $|\Pr[W_1|\neg(E_1 \cup F_1)] - \Pr[W_0|\neg E_0]| \leq \Pr[F_1] \leq Q_{\mathcal{H}_{\mathsf{FS}}}/q^2$.

In Game 1, when neither $E_1$ nor $F_1$ occurs, the signature is perfectly independent of $b \in_R \{0, 1\}$, so that $\Pr[W_1|\neg(E_1 \cup F_1)] = 1/2$. All the above observations together thus implies $\mathbf{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) \leq 2/q + (Q_{\mathcal{H}_{\mathsf{FS}}} - 1)/q^2$. □ □

**Theorem 4.5.** *The scheme is **u**nforgeable under the DDH assumption in the random oracle model. For any adversary $\mathcal{A}$ with running time $t$ and making $Q_V$ queries to the key generation oracle, $Q_S$ signing queries as well as $Q_{\mathcal{H}}$ and $Q_{\mathcal{H}_{\mathsf{FS}}}$ queries to the random oracles $\mathcal{H}$ and $\mathcal{H}_{\mathsf{FS}}$, respectively, there is a DDH distinguisher $\mathcal{B}$ with running time $t' \leq t + \mathsf{poly}(\lambda, Q_S, Q_V, Q_{\mathcal{H}})$ and such that*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{euf-cma}}(\lambda) \leq 2 \cdot \mathbf{Adv}_{\mathcal{B}}^{\text{DDH}}(\lambda) + \frac{Q_S + Q_{\mathcal{H}_{\mathsf{FS}}} \cdot (1 + \log Q_V) + 5}{q} + \frac{Q_S \cdot (Q_{\mathcal{H}_{\mathsf{FS}}} + 2Q_{\mathcal{H}} + 2Q_S)}{q^2}. \tag{4.12}$$

*Proof.* We use a sequence of games where, for each $i$, $W_i$ stands for the event that the challenger outputs 1 in Game $i$.

**Game** 0**:** This is the real game. At each query $i \in [Q_V]$ to the key generation oracle $\mathsf{Keygen}(\cdot)$, the challenger $\mathcal{B}$ honestly chooses $\alpha_i, \beta_i \xleftarrow{R} \mathbb{Z}_q$ and returns the public key $PK_i = \vec{X}_i = (X_i, Y_i) = (g^{\alpha_i} \cdot h^{\beta_i}, \tilde{g}^{\alpha_i} \cdot \tilde{h}^{\beta_i})$ and retains $SK_i = (\alpha_i, \beta_i)$ for later use. If $\mathcal{A}$ subsequently submits $\vec{X}_i = (X_i, Y_i)$ to the corruption oracle, $\mathcal{B}$ reveals $SK_i = (\alpha_i, \beta_i)$. Moreover, all signing queries are answered by faithfully running the

signing algorithm. At the end of the game, $\mathcal{A}$ outputs a forgery $(M^\star, \sigma^\star, \mathcal{R}^\star)$, where $\mathcal{R}^\star = \{\vec{X}_0^\star, \ldots, \vec{X}_{N^\star-1}^\star\}$,

$$\sigma^\star = \big(\{\mathbf{\Sigma}_j^\star\}_{j=1}^n, T_0^\star, T_1^\star, z_{d,\alpha}^\star, z_{d,\beta}^\star, z_{d,\theta_1}^\star, z_{d,\theta_2}^\star\big), \tag{4.13}$$

with $\mathbf{\Sigma}_j^\star = \big(\vec{C}_{\ell_j}^\star, \vec{C}_{a_j}^\star, \vec{C}_{b_j}^\star, \vec{C}_{d_{j-1}}^\star, f_j^\star, z_{r_j}^\star, z_{s_j}^\star, \bar{z}_{r_j}^\star, \bar{z}_{s_j}^\star\big)$. At this point, $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ wins, meaning that: (i) $\sigma^\star$ correctly verifies; (ii) $\mathcal{R}^\star$ only contains uncorrupted public keys; (iii) No signing query involved a tuple of the form $(\cdot, M^\star, \mathcal{R}^\star)$ . By definition, we have $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{euf-cma}}(\lambda) = \Pr[W_0]$.

**Game** 1: This game is like Game 0 but we modify the signing oracle. Note that each signing query triggers a query to the random oracle $\mathcal{H}(.)$ since the challenger $\mathcal{B}$ has to faithfully compute $T_0$ and $\{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n$ before obtaining

$$\mathcal{H}\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big).$$

In Game 1, at each signing query, $\mathcal{B}$ aborts in the event that $\mathcal{H}(\cdot)$ was already defined for the input $\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big)$. Since such an input contains uniformly random elements, the probability to abort during the entire game is at most $Q_S \cdot (Q_S + Q_{\mathcal{H}})/q^2$ and we have $|\Pr[W_1] - \Pr[W_0]| \leq Q_S \cdot (Q_S + Q_{\mathcal{H}})/q^2$.

**Game** 2: We modify the random oracle $\mathcal{H}$ when it is directly invoked by $\mathcal{A}$ (i.e., $\mathcal{H}$-queries triggered by signing queries are treated as in Game 0). At each $\mathcal{H}$-query $\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big)$, the challenger $\mathcal{B}$ returns the previously defined value if it exists. Otherwise, it picks $\gamma \xleftarrow{R} \mathbb{Z}_q$ and defines the hash value as $(H_1, H_2) = \mathcal{H}\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big) = (g^\gamma, h^\gamma)$. Note that $\mathcal{H}(\cdot)$ is no longer a truly random oracle since $(g, h, H_1, H_2)$ is a Diffie-Hellman tuple. Still, under the DDH assumption, this modification has no noticeable effect on $\mathcal{A}$'s winning probability. Lemma 4.6 describes a DDH distinguisher such that $|\Pr[W_2] - \Pr[W_1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathrm{DDH}}(\lambda) + 1/q$.

Since $(g, h, H_1, H_2)$ is a Diffie-Hellman tuple in Game 2, $\gamma \in \mathbb{Z}_q$ can be used as a decryption key for the DDH-based dual-mode encryption scheme. Another consequence of the last transition is that the matrix $\mathbf{L}_h$ of (4.3) has no longer full rank since its last row is linearly dependent with the first three rows.

**Game** 3: We introduce a failure event $F_3$ which causes the challenger $\mathcal{B}$ to output 0. When $\mathcal{A}$ outputs its forgery $\sigma^\star$, $\mathcal{B}$ parses $\sigma^\star$ as in (4.13) and computes

$$(H_1^\star, H_2^\star) = \mathcal{H}\big(M^\star, \mathcal{R}^\star, T_0^\star, \{C_{\ell_j,0}^\star, C_{a_j,0}^\star, C_{b_j,0}^\star\}_{j=1}^n\big).$$

Event $F_3$ is defined to be the event that either: (1) The hash value $(H_1^\star, H_2^\star)$ was not defined at any time; (2) It was defined but collides with a pair $(H_1, H_2) = \mathcal{H}\big(M, \mathcal{R}, T_0, \{C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0}\}_{j=1}^n\big)$ defined in response to a signing query $(\ell, M, \mathcal{R})$ for some index $\ell \in \{0, \ldots, |\mathcal{R}| - 1\}$, when $\mathcal{R}$ is arranged in lexicographic order. Note that

the probability of case (1) cannot exceed $1/q$ because $\mathcal{H}(\cdot)$ is unpredictable as a random oracle. Moreover, since a winning adversary must forge a signature on some $(M^\star, \mathcal{R}^\star)$ that has never been queried for signature, the probability of case (2) is bounded by $Q_S/q^2$ multiplied by $Q_\mathcal{H}$ since we must consider the probability that a tuple $(g, h, H_1, H_2)$ defined in a signing query is accidentally a Diffie-Hellman tuple and collides with the response of a hash query. We find $|\Pr[W_3]-\Pr[W_2]| \leq \Pr[F_3] \leq 1/q + Q_S \cdot Q_\mathcal{H}/q^2$.

**Game** 4: This game is identical to Game 3 with one modification. When the adversary $\mathcal{A}$ outputs its forgery $\sigma^\star$, $\mathcal{B}$ parses $\sigma^\star$ as in (4.13) and computes

$$(H_1^\star, H_2^\star) = \mathcal{H}\big(M^\star, \mathcal{R}^\star, T_0^\star, \{C_{\ell_j,0}^\star, C_{a_j,0}^\star, C_{b_j,0}^\star\}_{j=1}^n\big).$$

Then, $\mathcal{B}$ recalls the previously defined exponent $\gamma^\star \in \mathbb{Z}_q$ such that $(H_1^\star, H_2^\star) = (g^{\gamma^\star}, h^{\gamma^\star})$ and uses it to decrypt the dual-mode ciphertexts $\{\vec{C}_{\ell_j}^\star\}_{j=1}^n$. It aborts and outputs 0 if one of these ciphertexts turns out not to encrypt a bit $\ell_j^\star \in \{0,1\}$. Note that, if $\mathcal{B}$ does not abort, it decodes an $n$-bit string $\ell^\star = \ell_1^\star \ldots \ell_n^\star \in \{0,1\}^n$ from $\{\vec{C}_{\ell_j}^\star\}_{j=1}^n$. We claim that we have $|\Pr[W_4] - \Pr[W_3]| \leq (1 + Q_{\mathcal{H}_{\mathsf{FS}}})/q$.

The only situation where Game 4 deviates from Game 3 is the event $F_4$ that either: (i) $\mathcal{A}$ did not query $\mathcal{H}_{\mathsf{FS}}(\cdot)$ on the input that the forgery relates to; (ii) $\mathcal{A}$ manages to break the soundness of the proof system showing that each of the ciphertexts $\{\vec{C}_{\ell_j}^\star\}_{j=1}^n$ encrypts a bit. Lemma 4.7 shows that $\Pr[F_4] \leq (1 + Q_{\mathcal{H}_{\mathsf{FS}}})/q$.

**Game** 5: In this game, we modify the challenger's behavior when $\mathcal{A}$ outputs a forgery $\sigma^\star$. Having decoded the $n$-bit string $\ell^\star = \ell_1^\star \ldots \ell_n^\star \in \{0,1\}^n$ from the dual-mode ciphertexts $\{\vec{C}_{\ell_j}^\star\}_{j=1}^n$, $\mathcal{B}$ also runs the decryption algorithm for $(T_0^\star, T_1^\star)$ to compute $\Gamma^\star = T_1^\star / T_0^{\star\gamma^\star}$. At this point, $\mathcal{B}$ recalls the secret key $SK = (\alpha_{\ell^\star}, \beta_{\ell^\star})$ of the $\ell^\star$-th member of the ring $\mathcal{R}^\star = \{\vec{X}_0^\star, \ldots \vec{X}_{N^\star-1}^\star\}$ in lexicographical order. If $\Gamma^\star = U^{\alpha_{\ell^\star}} \cdot V^{\beta_{\ell^\star}}$, $\mathcal{B}$ outputs 1. Otherwise, it outputs 0. Lemma 4.8 shows that $|\Pr[W_5]-\Pr[W_4]| \leq Q_{\mathcal{H}_{\mathsf{FS}}} \cdot \log(Q_V)/q$.

**Game** 6: This game is identical to Game 5 except that we change the distribution of

$$\mathsf{pp} = \big(\lambda, \mathbb{G}, g, h, \tilde{g}, \tilde{h}, U, V\big).$$

Here, instead of choosing $g, h, \tilde{g}, \tilde{h} \xleftarrow{R} \mathbb{G}$ uniformly, we set $(g, h, \tilde{g}, \tilde{h}) = (g, h, g^\rho, h^\rho)$ for a randomly chosen $\rho \xleftarrow{R} \mathbb{Z}_q$. Clearly, the two distributions of $\mathsf{pp}$ are indistinguishable under the DDH assumption and $\mathcal{B}$ can immediately be turned into an efficient DDH distinguisher (the proof is straightforward) such that $|\Pr[W_6] - \Pr[W_5]| \leq \mathbf{Adv}_\mathcal{B}^{\mathrm{DDH}}(\lambda)$.

**Game** 7: This game is like Game 6 except that we now simulate the proof of knowledge of secret keys in all outputs of the signing oracle. On a signing query $(M, \mathcal{R}, \ell)$, where

$(0 \leq \ell \leq |\mathcal{R}|-1)$, the challenger parses $\mathcal{R}$ as $\{\vec{X}_0, \ldots, \vec{X}_{N-1}\}$ and returns $\perp$ if $\vec{X}_\ell$ is not public keys produced by the $\mathsf{Keygen}(.)$ oracle. Otherwise, the challenger chooses $x \xleftarrow{R} \mathbb{Z}_q$ as well as $z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2} \xleftarrow{R} \mathbb{Z}_q$ and $f_j, z_{r_j}, z_{s_j}, \bar{z}_{r_j}, \bar{z}_{s_j} \xleftarrow{R} \mathbb{Z}_q$, for all $j \in [n]$. Then, it picks $T_0 \xleftarrow{R} \mathbb{G}$ as well as $r_j, s_j \xleftarrow{R} \mathbb{Z}_q$ for all $j \in [n]$, and honestly computes $C_{\ell_j,0} = g^{r_j} \cdot h^{s_j}$ for all $j \in [n]$. It can now compute for all $j \in [n]$,

$$C_{a_j,0} = g^{z_{r_j}} \cdot h^{z_{s_j}} \cdot C_{\ell_j,0}^{-x}, \qquad C_{b_j,0} = g^{\bar{z}_{r_j}} \cdot h^{\bar{z}_{s_j}} \cdot C_{\ell_j,0}^{f_j - x},$$

and define $(H_1, H_2) = \mathcal{H}(M, \mathcal{R}, T_0, \{(C_{\ell_j,0}, C_{a_j,0}, C_{b_j,0})\}_{j=1}^n)$. Then, it completes the computation of dual-mode commitments as follows. First, it chooses $T_1 \xleftarrow{R} \mathbb{G}$ and computes $C_{\ell_j,1} = g^{\ell_j} \cdot H_1^{r_j} \cdot H_2^{s_j}$ for all $j \in [n]$. Then, it computes

$$C_{a_j,1} = \left(g^{f_j} \cdot H_1^{z_{r_j}} \cdot H_2^{z_{s_j}}\right) \cdot C_{\ell_j,1}^{-x}, \qquad C_{b_j,1} = \left(H_1^{\bar{z}_{r_j}} \cdot H_2^{\bar{z}_{s_j}}\right) \cdot C_{\ell_j,1}^{f_j - x},$$

for all $j \in [n]$. Then, for each $j \in \{2, \ldots, n\}$, the challenger faithfully computes $\vec{C}_{d_{j-1}}$ as per (4.7) but, for index $j = 1$, it computes

$$\vec{C}_{d_0} = \prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^n f_{j,i_j}} \prod_{j=2}^{n} \vec{C}_{d_{j-1}}^{-(x^{j-1})} \left(\mathbf{M}_{\mathcal{H}} \odot (-z_{d,\alpha}, -z_{d,\beta}, -z_{d,\theta_1}, -z_{d,\theta_2})^\top\right),$$

where $\vec{V}_i = (X_i, Y_i, T_0, T_1)^\top$, $f_{j,1} = f_j$ and $f_{j,0} = x - f_j$ for each $j \in [n]$. Finally, the challenger $\mathcal{B}$ programs the random oracle $\mathcal{H}_{\mathsf{FS}}$ to have the equality $x = \mathcal{H}_{\mathsf{FS}}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^n)$. If $\mathcal{H}_{\mathsf{FS}}$ was already defined for this input, $\mathcal{B}$ aborts and outputs 0. If the simulation does not fail, the oracle sets $\mathbf{\Sigma}_j = \left(\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}, f_j, z_{r_j}, z_{s_j}, \bar{z}_{r_j}, \bar{z}_{s_j}\right)$ for all $j \in [n]$ and outputs the signature $\sigma = \left(\{\mathbf{\Sigma}_j\}_{j=1}^n, T_0, T_1, z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_2}\right)$, which is distributed exactly as in Game 6 unless $(g, h, H_1, H_2)$ happens to form a Diffie-Hellman tuple. Indeed, although the adversary's signing queries may involve rings $\mathcal{R}$ that contain maliciously generated keys of the form $\vec{X}_i = (X_i, Y_i) = (X_i, \Omega_i \cdot X_i^{\log_g(\tilde{g})})$, with $\Omega_i \neq 1_{\mathbb{G}}$, this does not prevent the simulated commitments $\{\vec{C}_{d_{j-1}}\}_{j=1}^n$ from having the same distribution as in Game 6. In simulated signatures, we indeed have

$$\vec{C}_{d_{j-1}} = \prod_{i=0}^{N-1} \vec{V}_i^{p_{i,j-1}} \cdot g^{\mathbf{L}_h \cdot \vec{\rho}_j} \qquad \forall j \in \{2, \ldots, n-1\}$$

for random $\vec{\rho}_2, \ldots, \vec{\rho}_{n-1} \in_R \mathbb{Z}_q^4$, where $p_{i,0}, \ldots, p_{i,n-1}$ are the coefficients of $\prod_{j=1}^n f_{j,i_j} = \delta_{i,\ell} x^n + \sum_{j=1}^n p_{i,j-1} x^{j-1}$. Since $\vec{V}_\ell = g^{\mathbf{L}_h \cdot (\alpha_\ell, \beta_\ell, \theta_1, \theta_2)^\top}$ and defining

$$\vec{\rho}_1 = -(z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_1})^\top - \sum_{j=2}^{n} \vec{\rho}_j x^{j-1} + (\alpha_\ell, \beta_\ell, \theta_1, \theta_2) \cdot x^n,$$

we have

$$\begin{aligned} \vec{C}_{d_0} = \vec{V}_\ell^{x^n} \cdot \prod_{i=0}^{N-1} \vec{V}_i^{\sum_{j=1}^n p_{i,j-1} x^{j-1}} \cdot \prod_{i=0}^{N-1} \vec{V}_i^{-\sum_{j=2}^n p_{i,j-1} x^{j-1}} \\ \cdot g^{-\mathbf{L}_h \cdot (z_{d,\alpha}, z_{d,\beta}, z_{d,\theta_1}, z_{d,\theta_1})^\top - \mathbf{L}_h \cdot \sum_{j=2}^n \vec{\rho}_j x^{j-1}} = \prod_{i=0}^{N-1} \vec{V}_i^{p_{i,0}} \cdot g^{\mathbf{L}_h \cdot \vec{\rho}_1} \end{aligned}$$

Note that the Fiat-Shamir proof does not hide which index $\ell \in \{0,1\}^n$ the signing oracle uses (and it does not have to since $\mathcal{A}$ knows $\ell$): indeed, for any signing query, the matrix $\mathbf{L}_h$ has only rank 3 and $\vec{X}_\ell$ may be the only key of the ring $\mathcal{R}$ to be in the column span of $\mathbf{M}_H$. However, the same holds in Game 6. As long as the simulation does not fail because of a collision on $\mathcal{H}_{\mathsf{FS}}$ or because $(H_1, H_2)$ accidentally lands in the span of $(g, h)$ at some signing query, the simulated proof is perfectly indistinguishable from a real proof that would be generated as in Game 6. Taking into account the probability that the signing oracle fails at some query, we obtain the inequality $|\Pr[W_7] - \Pr[W_6]| \leq Q_S/q + Q_S \cdot (Q_{\mathcal{H}_{\mathsf{FS}}} + Q_S)/q^2$.

In Game 7, we claim that $\Pr[W_7] = 2/q$. To prove this claim, we recall that $\mathcal{B}$ only outputs 1 if $(T_0^\star, T_1^\star)$ decrypts to $\Gamma^\star = U^{\alpha_{\ell^\star}} \cdot V^{\beta_{\ell^\star}}$. We next argue that, except with probability $1/q$, $\Gamma^\star$ is independent of $\mathcal{A}$'s view in Game 7.

Indeed, since $(g, h, \tilde{g}, \tilde{h})$ is a Diffie-Hellman tuple, the only information that $\vec{X}_{\ell^\star} = (X_{\ell^\star}, Y_{\ell^\star}) = (g^{\alpha_{\ell^\star}} \cdot h^{\beta_{\ell^\star}}, \tilde{g}^{\alpha_{\ell^\star}} \cdot \tilde{h}^{\beta_{\ell^\star}})$ reveals about $(\alpha_{\ell^\star}, \beta_{\ell^\star}) \in \mathbb{Z}_q^2$ is $\log_g(X_{\ell^\star}) = \alpha_{\ell^\star} + \log_g(h) \cdot \beta_{\ell^\star}$ since $\log_g(Y_{\ell^\star})$ only provides redundant information. Also, in all outputs of the signing oracle, the pair $(T_0, T_1) \xleftarrow{R} \mathbb{G}^2$ is chosen independently of $U^{\alpha_{\ell^\star}} \cdot V^{\alpha_{\ell^\star}}$. Finally, in Game 7, all signing queries are answered by simulating a NIZK proof without using the witnesses $SK_{\ell^\star} = (\alpha_{\ell^\star}, \beta_{\ell^\star}) \in \mathbb{Z}_q^2$ at any time. This ensures that no information is leaked about $(\alpha_{\ell^\star}, \beta_{\ell^\star})$ whatsoever.

Taking into account the event that $(U, V)$ accidentally falls in the span of $(g, h)$, we find that $\Gamma^\star$ remains independent of $\mathcal{A}$'s view until the forgery stage. In this case, $(T_0^\star, T_1^\star)$ only decrypts to $U^{\alpha_{\ell^\star}} \cdot V^{\beta_{\ell^\star}}$ with probability $1/q$, which implies $\Pr[W_7] = 2/q$. When counting probabilities, we obtain the bound (4.12). □ □

**Lemma 4.6.** *There exists an efficient DDH distinguisher $\mathcal{B}$ that bridges between Game 1 and Game 2 and such that $|\Pr[W_2] - \Pr[W_1]| \leq \mathbf{Adv}_{\mathcal{B}}^{\mathrm{DDH}}(\lambda) + 1/q$.*

*Proof.* We consider a DDH instance $(g, g^a, g^b, g^{ab+c})$ for which $\mathcal{B}$ has to decide if $c = 0$ or $c \in_R \mathbb{Z}_q$. To do this, $\mathcal{B}$ initially defines $h = g^b$ and emulates the random oracle $\mathcal{H}(\cdot)$ at each (direct) query by randomly choosing $\delta_1, \delta_2 \xleftarrow{R} \mathbb{Z}_q$ and setting $(H_1, H_2) = \big((g^a)^{\delta_1} \cdot g^{\delta_2}, (g^{ab+c})^{\delta_1} \cdot (g^b)^{\delta_2}\big) = \big(g^{a\delta_1+\delta_2}, g^{(a\delta_1+\delta_2)b+c\delta_1}\big)$. If $c = 0$, $(H_1, H_2)$ is distributed as in Game 2 for $\gamma = a\delta_1 + \delta_2$. If $c \in_R \mathbb{Z}_q$, we have $c \neq 0$ with probability $1 - 1/q$, so that $(H_1, H_2)$ are uniform over $\mathbb{G}^2$ and independently distributed across distinct queries, exactly as in Game 1. When $\mathcal{A}$ halts, $\mathcal{B}$ outputs 1 if $\mathcal{A}$ creates a valid forgery and 0 otherwise. □ □

**Lemma 4.7.** *From Game 3 to Game 4, the adversary's winning probabilities differ by at most $|\Pr[W_4] - \Pr[W_3]| \leq (1 + Q_{\mathcal{H}_{\mathsf{FS}}})/q$.*

*Proof.* We bound the probability $\Pr[F_4]$. Recall that $F_4$ occurs if $\mathcal{A}$ breaks the soundness of the proof that a dual-mode ciphertext encrypts a bit. This implies that $\sigma^\star = \big(\{\mathbf{\Sigma}_j^\star\}_{j=1}^n, T_0^\star, T_1^\star, z_{d,\alpha}^\star, z_{d,\beta}^\star, z_{d,\theta_1}^\star, z_{d,\theta_2}^\star\big)$ verifies and there exists $k \in [n]$ such that $\mathbf{\Sigma}_k^\star = (\vec{C}_{\ell_k}^\star, \vec{C}_{a_k}^\star, \vec{C}_{b_k}^\star, \vec{C}_{d_{k-1}^\star}^\star, f_k^\star, z_{r_k}^\star, z_{s_k}^\star, \bar{z}_{r_k}^\star, \bar{z}_{s_k}^\star)$ contains a ciphertext $\vec{C}_{\ell_k}^\star$ that decrypts to $\ell_k \notin \{0,1\}$. For this index $k$, $\sigma^\star$ contains a NIZK proof

$$\big((\vec{C}_{a_k}^\star, \vec{C}_{b_k}^\star), x, (f_k^\star, z_{r_k}^\star, z_{s_k}^\star, \bar{z}_{r_k}^\star, \bar{z}_{s_k}^\star)\big) \tag{4.14}$$

that $\vec{C}^{\star}_{\ell_k}$ encrypts $\ell^{\star}_k \in \{0,1\}$. This proof, which is obtained from the $\Sigma$-protocol of [GK15, Figure 1], is known [GK15, Theorem 2] to provide special soundness with soundness error $1/q$. Hence, if the statement is false and $\vec{C}^{\star}_{\ell_k}$ does not encrypt a bit, for any given pair $(\vec{C}^{\star}_{a_k}, \vec{C}^{\star}_{b_k})$, only one challenge value $x \in \mathbb{Z}_q$ admits a response $(f^{\star}_k, z^{\star}_{r_k}, z^{\star}_{s_k}, \bar{z}^{\star}_{r_k}, \bar{z}^{\star}_{s_k})$ that makes (4.14) into an accepting transcript.

At each query $\mathcal{H}_{\mathsf{FS}}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^{n})$ such that one of the $\{\vec{C}_{\ell_j}\}_{j=1}$ does not encrypt a binary value, the probability that oracle $\mathcal{H}_{\mathsf{FS}}(\cdot)$ returns the unique "bad" $x \in \mathbb{Z}_q$ for which a correct response exists is exactly $1/q$. Finally, since $\mathcal{H}_{\mathsf{FS}}$ is simulated by the challenger $\mathcal{B}$, we may assume that $\mathcal{B}$ makes the query

$$\mathcal{H}_{\mathsf{FS}}(M^{\star}, \mathcal{R}^{\star}, T^{\star}_0, T^{\star}_1, \{\vec{C}^{\star}_{\ell_j}, \vec{C}^{\star}_{a_j}, \vec{C}^{\star}_{b_j}, \vec{C}^{\star}_{d_{j-1}}\}_{j=1}^{n})$$

for itself in case it was not explicitly made by the time $\mathcal{A}$ terminates. Taking a union bound over all $\mathcal{H}_{\mathsf{FS}}$-queries, we obtain $|\Pr[W_4] - \Pr[W_3]| \leq \Pr[F_4] \leq (1 + Q_{\mathcal{H}_{\mathsf{FS}}})/q$. □ □

**Lemma 4.8.** *From Game 4 to Game 5, the adversary's winning probabilities differ by at most* $|\Pr[W_5] - \Pr[W_4]| \leq Q_{\mathcal{H}_{\mathsf{FS}}} \cdot \log(Q_V)/q$.

*Proof.* The only situation where Game 5 differs from Game 4 is the event $F_5$ that extracting $\{\vec{C}^{\star}_{\ell_j}\}_{j=1}^{n}$ leads to a string $\ell^{\star} \in \{0,1\}^n$ but $(T^{\star}_0, T^{\star}_1)$ does not decrypt to an encoding $U^{\alpha_{\ell^{\star}}} \cdot V^{\beta_{\ell^{\star}}}$ of the $\ell^{\star}$-th ring member's secret key. This implies that $\vec{V}_{\ell^{\star}} = (X_{\ell^{\star}}, Y_{\ell^{\star}}, T^{\star}_0, T^{\star}_1)$ is not in the column space of $\mathbf{M}_H$ (as defined in (4.3)) and we show that this event can only happen with probability $Q_{\mathcal{H}_{\mathsf{FS}}} \cdot n/q \leq Q_{\mathcal{H}_{\mathsf{FS}}} \cdot \log(Q_V)/q$, where $n = \log N^{\star}$.

Note that (4.10) implies that $f^{\star}_j$ equals $f^{\star}_j = a^{\star}_j + \ell^{\star}_j \cdot x^{\star}$ for all $j \in [n]$, where $a^{\star}_j \in \mathbb{Z}_q$ is encrypted by $\vec{C}^{\star}_{a_j}$. Defining $f^{\star}_{j,1} = f^{\star}_j$ and $f^{\star}_{j,0} = x - f^{\star}_j$, we know that

$$\prod_{j=1}^{n} f^{\star}_{j,i_j} = \delta_{i,\ell^{\star}} \cdot x^{\star n} + \sum_{k=0}^{n-1} p_{i,k} \cdot x^{\star k} \qquad \forall i \in [N^{\star}],$$

for some $p^{\star}_{i,0}, \ldots, p^{\star}_{i,n-1} \in \mathbb{Z}_q$. This implies

$$\begin{aligned}\prod_{i=0}^{N-1} \vec{V}_i^{\prod_{j=1}^{n} f^{\star}_{j,i_j}} &= \prod_{i=0}^{N-1} \vec{V}_i^{\delta_{i,\ell^{\star}} \cdot x^n + \sum_{k=0}^{n-1} p^{\star}_{i,k} \cdot x^k} \\ &= \vec{V}_{\ell^{\star}}^{x^n} \cdot \prod_{i=0}^{N-1} \vec{V}_i^{\sum_{k=0}^{n-1} p^{\star}_{i,k} \cdot x^k} = \vec{V}_{\ell^{\star}}^{x^n} \cdot \prod_{k=0}^{n-1} \cdot \Big(\prod_{i=0}^{N-1} \vec{V}_i^{p^{\star}_{i,k}}\Big)^{x^k}.\end{aligned}$$

Moreover, the last verification equation (4.11) implies

$$\vec{V}_{\ell^{\star}}^{x^n} \cdot \prod_{k=0}^{n-1} \cdot \Big(\prod_{i=0}^{N-1} \vec{V}_i^{p^{\star}_{i,k}}\Big)^{x^k} \cdot \prod_{k=0}^{n-1} \vec{C}_{d_k}^{-(x^k)} = g^{\mathbf{L}_h \cdot (z^{\star}_{d,\alpha}, z^{\star}_{d,\beta}, z^{\star}_{d,\theta_1}, z^{\star}_{d,\theta_2})^{\top}}. \tag{4.15}$$

By taking the discrete logarithms $\log_g(\cdot)$ of both members of (4.15), we get

$$x^n \cdot \vec{v}_{\ell^{\star}} + \sum_{i=0}^{N-1} \sum_{k=0}^{n-1} (p^{\star}_{i,k}\, x^k) \cdot \vec{v}_i - \sum_{k=0}^{n-1} x^k \cdot \vec{c}_{d_k} = \mathbf{L}_h \cdot (z^{\star}_{d,\alpha}, z^{\star}_{d,\beta}, z^{\star}_{d,\theta_1}, z^{\star}_{d,\theta_2})^{\top}. \tag{4.16}$$

Since $\mathbf{L}_h$ has rank at most 3 due to the modification introduced in Game 2 and Game 3, assuming that $\vec{v}_{\ell^\star} = \log_g(\vec{V}_{\ell^\star}) \in \mathbb{Z}_q^4$ is not in the column space of $\mathbf{L}_h$, there exists a non-zero vector $\vec{t} \in \mathbb{Z}_q^4$ such that $\vec{t}^\top \cdot \mathbf{L}_h = \mathbf{0}^{1\times 4}$ and $\vec{t}^\top \cdot \vec{v}_{\ell^\star} \neq 0$. If we multiply both members of (4.16) on the left by $\vec{t}^\top$, we obtain

$$x^n \cdot (\vec{t}^\top \cdot \vec{v}_{\ell^\star}) + \sum_{i=0}^{N-1} \sum_{k=0}^{n-1} (p^\star_{i,k} \cdot x^k) \cdot (\vec{t}^\top \cdot \vec{v}_i) - \sum_{k=0}^{n-1} x^k \cdot (\vec{t}^\top \cdot \vec{c}_{d_k}) = 0. \tag{4.17}$$

If $\vec{t}^\top \cdot \vec{v}_{\ell^\star} \neq 0$, equality (4.17) implies that $x$ is a root of a non-zero polynomial of degree $n$. However, $x$ is uniformly distributed over $\mathbb{Z}_q$ and the Schwartz-Zippel Lemma implies that (4.17) can only hold with probability $n/q < \log(Q_V)/q$.

In order to bound the probability $\Pr[F_5]$, we have to consider all hash queries

$$\mathcal{H}_{FS}(M, \mathcal{R}, T_0, T_1, \{\vec{C}_{\ell_j}, \vec{C}_{a_j}, \vec{C}_{b_j}, \vec{C}_{d_{j-1}}\}_{j=1}^n)$$

for which $\mathcal{R}$ only contains honestly generated keys and $(T_0, T_1)$ does not decrypt to an encoding $U^{\alpha_\ell} \cdot V^{\beta_\ell}$ of the $\ell$-th key of $\mathcal{R}$, where $\ell \in \{0, \ldots, |\mathcal{R}| - 1\}$ is determined by $\{\vec{C}_{\ell_j}\}_{j=1}^n$. Taking a union bound over all hash queries, we obtain $\Pr[F_5] \leq Q_{\mathcal{H}_{\mathsf{FS}}} \cdot \log(Q_V)/q$. □ □

# Lattice-based Designated-Verifiable NIZK Argument and Application to a Voting Scheme

5

## 5.1. Introduction

Non-interactive zero-knowledge (NIZK) proof systems have been introduced by Blum, Feldman and Micali [BFM88] and allow a prover to prove membership of an NP language without interactions. They can be used to convince anybody that a statement belongs to the language and their zero-knowledge property ensures that a proof reveals nothing beyond the membership of the language. NIZK proofs are fundamental cryptographic primitives used to construct public-key encryptions secure against chosen-ciphertext attacks, digital signatures, voting schemes and other cryptographic protocols.

Designated-verifier non-interactive zero-knowledge (DVNIZK) argument systems are argument systems which can only be verified with a secret verification key, the soundness of the system holds only for an adversary who does not know the secret key and its zero-knowledge property holds for everyone, even for an adversary who knows the secret key. DVNIZKs have many applications for building more complex cryptographic protocols including tightly CCA encryption schemes [GHKW16] and electronic voting systems [CG15].

Damgård, Fazio and Nicolosi [DFN06] have been the first to establish a generic transformation from a $\Sigma$-protocol, *i.e.* a 3-move honest zero-knowledge proof, into an efficient DVNIZK using an additively homomorphic encryption scheme if the answer can be computed linearly. Their construction is secure for a logarithmic number of proofs but the soundness relies on an unclassical complexity leveraging assumption. This technique has been later extended by Chaidos and Groth [CG15] to a variant of designated verifier argument, where the soundness is weakened, and called culpable soundness. They use the Okamoto-Uchiyama encryption scheme [OU98] and require that the original $\Sigma$-protocol is secure with respect to unique identifiable challenge, meaning that for words in a strict subset of the complementary of the original language, there is at most one challenge that can be answered with a valid proof. One interesting open problem is how to extend their idea to construct more general DVNIZK arguments based on other assumptions.

Lattice-based constructions. The possible development of quantum computers would give efficient solutions for many classical cryptographic problems. Fortunately, we still

have some cryptographic problems which remain hard even with quantum computers like lattice-based problems, code-based problems, isogeny-based problems *etc.* Lattice-based constructions are the most prominent ones, and they benefit from a security based on worst-case to average-case reductions. They also give us the most possibilities to construct various functionalities over cryptographic primitives (as fully homomorphic encryption (FHE) [Gen09; BV11; GSW13]...).

Most lattice-based constructions rely on two fundamental problems, the Learning with Errors (LWE) problem and the (Inhomogeneous) Small Integer Solution ((I)SIS) problem, both shown to be as hard as lattices problems with worst-case to average-case reductions [Ajt96; Reg09]. The ISIS problem is that: given a matrix $\mathbf{A}$ uniformly sampled in $\mathbb{Z}_q^{n\times m}$ (for some $m > n$), and a uniform vector $\mathbf{y} \in \mathbb{Z}_q^n$, we can not find a small vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{Ax} = \mathbf{y} \bmod q$. Then, in this area, an interesting relation to prove is the knowledge of an (I)SIS solution for a given matrix $\mathbf{A}$ and vector $\mathbf{y}$, i.e. a short vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{y} \bmod q$. There already exists many interactive proofs to show this [MV03; LNSW13; BCK+14; BKLP15]. One of the main difficulty in building a proof of knowledge for a lattice problem is that the proof has to be done many times, as it is the case for Stern proof [LNSW13], since the soundness probability is only 2/3 and extending the challenge space is a well-known hard problem. Many new results on how to improve the soundness probability have been published recently [BDOP16; BDLN16; PL17; LS17; BL17; BBC+18].

Usually, when we prove the soundness property of a proof of knowledge, we need to extract a witness. In the random oracle model, we can use the forking lemma [PS96] that allows us to rewind the proof and given two transcripts with the same commitment for two different challenges, we can extract the witness, proving that the proof is a proof of knowledge. In this case, the difference of these two challenges needs to be invertible. However, in the lattice world, the inverse has not always small coefficients and when we work in the efficiently-computable ring settings, it is possible that this difference will not be invertible, such as $\mathbb{Z}_q[X]/\langle X^n + 1\rangle$ for $q$ prime, $n$ a power of 2 and $q = 1 \mod 2n$. In [BCK+14], Benhamouda *et al.* showed that $2(X^i - X^j)$ are invertible with coefficients in $\{-1, 0, 1\}$, but the challenges space is very small and the proof still need to be done many times.

In [BDOP16; LN17], the authors described a proof of knowledge to approximate zero-knowledge proofs in one iteration in the random oracle model. The main idea is to build an approximate zero-knowledge proof since they are not able to prove that we can extract the witness but only that there exists a small $\mathbf{x}'$ such that $\mathbf{Ax}' = c\mathbf{y}$, where $c$ is a small number in $\mathbb{Z}_q$. This approximate solution is useful for some applications including verifiable encryption schemes, group signature, key escrow, optimistic fair exchanges and verifiable secret sharing.

Our Contributions. Our main contributions are the new construction of a lattice-based DVNIZK argument for the OR relation in the standard model and the first lattice-based voting scheme in the standard model, i.e. both without random oracle. In order to construct our scheme we will use sub-exponential LWE (i.e. modulus $q$ is sub-exponential w.r.t. the security parameter $n$).

- We first build a $\Sigma$-protocol which proves that a commitment is either a commit-

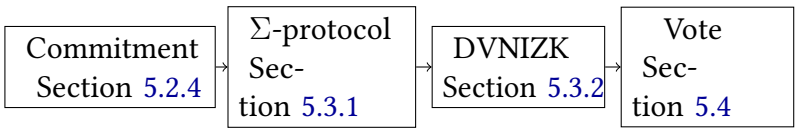


**Figure 5.1.** – Our contributions to build a voting scheme.

ment of $0$ or $1$. Our protocol is adapted from the $\Sigma$-protocol of Benhamouda *et al.* in [BKLP15], which shows that the plaintexts satisfy the relation $\mu_3 = \mu_1\mu_2$ for three encryptions. Our proof shows a slightly different relation: of the form $\mu = 0$ or $1$ if, and only if, $\mu(\mu - 1) = 0$. We also prove an additional security property: the soundness with unique identifiable challenge that is needed to build our DVNIZK (Section 5.3.1).

- Then, we use similar techniques as in [CG15] to derive a DVNIZK argument from the previous $\Sigma$-protocol. But this transformation does not directly apply and we had to solve several issues. This is our main contribution, described Section 5.3.2, our scheme allows to prove an OR relation: that a commitment commits to the message $0$ or $1$, and its security is proven under the Ring Learning With Errors assumption [LPR13b].
- Finally, we show how to use our DVNIZK scheme to build a voting schemes. We construct the first lattice-based voting scheme in the standard model in Section 5.4, but it requires an honest decryption party.

The principal idea of the transformation from a $\Sigma$-protocol to a DVNIZK proof is based on the observation that: in the $\Sigma$-protocol, the challenge generated by the verifier is independent from the output sent by the prover. Thus, we can use an decryptable homomorphic commitment scheme to commit a random challenge, and publish it in the public key at the beginning. The prover uses the commitment of the challenge and its homomorphic property to compute an committed proof. Thanks to the decryption key, the verifier can decrypt this proof and verify it, as in the $\Sigma$-protocol. But applying this transformation to a lattice-based commitment scheme is not straightforward. The main difficulty is that, in a lattice-based $\Sigma$-protocol, we usually need a rejection sampling procedure to make the proof independent from the witness. But in this transformation, the prover can only compute the proof using the ciphertexts, then this makes the rejection sampling on the plaintext impossible. To solve this issue, we require the message space to be a ring and the challenge space to be a subset of the message space with only invertible elements. We also need that the difference of two challenges is invertible $\bmod 2$, this explains our choice of the ring $\mathcal{R} = \mathbb{Z}[X]/\langle X^{2\cdot 3k} + X^{3k} + 1\rangle$ for our constructions (more details in Section 5.2.1).

Finally, we describe a lattice-based voting scheme in the standard model as an application of our DVNIZK. The main idea of the construction is that each ballot is a valid commitment. Using our DVNIZK, we can prove that the committed value is either a commitment of $0$ or $1$. This proof ensures that even an adversary can only vote at most one time during the voting procedure. However, our voting scheme needs an honest decryption party. The proof of validity using our DVNIZK is not sufficient to prove the homomorphic property of

the commitment. Because of this, we require an honest decryption party to add the votes privately and we cannot publicly add all the ballots together.

Upon our knowledge, there are two existing post-quantum e-voting schemes [CGGI16] and [PLNS17], both of them are based on the random oracle model.

**Open Problems.** In the first part, we construct a lattice-based DVNIZK argument system in the standard model. This argument system is only culpable soundness, which requires the successful attacker to reveal a witness of the fact the attack is in the guilt relation $R_{guilt}$. This weak soundness notion is sufficient in the case of voting scheme as mentioned in [CG15]. But it would be interesting to build a DVNIZK scheme which achieve the classical soundness property, in which the adversary cannot construct a proof of a false statement

We construct also the first post-quantum voting scheme in the standard model in Section 5.4.2. However, this first construction needs to have a trusted decryption party. The post-quantum voting scheme without trusted decryption party in the standard model remains as an open problem.

## 5.2. Preliminaries

Notations. Let $n > 0$ and $\mathbb{R}$ be the ring $\mathbb{Z}[X]/\langle f(X)\rangle$ for $f(X)$ an irreducible polynomial in $\mathbb{R}_q$ of degree $n$. For $q > 0$, we define $\mathbb{R}_q = \mathbb{R}/q\mathbb{R}$. The elements of $\mathbb{R}_2$ are then polynomials with coefficients in $\{0, 1\}$. We use bold lower case letter to denote vectors and bold capital letters to denote matrices. We note $[f]_2$ the binary rounding for every coefficient of the polynomial $f$. We use $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_\infty$ and $\|\mathbf{x}\|$ to denote respectively the $L_1$, $L_\infty$ and $L_2$ norm of the vector $\mathbf{x}$.

We define as follows the gadget matrix [MP12]. It was first introduced as an efficient tool to construct lattice trapdoors. Later on, the gadget matrix is widely used to construct fully homomorphic encryption schemes like [GSW13].

**Definition 5.1** (Gadget Matrix [MP12])**.** *For a vector* $\mathbf{g} = \begin{bmatrix}1 & 2 & \dots & 2^{\lfloor \log(q) \rfloor}\end{bmatrix}$ *and* $\mathbf{I}_m$ *the* $m \times m$ *identity matrix, we define the gadget matrix* $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_m$*. We also use the notation* $\mathbf{G}^{-1}$ *for the function which, given a matrix* $\mathbf{M} \in \mathbb{R}_q^{m \times m}$*, outputs* $\tilde{\mathbf{M}} \in \mathbb{R}_2^{m\lfloor \log_2(q) \rfloor \times m}$ *such that* $\mathbf{G} \cdot \tilde{\mathbf{M}} = \mathbf{M}$*.*

Notice that we can also see the function $\mathbf{G}^{-1}$ as a decomposition of each coefficient in bits and which put them in the same column.

### 5.2.1. Choice of the ring

In this work, we use $\mathcal{R} = \mathbb{Z}/\langle X^{2\cdot 3k} + X^{3k} + 1\rangle$. We need this particular ring as we have constraints on the polynomial $\Phi_3(X^{3^k}) = X^{2\cdot 3k} + X^{2k} + 1$: It needs to be irreducible in both $\mathbb{F}_2[X]$ and $\mathbb{F}_q[X]$. In this section, we show that this is indeed the case and give some properties on the rings $\mathbb{R} = \mathbb{Z}[X]/\langle X^{2\cdot 3^k} + X^{3^k} + 1\rangle$.

**Lemma 5.1** ([LN86, Th 3.35]). *Let $f(X)$ be an irreducible polynomial in $\mathbb{F}_q[X]$ of degree $d$ and multiplicative order[1] $r$. Let $t \geq 2$ be an integer whose prime factors divide $r$ but not $\frac{q^d-1}{r}$. We also need to assume $q^d = 1 \bmod 4$ if $t = 0 \bmod 4$. Then $f(X^t)$ is irreducible in $\mathbb{F}_q[X]$.*

**Lemma 5.2** ([LN86, Th 2.47]). *In the prime finite field $\mathbb{F}_q$, let $n$ be a positive integer and $r$ the smallest integer such that $q^r = 1 \bmod n$. Then all the irreducible factors of the $n$-th cyclotomic polynomial have degree $r$.*

**Lemma 5.3.** *Let $q = 2 \bmod 9$ be a prime and $k \geq 0$ a non-negative integer. The polynomial $X^{2 \cdot 3^k} + X^{3^k} + 1$ is an irreducible polynomial in both $\mathbb{F}_2[X]$ and $\mathbb{F}_q[X]$.*

In the rest of this work, we use the function $\Phi_3(X^{3^k}) = X^{2 \cdot 3^k} + X^{3^k} + 1$ and we apply the condition $q = 2 \bmod 9$ to generate the extension $\mathbb{F}_{q^{2 \cdot 3^k}}$ of $\mathbb{F}_q$. Recall that since $X^{2 \cdot 3^k} + X^{3^k} + 1$ is irreducible in $\mathbb{F}_q[X]$, $\mathbb{R}_q$ is a finite field. We denote by $n = 2 \cdot 3^k$ the extension degree of $\mathbb{R}_q$ over $\mathbb{F}_q$.

A critical value for the choice of the ring is the expansion factor. It measures how large the product of two polynomials becomes in the ring. In cases of lattice-based construction, smaller expansion factor leads to smaller parameters.

**Definition 5.2** (Expansion Factor of $\mathbb{R}_q$ [LM06]). *The expansion factor of a finite field $\mathbb{R}_q$, with $n$ the extension degree of $\mathbb{R}_q$ over $\mathbb{F}_q$, is defined as: $EF(\mathbb{R}_q) = \max_{g,h \in \mathbb{R}_q} \min\{k \in \mathbb{N}$ s.t. $\|g \cdot h\|_\infty \leq k \cdot n \cdot \|g\|_\infty \cdot \|h\|_\infty\}$.*

Let $f$ be the definition polynomial of the finite field $\mathbb{R}_q$. As showed in [LM06], the expansion factor $EF(\mathbb{R}_q)$ can be bounded by:

$$EF(\mathbb{R}_q) \leq \min_{g \in \mathbb{Z}[x]} 2 \cdot \|f\|_1 \cdot \|g\|_1 \cdot (2 \cdot \|f \cdot g\|_1)^{\lceil \frac{\deg(f)-2}{\text{gap}(f \cdot g)} \rceil}. \tag{5.1}$$

Where $\deg(\cdot)$ is the degree of a polynomial and $\text{gap}(\cdot)$ is the difference between the polynomial degree and the second highest degree of non-zero term in a polynomial. Using this bound with $g(x) = x^{3^k} - 1$, we get the following lemma.

**Lemma 5.4.** *If $\mathbb{R} = \mathbb{Z}[X]/\langle X^{2 \cdot 3^k} + X^{3^k} + 1 \rangle$, we have $EF(\mathbb{R}_q) \leq 48$.*

### 5.2.2. Gaussian distribution

For $v \in \mathbb{R}_q$, we define the Gaussian function centered at $c$ with standard deviation $\sigma$ as $\rho_{\sigma,c}(v) = e^{\frac{-\|v-c\|^2}{2\sigma^2}}$ and the discrete Gaussian distribution centered at $c$ over $\mathbb{R}_q$ as $\mathcal{D}_{\mathbb{R}_q,c,\sigma}(v) = \rho_{\sigma,c}(v) / \sum_{w \in \mathbb{R}_q} \rho_{\sigma,c}(w)$. If the center is 0, we denote this distribution $\mathcal{D}_{\mathbb{R}_q,\sigma}$.

**Lemma 5.5** ([Ban93, Lemma 1.5(i)]). *In $\mathbb{R}_q$ of extension degree $n$, an element $s \in \mathbb{R}_q$ sampled according to $\mathcal{D}_{\mathbb{R}_q,\sigma}$, and parameter $\sigma > 0$, we have: $\Pr_{s \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma}}[\|s\| > \sigma\sqrt{n}] < 2^{-\Omega(n)}$.*

[1] The multiplicative order of a polynomial $f$ with $f(0) \neq 0$ is the smallest positive integer $r$ such that $f$ divides $X^r - 1$.

We denote in the rest of this chapter $B_\sigma = \sigma\sqrt{n}$ a bound on the discrete Gaussian distribution which is verified with probability $1 - 2^{-\Omega(n)}$, where $n$ is the extension degree of $R_q$ over $\mathbb{F}_q$.

In our construction, we prove that the distribution of the proof is unrelated to the witness used by the prover. A common approach with rejection sampling would imply rejections on the plaintext space which is not possible here without the secret verification key. Thus we need to use smudging techniques which are based on the following lemma.

**Lemma 5.6** (Noise smudging [DGK+10, Theorem B.1]). *Let $q \in \mathbb{Z}$, $\sigma \in \mathbb{R}$ and $y \in \mathbb{R}_q$. The statistical distance between the distribution $\mathcal{D}_{\mathbb{R}_q,\sigma}$ and $y + \mathcal{D}_{\mathbb{R}_q,\sigma}$ is at most $\frac{\|y\|}{2\sigma}$.*

### 5.2.3. Ring Learning With Errors

We now recall the Ring Learning With Errors (RLWE) assumption used in the security proof of our scheme.

**Definition 5.3** (RLWE assumption [PR06; LPR13b]). *For the security parameter $\lambda \in \mathbb{N}$, set a degree parameter $n = n(\lambda)$, choose the degree $n$ cyclotomic polynomial $\Phi_n(X)$ and a prime integer $q = q(\lambda) \in \mathbb{Z}$, let the ring $\mathbb{R}$ be $\mathbb{Z}[X]/\langle\Phi_n(X)\rangle$, the quotient ring $\mathbb{R}_q = \mathbb{R}/q\mathbb{R}$ and $\mathcal{D}_{\mathbb{Z}^n,\sigma}$ for $\sigma \geq \omega(\sqrt{\log n})$, a distribution over the ring $\mathbb{R}_q$.*

*The ring learning with errors assumption $RLWE_{\mathbb{R}_q,\mathcal{D}_{\mathbb{Z}^n,\sigma}}$ states that for any $\ell = poly(\lambda)$ we have: $\{(a_i, a_i \cdot s + e_i)\}_{i\in[\ell]} \approx_c \{(a_i, u_i)\}_{i\in[\ell]}$, where $a_i$ and $u_i$ are sampled uniformly from $\mathbb{R}_q$, and $(s, e_i)$ from the error distribution $\mathcal{D}_{\mathbb{Z}^n,\sigma}$.*

We note that the polynomial we use in this paper $\Phi_3(X^{3^k}) = X^{2\cdot 3k} + X^{2k} + 1$ is the $2 \cdot 3k$-th cyclotomic polynomial. Thus the security of our scheme relies on the RLWE assumption, recently the relation between RLWE and Polynomial-LWE has been clarified by [RSW18].

We need the following lemma for the construction of the RLWE commitment, which shows that given one RLWE sample $(a, b = as + e)$ with the secret $s$ and the noise $e$, we can create many other RLWE instances without knowing the secret $s$.

**Lemma 5.7** ([BV11]). *Let $q$, $n$, $\sigma$ be the parameters of RLWE, and let $\sigma' \geq 2^{\omega(\log n)} \cdot \sigma$. Then, under the $RLWE_{\mathbb{R}_q,\mathcal{D}_{\mathbb{R}_q}}$ assumption, for $s, v, e, e' \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma}$, $a, a' \xleftarrow{R} \mathbb{R}_q$, $e'' \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma'}$ and $b = as + 2e$, we have:*
$(s, (a, b), (av + 2e', bv + 2e'')) \approx_c (s, (a, b), (a', a's + 2e''))$.

### 5.2.4. RLWE commitment scheme [LPR13a]

We present in this part a commitment scheme derived from the RLWE encryption of [LPR13a]. Note that it can also be seen as a fully homomorphic encryption scheme from [BV11] with operations RLWE.Add and RLWE.Mult over the ciphertexts. For clarity, we refer to the original paper [BV11] for the explicit construction of these two algorithms.

Setup$(1^\lambda)$: Given the security parameter $\lambda$, choose parameters $n$, $q$, $\sigma$, $\sigma' \geq 2^{\omega(\log n)} \cdot \sigma$ as specified in Lemma 5.8. Then, proceed as follows:

1. Sample polynomials $a \xleftarrow{R} \mathbb{R}_q$ from a uniform distribution and $s, e \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma}$ from a discrete Gaussian distribution.
2. Compute $b = as + 2e$.
3. Output the commitment key $\mathsf{ck} = (a, b)$ and the decryption key $\mathsf{dk} = s^2$.

$\mathsf{Commit}(\mathsf{ck}, \mu)$: Given $\mathsf{ck} = (a, b)$, commit the message $\mu \in \mathbb{R}_2$ as follows:

1. Sample the error terms $(v, e') \leftarrow \mathcal{D}_{R_q,\sigma}$ and $e'' \leftarrow \mathcal{D}_{R_q,\sigma'}$.
2. Compute the commitment $(c_0, c_1)$: $c_0 = bv + 2e'' + \mu$ and $c_1 = av + 2e'$.
3. Output the commitment and the opening information: $\mathsf{com} = (c_0, c_1), \mathsf{open} = (1, v, e', e'')$.

$\mathsf{Verify}(\mathsf{com}, \mathsf{open}, \mu)$: Given the commitment $\mathsf{com} = (c_0, c_1)$, the opening information $\mathsf{open} = (\gamma, v, e', e'')$, and the message $\mu$, the verification algorithm proceeds as follows:

1. Let $B = n \cdot EF(\mathbb{R}_q)B_\sigma$ and $B' = n \cdot EF(\mathbb{R}_q)B_{\sigma'} + \frac{n \cdot EF(\mathbb{R}_q)}{2}$ verify that all the vectors in the opening information are small:

$$\|v\|_\infty \leq B, \qquad \|e'\|_\infty \leq B, \qquad \|e''\|_\infty \leq B', \qquad \gamma \in \mathbb{R}_2 \backslash \{0\}.$$

2. Verify that the following equation is true:

$$\begin{bmatrix} bv + 2e'' + [\gamma \cdot \mu]_2 \\ av + 2e' \end{bmatrix} = \gamma \cdot \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}.$$

In the verification algorithm we relax the acceptance condition, then we can have a small factor $\gamma$ both in the commitment and the message *i.e.* $\gamma \mathbf{c} = \mathsf{Commit}(\mathsf{ck}, \gamma\mu; \mathbf{r})$. We note that this is an honestly generated RLWE commitment scheme that can be seen also as an encryption scheme, thus we also introduce the following decryption algorithm. In this case, a decryption key $\mathsf{dk} = s$ is defined in the Setup algorithm.

$\mathsf{Decrypt}(\mathsf{ck}, \mathsf{dk}, \mathsf{com})$: Given the commitment key $\mathsf{ck}$, the decryption key $\mathsf{dk} = s$ and an honestly generated commitment $\mathsf{com} = (c_0, c_1)$, the decryption algorithm outputs the message $\mu = \lfloor c_0 - c_1 s \rceil_2$.

In the following lemma, we state that even with the previous relaxation, the commitment is still binding for the message (the proof is in the full version).

**Lemma 5.8.** *Assume that $\sigma' > 2^{\omega(\log n)} \cdot \sigma$. Then, this commitment is computationally hiding if the RLWE$_{\mathbb{R}_q, D_{R_q,\sigma}}$ problem is hard, and is statistically binding.*

[2] In the Definition **??**, there is no decryption key for the commitment scheme. Here the provided decryption key will used in the construction of $\Sigma$-protocol

### 5.2.5. Ring-GSW encryption scheme [KGV16]

In this section, we present a construction of a fully homomorphic encryption scheme based on the Ring-LWE assumption [KGV16], which is a ring version of the GSW scheme [GSW13]. In fact, we only require that it is linearly homomorphic. The advantage of the following encryption scheme is that the message space $\mathbb{R}_q$ is very large, which allows to encrypt the noise polynomials of the RLWE based commitment in the $\Sigma$-protocol.

$\mathsf{Setup}(1^\lambda)$: Given $\lambda > 0$, the security parameter of the encryption scheme, choose parameters $q$, $\tau$, $\tau'$ as specified in the underlying RLWE assumption, then proceeds as follows:

1. Sample polynomials $t \leftarrow \mathcal{D}_{\mathbb{R}_q,\tau}$, $u \xleftarrow{R} \mathbb{R}_q$ and noise $f \leftarrow \mathcal{D}_{\mathbb{R}_q,\tau}$.
2. Compute $w = t \cdot u + f \in \mathbb{R}_q$.
3. Set $\mathsf{SSK} = \mathbf{t} = [1 \ -t]^T, \mathsf{PK} = \mathbf{u} = [w \mid u]$. with $\mathbf{t}^T \cdot \mathbf{u} = \begin{bmatrix}1 & -t\end{bmatrix} \cdot \begin{bmatrix}w \\ u\end{bmatrix} = f$.

$\mathsf{Encrypt}(\mathsf{PK}, \mu)$: Given a public key $\mathsf{PK} = \mathbf{u}$ and a message $\mu$ in $\mathbb{R}_q$:

1. Set $N = \log_2(q)$, sample an uniform $(1 \times N)-$matrix $\mathbf{R} \xleftarrow{R} \mathbb{R}_2^{1\times N}$.
2. Sample a $(2 \times N)-$noise matrix $\mathbf{F} \leftarrow \mathcal{D}^{2\times N}_{(\mathbb{R}_q,\tau')}$.
3. Compute $\mathbf{C}$ the ciphertext of $\mu \in \mathbb{R}_q$ using the gadget matrix $\mathbf{G}$ (Definition 5.1): $\mathbf{C} = \mu \cdot \mathbf{G} + \mathbf{u} \cdot \mathbf{R} + \mathbf{F}$.

$\mathsf{Decrypt}(\mathsf{SSK}, \mathbf{C})$: Given the secret key $\mathsf{SSK} = \mathbf{t}$ and a ciphertext $\mathbf{C}$:

1. Compute the plaintext as follows:
   $\mathbf{t}^T \cdot \mathbf{C} = \mu \cdot \mathbf{t}^T \cdot \mathbf{G} + \mathbf{t}^T \cdot \mathbf{u} \cdot \mathbf{R} + \mathbf{t}^T \cdot \mathbf{F} = \mu \cdot \mathbf{t}^T \cdot \mathbf{G} + \text{error}$.
   Then, as the first coefficient of $\mathbf{t}$ is 1, we have $\mu \cdot \begin{bmatrix}1 & 2 & \cdots & 2^{\ell-1}\end{bmatrix} + \text{error}$. Since our ring is $\mathbb{R}_q$ with $q$ not a power of 2, the inversion is not straightforward, but as explained in [MP12], it is possible to use Babai algorithm [Bab86] to recover the value $\mu$.

$\mathsf{Add}(\mathbf{C}_1, \mathbf{C}_2)$: To add two ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, simply outputs $\mathbf{C}_1 + \mathbf{C}_2$.

$\mathsf{Mult}(\mathbf{C}, k)$: To multiply a ciphertext $\mathbf{C}$ by a scalar $k$, outputs $\mathbf{C} \cdot \mathbf{G}^{-1}(k \cdot \mathbf{G})$.

This scheme is IND-CPA secure under the $\mathrm{RLWE}_{\mathbb{R}_q,\mathcal{D}_{\mathbb{R}_q,\tau}}$ assumption (and we can set $\tau' = \tau$) [GSW13; KGV16] .

## 5.3. DV-NIZK Argument for an OR-gate

Our first contribution is a DVNIZK argument to show that a given RLWE commitment is a commitment of either 0 or 1. Note that this relation is non-linear as it can be written as: given $\mathbf{c}$ a commitment of $\mu$, show that the plaintext satisfies the relation $\mu(\mu - 1) = 0$. To the best of our knowledge, only Benhamouda *et al.* in [BKLP15] described previously a

non-linear relation $\Sigma$-protocol, but there is no such existing argument concerning DVNIZK constructions.

We present in the first part a $\Sigma$-protocol which proves that a commitment commits to either $0$ or $1$. Then, in the second part, we use similar techniques as in [CG15] to transform the $\Sigma$-protocol into a DVNIZK argument.

### 5.3.1. $\Sigma$-protocol to prove a OR-gate for a RLWE commitment

We first show how to construct a $\Sigma$-protocol to prove that $\mathbf{c} = \mathsf{Com.Commit}(\mathsf{ck}, \mu; \mathbf{r})$ commits to either $\mu = 0$ or $\mu = 1$. The principle of the protocol is explained in Figure 5.2. The $\Sigma$-protocol conducts the following steps: the prover first computes $\mathbf{c}_\alpha$, a commitment of a random element $\mu_\alpha \in \mathbb{R}_2$, and $\mathbf{c}_\beta$, a commitment of $\mu_\beta = -\mu\mu_\alpha$. Then, after receiving the random challenge $\delta$ from the verifier, the prover computes the proof $\pi = (\gamma, \mathbf{r}_\gamma, \mathbf{r}_0)$, with $\gamma = \delta\mu + \mu_\alpha$, such that:

$$\delta\mathbf{c} + \mathbf{c}_\alpha = \mathsf{Com.Commit}(\mathsf{ck}, \gamma; \mathbf{r}_\gamma), \tag{5.2}$$

$$(\gamma - \delta)\mathbf{c} + \mathbf{c}_\beta = \mathsf{Com.Commit}(\mathsf{ck}, 0; \mathbf{r}_0). \tag{5.3}$$

To accept the proof, the verifier checks the above equalities and that $(\mathbf{r}_\gamma, \mathbf{r}_0)$ is smaller than some given bounds. Intuitively, by the linearity of the commitment scheme, verifying these two equations implies that $\delta\mu + \mu_\alpha = \gamma$ and $(\gamma - \delta)\mu + \mu_\beta = 0$. Combining them together we have $\delta(\mu^2 - \mu) + \mu\mu_\alpha + \mu_\beta = 0$, which implies the relation we want to prove: that $(\mu - 1)\mu = 0$.

| Prover | | Verifier |
|---|---|---|
| | $(\mathsf{PP}, \mathbf{ck}) \leftarrow \mathsf{Setup}\,(1^n)$ | |
| $(\mathbf{c}_\alpha, \mathbf{c}_\beta) \leftarrow \mathsf{Prove}_1\,(\mathbf{ck}, \mathbf{c}, (\mu, \mathbf{r}))$ | | |
| | $\xrightarrow{(\mathbf{c}_\alpha, \mathbf{c}_\beta)}$ | |
| | $\xleftarrow{\delta}$ | |
| $\pi \leftarrow \mathsf{Prove}_2\,(\mathbf{ck}, \mathbf{c}, (\mu, \mathbf{r}), \delta)$ | | |
| | $\xrightarrow{\pi}$ | $\mathsf{Verify}\,(\mathbf{ck}, \mathbf{c}; \pi)$ |

**Figure 5.2.** – $\Sigma$-protocol to show that $\mathbf{c}$ commits to $\mu \in \{0, 1\}$.

As described above, the security of the $\Sigma$-protocol is based on the fact that the prover only have a small probability to correctly guess the value of $\delta$ before running $\mathsf{Prove}_2$. Thus, we define the challenge space with size $2^\xi$, where $\xi$ can be fixed on any positive integer value smaller than the extension degree $n$ of $\mathbb{R}_q$. The parameter $\xi$ can be consider as the security parameter of the $\Sigma$-protocol.

$\mathsf{Setup}(1^\lambda)$: Given the security parameter $\lambda > 0$, choose parameters $q$, $\sigma$ and $\sigma'$ as described in the RLWE assumption. Then, proceeds as follows:

1. Define $\mathsf{PP}_{RLWE} = (\mathbb{R}_q, \mathbb{R}_2, \mathcal{D}_{\mathbb{R}_q,\sigma}, \mathcal{D}_{\mathbb{R}_q,\sigma'})$.

2. Define $\sigma_\alpha, \sigma_\beta$ and $\sigma'_\alpha, \sigma'_\beta$ standard deviations super-polynomially larger than $\sigma$ and $\sigma'$.
3. Generate the commitment key $\mathsf{ck} = (a, b)$ and the decryption key $\mathsf{dk} = s$.

$\mathsf{Prove}_1(\mathsf{PP}, \mathsf{ck}, \mathbf{c}, (\mu, \mathbf{r}))$: Given the commitment key ck, the prover generates two commitments for two new elements $\mu_\alpha$ and $\mu_\beta$.

1. Uniformly sample a polynomial $\mu_\alpha \xleftarrow{R} \mathbb{R}_2$ and the randomness $(v_\alpha, e'_\alpha, e''_\alpha) \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma_\alpha} \times \mathcal{D}_{\mathbb{R}_q,\sigma_\alpha} \times \mathcal{D}_{\mathbb{R}_q,\sigma'_\alpha}$ which will be used to commit $\mu_\alpha$.
2. Compute $\mu_\beta = -\mu\mu_\alpha$ and similarly as before sample the randomness $(v_\beta, e'_\beta, e''_\beta) \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma_\beta} \times \mathcal{D}_{\mathbb{R}_q,\sigma_\beta} \times \mathcal{D}_{\mathbb{R}_q,\sigma'_\beta}$ which will be used to commit $\mu_\beta$.
3. Compute the commitments:
$\mathbf{c}_a = \mathsf{Com.Commit}(\mathsf{ck}, \mu_\alpha; (v_\alpha, e'_\alpha, e''_\alpha)), \mathbf{c}_b = \mathsf{Com.Commit}(\mathsf{ck}, \mu_\beta; (v_\beta, e'_\beta, e''_\beta))$
$= \begin{bmatrix} bv_\alpha + 2e''_\alpha + \mu_\alpha \\ av_\alpha + 2e'_\alpha \end{bmatrix}, \quad = \begin{bmatrix} bv_\beta + 2e''_\beta + \mu_\beta \\ av_\beta + 2e'_\beta \end{bmatrix}.$
4. Send the commitments $\mathbf{c}_\alpha, \mathbf{c}_\beta$ to the verifier.

$\mathsf{Prove}_2(\mathsf{PP}, \mathsf{ck}, \mathbf{c}, (\mu, \mathbf{r}), \delta)$: After giving the commitments to the verifier, the prover receives a challenge $\delta$ from the verifier which is uniformly sampled from all polynomials with degree less than $\xi$.

1. Compute $\gamma = \delta\mu + \mu_\alpha \bmod 2$.
2. Compute also the following randomness: $v_\gamma = \delta v + v_\alpha, e'_\gamma = \delta e' + e'_\alpha, e''_\gamma = \delta e'' + e''_\alpha + \lfloor \frac{\delta\mu+\mu_\alpha}{2} \rceil, v_0 = (\gamma - \delta)v + v_\beta, e'_0 = (\gamma - \delta)e' + e'_\beta, e''_0 = (\gamma - \delta)e'' + e''_\beta + \lfloor \frac{(\gamma-\delta)\mu+\mu_\beta}{2} \rceil$.
3. Output the proof to the verifier $\pi = (\gamma, \mathbf{r}_\gamma = (v_\gamma, e'_\gamma, e''_\gamma), \mathbf{r}_0 = (v_0, e'_0, e''_0))$.

$\mathsf{Verify}(\mathsf{PP}, \mathsf{ck}, \mathbf{c}; \pi)$: Given the commitment and its key, the verifier checks the proof $\pi = (\gamma, \mathbf{r}_\gamma = (v_\gamma, e'_\gamma, e''_\gamma), \mathbf{r}_0 = (v_0, e'_0, e''_0))$ as follows:

1. Let $B = n \cdot EF(\mathbb{R}_q) \cdot B_\sigma$ and $B' = n \cdot EF(\mathbb{R}_q) \cdot B_{\sigma'} + \frac{n \cdot EF(\mathbb{R}_q)+1}{2}$.
Verify that all the randomness are small:
$|v_\gamma| \leq B + B_{\sigma_\alpha}, |e'_\gamma| \leq B + B_{\sigma_\alpha}, |e''_\gamma| \leq B' + B_{\sigma'_\alpha}$,
$|v_0| \leq 2 \cdot B + B_{\sigma_\beta}, |e'_0| \leq 2 \cdot B + B_{\sigma_\beta}, |e''_0| \leq 2 \cdot B' + B_{\sigma'_\beta}$.
2. Verify that the following equations about the ciphertexts are correct: $\delta \cdot \mathbf{c} + \mathbf{c}_\alpha = \mathsf{Com.Commit}(\mathsf{ck}, \gamma; \mathbf{r}_\gamma)$,
$(\gamma - \delta) \cdot \mathbf{c} + \mathbf{c}_\beta = \mathsf{Com.Commit}(\mathsf{ck}, 0; \mathbf{r}_0)$.
3. If the above equations are verified, then output True, otherwise False.

**Theorem 5.9.** *The above $\Sigma$ protocol is statistically complete, 2-special sound, statistical Special Honest Verifier Zero Knowledge (SHVZK) and almost unique identifiable challenge with respect to the guilt relation $R_{guilt} = \{((\mathbf{c}, \mathsf{ck}), \mathsf{dk}) |$ KeyVerify$(\mathsf{ck}, \mathsf{dk}) = \mathsf{True} \wedge \mu \notin \{0, 1\}$ where $\mu =$ Decrypt$(\mathsf{ck}, \mathsf{dk}, \mathbf{c})\}$, where KeyVerify is an algorithm to verify that the decryption key is a decryption key w.r.t. the public key, if the underlying commitment is computationally hiding and statistically binding.*

### 5.3.2. Construction of a DVNIZK for encryption of 0 or 1

We use the [CG15] transformation in the lattice setting to build a DVNIZK scheme which proves that a commitment commits to the message 0 or 1 under the RLWE assumption. The main idea behind this transformation is that the verifier encrypts the challenge of the $\Sigma$-protocol. Since the challenge is independent from the outputs of $\mathsf{Prove}_1$, the verifier can generate it and publish its encryption in the public key. Then, the prover generates an encrypted proof using the ciphertext of the challenge (but without knowing the challenge) and its homomorphic properties, this part is efficient since there are only linear operations in $\mathsf{Prove}_2$. In the end, the verifier decrypts everything and runs the verification algorithm as in the $\Sigma$-protocol.

#### Principle of the construction.

At a high level, the construction is:

$\mathsf{Setup}(1^\lambda)$**:** Given $\lambda$, define the parameters of the scheme, then:

1. Generate the commitment key $\mathsf{ck}$.
2. Generate the RGSW public and secret keys: $\mathsf{PK}_{\text{RGSW}}$ and $\mathsf{SSK}_{\text{RGSW}}$.
3. Sample a challenge $\delta \xleftarrow{R} \{f \in R_2 | \deg f < \xi\}$, compute its encryption
$$\mathbf{C}_\delta = \text{RGSW.}\mathsf{Encrypt}(\mathsf{PK}_{\text{RGSW}}, \delta; \mathbf{r}_\delta).$$
4. Output the secret verification key $\mathsf{SVK} = \mathsf{SSK}_{\text{RGSW}}$ and the public key
$$\mathsf{PK} = (\mathsf{ck}, \mathsf{PK}_{\text{RGSW}}, \mathbf{C}_\delta).$$

$\mathsf{Prove}(\mathsf{PP}, \mathsf{PK}, (\mathbf{c}, (\mathbf{r}, \mu)))$**:** To prove that $\mathbf{c}$ is either a commitment of 0 or 1:

1. Uniformly sample a random message $\mu_\alpha \xleftarrow{R} \mathbb{R}_2$ and $\mu_\beta = -\mu\mu_\alpha$.
2. $\mathbf{c}_a = \text{Com.}\mathsf{Commit}(\mathsf{ck}, \mu_\alpha; r_\alpha)$ and $\mathbf{c}_b = \text{Com.}\mathsf{Commit}(\mathsf{ck}, \mu_\beta; r_\beta)$.
3. Homomorphically compute:
   - $\mathbf{C}_\gamma$ the RGSW encryption of $\gamma = \delta\mu + \mu_\alpha$,
   - $\mathbf{C}_{\mathbf{r}_\gamma}$ the RGSW encryption of $\mathbf{r}_\gamma$, the randomness of $\gamma = \delta\mu + \mu_\alpha$,

- $\mathbf{C}_{\mathbf{r}_0}$ the RGSW encryption of $\mathbf{r}_0$, the randomness of $(\gamma-\delta)\mu+\mu_\beta$.

4. Output the proof $\mathbf{\Pi}=(\mathbf{c}_\alpha,\mathbf{c}_\beta,\mathbf{C}_\gamma,\mathbf{C}_{\mathbf{r}_\gamma},\mathbf{C}_{\mathbf{r}_0})$.

$\mathsf{Verify}(\mathsf{PK},\mathsf{SVK},\mathbf{c},\mathbf{\Pi})$: To verify the proof,

1. Compute $\gamma=\text{RGSW.}\mathsf{Decrypt}(\mathsf{SSK}_{\text{RGSW}},\mathbf{C}_\gamma)$.
2. Decrypt also $\mathbf{C}_{\mathbf{r}_\gamma}$ and $\mathbf{C}_{\mathbf{r}_0}$, verify that all the randomness are small.
3. Then, using that $\delta=\text{RGSW.}\mathsf{Decrypt}(\mathsf{SSK}_{\text{RGSW}},\mathbf{C}_\delta)$, verify that:

$$\begin{aligned}\delta\cdot\mathbf{c}+\mathbf{c}_\alpha&=\text{Com.}\mathsf{Commit}(\mathsf{ck},\gamma;\mathbf{r}_\gamma),\\(\gamma-\delta)\cdot\mathbf{c}+\mathbf{c}_\beta&=\text{Com.}\mathsf{Commit}(\mathsf{ck},0;\mathbf{r}_0).\end{aligned}$$

4. If all the verification are true then return $\mathsf{True}$ otherwise return $\mathsf{False}$.

This is the main idea behind the protocol, but two steps in particular are more technical. We solved the following issues:

- **Number of challenges**. In the $\mathsf{Setup}$ algorithm, we have to encrypt $\lceil n/\xi\rceil$ challenges $(\delta^k)_{k\in[0,\lceil n/\xi\rceil)}$ and denote by $\mathbf{C}_{\delta^k}$ their RGSW encryption.
  Indeed, we use the extractor of the $\Sigma$-protocol to be able to show the culpable soundness property of the DVNIZK scheme. However, the efficiency of the extractor is linear in the size of the challenge space of the $\Sigma$-protocol, i.e., $2^\xi$. Then, we cannot take the same $\xi$, which was defined as the security parameter, we have to choose a smaller one (as a concrete example we could set $\xi$ as 20). In order to keep the same security level, we then have to repeat the $\Sigma$-protocol $\lceil n/\xi\rceil$ times.

- **Homomorphic operations on RGSW ciphertexts.** In the protocol, we compute $\gamma=\delta\mu+\mu_\alpha\in\mathbb{R}_2$. But we use in our scheme the RGSW encryption scheme, in which the operations of the plaintexts are in $\mathbb{R}_q$. As a consequence, we need to implement our own operation $[\cdot]_2$ in $\mathbb{R}_q$, which is not trivial using only the linear operations of the RGSW encryption scheme. Our main observations are that:
  - We can compute the bit addition of two RGSW ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, we denote by $\mathbf{C}_1\oplus\mathbf{C}_2=(\mathbf{C}_1-\mathbf{C}_2)\cdot\mathbf{G}^{-1}(\mathbf{C}_1-\mathbf{C}_2)$.
  - Using the special structure of the ring $\mathbb{R}_q=\mathbb{Z}_q[X]/\langle X^n+X^{n/2}+1\rangle$, for all polynomial $\delta$ with degree less than $\xi$, given $\{\mathbf{C}_{\delta_i}\}_{i\in[0,\xi)}$ an RGSW encryption of the coefficient of degree $i$ in the polynomial $\delta$, we can easily compute an encryption of the coefficient of degree $i$ in the polynomial $[\delta\cdot X^j]_2$ that we denote by $\mathbf{C}_{\delta_{i,j}}$.

  Using these two ideas together, we compute $\mathbf{C}_{[\delta\mu]_2[i]}=\bigoplus_{\forall j\in[0,n)\wedge\mu[j]=1}\mathbf{C}_{\delta_{i,j}}$, it is an encryption of the $i$-th coefficient of $[\delta\mu]_2$ and $\mathbf{C}_{[\gamma]_2}=\mathbf{C}_{[\delta\mu]_2}\oplus\mathbf{C}_{[\mu_\alpha]_2}$.

**Detailed DVNIZK construction.**

We now describe our full construction:

$\mathsf{Setup}(1^\lambda)$**:** Given the security parameter $\lambda > 0$, choose parameters $q$, $q' > q$, $\sigma$, $\sigma'$ (for the RLWE commitment scheme), $\tau$ and $\tau'$ (for the RGSW encryption scheme). Then, proceeds as follows:

1. Define: $\mathsf{PP}_{\text{RLWE}} = (\mathcal{D}_{\mathbb{R}_q,\sigma}, \mathcal{D}_{\mathbb{R}_q,\sigma'})$ and $\mathsf{PP}_{\text{RGSW}} = (q', \mathcal{D}_{\mathbb{R}_{q'},\tau}, \mathcal{D}_{\mathbb{R}_{q'},\tau'})$.
2. Sample $s, e \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma}$ and $a \xleftarrow{R} \mathbb{R}_q$, then generate a commitment key $\mathsf{ck} = (a, b = as + 2e)$.
3. Sample $t, f \leftarrow \mathcal{D}_{\mathbb{R}_{q'},\tau}$ and $u \xleftarrow{R} \mathbb{R}_{q'}$, compute the key pair for the underlying RGSW scheme: $\mathsf{PK}_{\text{RGSW}} = \begin{bmatrix} tu + f \\ u \end{bmatrix}$, $\mathsf{SSK}_{\text{RGSW}} = \begin{bmatrix} 1 \\ -t \end{bmatrix}$.
4. For all $k \in [1, \lceil n/\xi \rceil]$, sample $\delta^k$ uniformly at random from the set of all polynomials with degree less than $\xi$ in $\mathbb{R}_2$. Then, compute the RGSW encryption $\mathbf{C}_{\delta^k}$ of $\delta^k$: $\mathbf{C}_{\delta^k} = \text{RGSW.Encrypt}(\mathsf{PK}_{\text{RGSW}}, \delta^k; \mathbf{r}_{\delta^k})$ where $\mathbf{r}_{\delta^k}$ is a randomness generated as specified in the underlying schemes.
5. For all $i \in [0, \xi)$, we denote $\delta_i^k$ the coefficient of degree $i$ of $\delta^k$.
6. Compute $\mathbf{C}_{\delta_i^k}$ the RGSW encryption of $\delta_i^k$ for all $k \in [1, \lceil n/\xi \rceil]$ and for all $i \in [0, \xi)$.
7. Output the secret verification key $\mathsf{SVK} = \mathsf{SSK}_{\text{RGSW}}$, the public parameter $\mathsf{PP} = (\mathsf{PP}_{\text{RLWE}}, \mathsf{PP}_{\text{RGSW}})$ and the public key $\mathsf{PK} = (\mathsf{ck}, \mathsf{PK}_{\text{RGSW}}, \{\mathbf{C}_{\delta^k}\}_{k\in[1,\lceil n/\xi \rceil]}, \{\mathbf{C}_{\delta_i^k}\}_{k\in[1,\lceil n/\xi \rceil],i\in[0,\xi)})$.

$\mathsf{Prove}(\mathsf{PP}, \mathsf{PK}, (\mathbf{c}, (\mathbf{r}, \mu)))$**:** To prove that $\mathbf{c}$ is either a commitment of $0$, or a commitment of $1$ using $(\mathbf{r}, \mu)$ as a witness, we run the following proof $\lceil n/\xi \rceil$ times with $\{\delta_k\}_{k\in[1,\lceil n/\xi \rceil]}$. For clarity, we simply describe one proof with $\delta$ as follows:

1. Parse the public key as $\mathsf{PK} = (\mathsf{ck}, \mathsf{PK}_{\text{RGSW}}, \{\mathbf{C}_{\delta^k}\}_{k\in[1,\lceil n/\xi \rceil]}, \{\mathbf{C}_{\delta_i^k}\}_{k\in[1,\lceil n/\xi \rceil],i\in[0,\xi)})$.
2. Parse $(\mathbf{r}, \mathsf{ck}) = ((v, e', e''), (a, b))$.
3. Uniformly sample a random message $\mu_\alpha \xleftarrow{R} \mathbb{R}_2$ and $\mu_\beta = -\mu\mu_\alpha$.
4. Sample the randomness $v_\alpha, e'_\alpha, v_\beta, e'_\beta \leftarrow \mathcal{D}_{R_q,\sigma}$ and $e''_\alpha, e''_\beta \leftarrow \mathcal{D}_{\mathbb{R}_q,\sigma'}$.
5. Use the above generated randomness to commit $\mu_\alpha$, $\mu_\beta$ w.r.t. $\mathsf{ck}$ as follows: $\mathbf{c}_\alpha = \begin{bmatrix} bv_\alpha + 2e''_\alpha + \mu_\alpha \\ av_\alpha + 2e'_\alpha \end{bmatrix}$, $\mathbf{c}_b = \begin{bmatrix} bv_\beta + 2e''_\beta + \mu_\beta \\ av_\beta + 2e'_\beta \end{bmatrix}$
6. Using the homomorphic properties of the RGSW encryption, compute $(\mathbf{C}_\gamma, \mathbf{C}_{\mathbf{r}_\gamma}, \mathbf{C}_{\mathbf{r}_0})$, which correspond to the RGSW encryption of $\delta\mu + \mu_\alpha$ and of the randomness of the commitments of $\gamma$ and $0$. We also let $\sigma_{\mathsf{smudge}} = EF(\mathbb{R}_q) \cdot n \cdot 2^\lambda \cdot 2\log q$ where $\lambda$ is the security parameter. We draw $\mathbf{R}_\gamma, \mathbf{R}^{(i)}_{v_\gamma}, \mathbf{R}^{(i)}_{e'_\gamma}, \mathbf{R}^{(i)}_{e''_\gamma}, \mathbf{R}_0, \mathbf{R}^{(i)}_{v_0}, \mathbf{R}^{(i)}_{e'_0}, \mathbf{R}^{(i)}_{e''_0} \leftarrow \mathcal{D}_{\mathbb{R}_q^{2\log q \times 2\log q}, \sigma_{\mathsf{smudge}}}$.

7. To compute $\mathbf{C}_{[\gamma]_2} = \mathbf{C}_{[\delta\mu+\mu_\alpha]_2}$, as explained in the introduction we can compute $\mathbf{C}_{[\gamma]_2} = \mathbf{C}_{[\delta\mu]_2} \oplus \mathbf{C}_{[\mu_\alpha]_2}$ and $\mathbf{C}_{[\delta\mu]_2[i]} = \bigoplus_{\forall j\in[0,n)\wedge\mu[j]=1} \mathbf{C}_{\delta_{i,j}}$ which is an encryption of $i$-th coefficient of $[\delta\mu + \mu_\alpha]_2$. We recall that $\mathbf{C}_{\delta_{i,j}}$ is an encryption of the coefficient of degree $i$ in the polynomial $[\delta \cdot X^j]_2$. We give the details of the computation as follows:
   a) For $j \in [0, n)$ and $i \in [0, n)$, we first compute $\mathbf{C}_{\delta_{i,j}}$. Since $\delta$ is a polynomial of degree less than $\xi$, and $\mathbb{R}_q = \mathbb{Z}_q[X]/\langle X^n + X^{n/2} + 1\rangle$ where $n = 2 \cdot 3^k$ for some integer $k$, there are many coefficients in $[\delta \cdot X^j]_2$ are $0$ only depend on the degree of $\delta$. We define $\mathbf{C}_{\delta_{i,j}} = 0$ if the degree $i$'s coefficient of $[\delta \cdot X^j]_2$ is $0$ what ever the value of $\delta$ is. Then, for all $(i, j) \in [0, n)^2$, we compute $\mathbf{C}_{\delta_{i,j}}$ as follows $\mathbf{C}_{\delta_{i,j}} = \mathbf{C}_{\delta_{i-j}} \oplus \mathbf{C}_{\delta_{i+n-j}} \oplus \mathbf{C}_{\delta_{i+n/2-j}}$, where $\mathbf{C}_{\delta_i}$ with negative $i$ are all $0$.
   b) We compute the ciphertext $\mathbf{C}_{[\gamma]_2}$, which is the encryption of $\gamma \bmod 2$:
      i. Compute $\mathbf{C}_{[\delta\mu]_2[i]} = \bigoplus_{\forall j\in[0,n)\wedge\mu[j]=1} \mathbf{C}_{\delta_{i,j}}$, which represents the $i$-th coefficient of the polynomial $\delta\mu \in \mathcal{R}_2$.
      ii. Compute $\mathbf{C}_{[\gamma]_2} = \sum_{i=0}^{n-1}(\mathbf{C}_{(\delta\mu)_i} \oplus (\mu_a)[i] \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(X^i \cdot \mathbf{G}) + \mathsf{PK}_{\mathsf{RGSW}} \cdot \mathbf{R}_\gamma$.
8. Then we compute the following elements:

$$
\begin{aligned}
\mathbf{C}_\gamma =& \mathbf{C}_\delta \cdot \mathbf{G}^{-1}(\mu \cdot \mathbf{G}) + \mu_a \cdot \mathbf{G} + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_\gamma,\\
\mathbf{C}_{\mathbf{r}_\gamma} =& (\mathbf{C}_\delta \cdot \mathbf{G}^{-1}(v \cdot \mathbf{G}) + v_\alpha \cdot \mathbf{G} + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_{v_\gamma},\\
& \mathbf{C}_\delta \cdot \mathbf{G}^{-1}(e' \cdot \mathbf{G}) + e'_\alpha \cdot \mathbf{G} + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_{e'_\gamma},\\
& \mathbf{C}_\delta \cdot \mathbf{G}^{-1}(2e'' \cdot \mathbf{G}) + 2e''_\alpha \cdot \mathbf{G} + \mathbf{C}_\delta \cdot \mathbf{G}^{-1}(\mu \cdot \mathbf{G}) + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_{e''_\gamma}),\\
\mathbf{C}_{\mathbf{r}_0} =& (\mathbf{C}_{[\gamma]_2} - \mathbf{C}_\delta) \cdot \mathbf{G}^{-1}(v \cdot \mathbf{G}) + v_\beta \cdot \mathbf{G} + \mathsf{PK}_{RGSW} \cdot \mathbf{R}v_0,\\
& \mathbf{C}_{[\gamma]_2} - \mathbf{C}_\delta) \cdot \mathbf{G}^{-1}(e' \cdot \mathbf{G}) + e'_\beta \cdot \mathbf{G} + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_{e'_0},\\
& \mathbf{C}_{[\gamma]_2} - \mathbf{C}_\delta) \cdot \mathbf{G}^{-1}(2e'' \cdot \mathbf{G}) + 2e''_\beta \cdot \mathbf{G} + (\mathbf{C}_{[\gamma]_2} - \mathbf{C}_\delta) \cdot \mathbf{G}^{-1}(\mu \cdot \mathbf{G})\\
& + \mathsf{PK}_{RGSW} \cdot \mathbf{R}_{e''_0}).
\end{aligned}
$$

9. Output the proof $\mathbf{\Pi} = (\mathbf{c}_\alpha, \mathbf{c}_\beta, \mathbf{C}_\gamma, \mathbf{C}_{\mathbf{r}_\gamma}, \mathbf{C}_{\mathbf{r}_0})$.

$\mathsf{Verify}(\mathsf{PP}, \mathsf{PK}, \mathsf{SVK}, \mathbf{c}, \mathbf{\Pi} = \{\pi^k\}_{k\in[1,\lceil n/\xi\rceil]})$**:** To verify the proof $\pi$ for $\mathbf{c}$ using the secret verification key $\mathsf{SVK}$, we need to proceed as follows for all $k \in [1, \lceil n/\xi\rceil]$. For the clarity of the description, we only describe one with $\pi$:

1. Parse the secret verification key $\mathsf{SVK} = \mathsf{SSK}_{\mathsf{RGSW}}$,
2. Decrypt and set: $\gamma = \mathsf{RGSW.Decrypt}(\mathsf{SSK}_{\mathsf{RGSW}}, \mathbf{C}_\gamma)$. If $\gamma$ has coefficients other than $0$ or $1$ then abort.
3. Decrypt also: $(v_\gamma, e'_\gamma, \bar{e''_\gamma}) \leftarrow \mathsf{RGSW.Decrypt}(\mathsf{SSK}_{\mathsf{RGSW}}, \mathbf{C}_{\mathbf{r}_\gamma})$,
   and $(v_0, e'_0, \bar{e''_0}) \leftarrow \mathsf{RGSW.Decrypt}(\mathsf{SSK}_{\mathsf{RGSW}}, \mathbf{C}_{\mathbf{r}_0})$,
   then set: $\mathbf{r}_\gamma = (v_\gamma, e'_\gamma, \lfloor \bar{e''_\gamma}/2 \rfloor)$ and $\mathbf{r}_0 = (v_0, e'_0, \lfloor \bar{e''_0}/2 \rfloor)$.

4. Let $B = n \cdot EF(\mathbb{R}_q) \cdot B_\sigma$ and $B' = n \cdot EF(\mathbb{R}_q) \cdot B_{\sigma'} + \frac{n \cdot EF(\mathbb{R}_q)+1}{2}$, we verify that $\mathbf{r}_\gamma = (v_\gamma, e'_\gamma, e''_\gamma)$ and $\mathbf{r}_0 = (v_0, e'_0, e''_0)$ are all small with the following bounds:

$$\begin{aligned} &\|v_\gamma\|_\infty \leq B + B_{\sigma_\alpha}, \quad &&\|e'_\gamma\|_\infty \leq B + B_{\sigma_\alpha}, \quad &&\|e''_\gamma\|_\infty \leq B' + B_{\sigma'_\alpha}, \\ &\|v_0\|_\infty \leq 2 \cdot B + B_{\sigma_\beta}, \quad &&\|e'_0\|_\infty \leq 2 \cdot B + B_{\sigma_\beta}, \quad &&\|e''_0\|_\infty \leq 2 \cdot B' + B_{\sigma'_\beta}. \end{aligned}$$

5. Then, using that $\delta = \mathsf{RGSW.Decrypt}(\mathsf{SSK}_{\mathrm{RGSW}}, \mathbf{C}_\delta)$, verify the following equations:

$$\begin{aligned} \delta \cdot \mathbf{c} + \mathbf{c}_\alpha &= \mathsf{Com.Commit}(\mathsf{ck}, \gamma; \mathbf{r}_\gamma), \\ (\gamma - \delta) \cdot \mathbf{c} + \mathbf{c}_\beta &= \mathsf{Com.Commit}(\mathsf{ck}, 0; \mathbf{r}_0). \end{aligned}$$

6. If all the verification are true then return True otherwise return False.

**Theorem 5.10.** *DVNIZK* $= (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is a Designated-Verifier Non-Interactive Zero-Knowledge argument which ensures culpable soundness with respect to the guilt language* $R_{guilt} = \{((\mathbf{c}, \mathsf{ck}), \mathsf{dk}) | \mathit{KeyVerify}(\mathsf{ck}, \mathsf{dk}) = \mathsf{True} \wedge \mu \notin \{0, 1\} \; where \; \mu = \mathit{Decrypt}(\mathsf{ck}, \mathsf{dk}, \mathbf{c})\}$.

## 5.4. Application to Voting Schemes

In this section, we show how to use our DVNIZK scheme to build a voting scheme. We first recall the definition and the security properties of a voting scheme, then we give our construction of a voting scheme using our DVNIZK argument.

- Our voting scheme (Section 5.4.2) is the first construction of a lattice-based voting scheme in the standard model. We follow directly the techniques described in [CG15] to build this scheme. However, we request a stronger model for the voting scheme as the decryption party can not provide a proof of the tally process. Thus, in this construction, we need to suppose that the decryption party is honest.
- Note that we also propose a second construction in the full version of the paper, where we combine our DVNIZK with an amortized zero-knowledge proof system [CDXY17; PL17] which allows us to run efficiently many proofs in parallel in the random oracle model. This gives us a new lattice-based voting scheme, different from the one described in [PLNS17], as our main building block is in the standard model.

### 5.4.1. Definitions

We start with the formal definition of a voting scheme and its security properties.

**Definition 5.4** (Voting Scheme[CG15])**.** *A voting scheme* $\Pi$ *consists of five PPT algorithms* $\Pi = (\mathit{Setup}, \mathit{Vote}, \mathit{SubmitBallot}, \mathit{CheckBoard}, \mathit{Tally})$ *such that:*

***Setup***$(1^\lambda) \rightarrow (\boldsymbol{VSK}, \boldsymbol{VPK}, \boldsymbol{VVK}, \boldsymbol{BB})$***:*** *The setup algorithm takes the security parameter* $1^\lambda$*, then it produces a secret key VSK, a public key VPK and a verification key VVK, it also initializes a bulletin board BB.*

$\underline{\mathsf{Exp}^{Correctness}_{\pi,\mathcal{A}}(\lambda)}$:
$(\mathsf{VSK}, \mathsf{VPK}, \mathsf{VVK}, \mathsf{BB}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(sum, nb_{malicious}) \leftarrow (0, 0)$
$\mathcal{A}^{\mathcal{O}_{HonestVoter}(\cdot),\mathcal{O}_{MaliciousVoter}(\cdot)}(\mathsf{VPK})$
$\mathsf{BB} \leftarrow \mathsf{CheckBoard}(\mathsf{VPK}, \mathsf{VVK}, \mathsf{BB})$
$result \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{VSK})$
Outputs $(result, sum, nb_{malicious})$

$\underline{\mathcal{O}_{\mathsf{HonestVoter}}(\mu \in \{0,1\})}$:
$\mathsf{B} \leftarrow \mathsf{Vote}(\mathsf{VPK}, \mu)$
$\mathsf{SubmitBallot}(\mathsf{VPK}, \mathsf{B}, \mathsf{BB})$
$sum \leftarrow sum + \mu$
Return B
$\underline{\mathcal{O}_{\mathsf{MaliciousVoter}}(\mathsf{B})}$:
$\mathsf{SubmitBallot}(\mathsf{VPK}, \mathsf{B}, \mathsf{BB})$
$nb_{malicious} \leftarrow nb_{malicious} + 1$

**Figure 5.3.** – Correctness experiment of the voting scheme

***Vote***$(\mathsf{VPK}, \mu \in \{0,1\}) \rightarrow \mathsf{B}$*: The vote algorithm takes the public key VPK, a message* $\mu \in \{0,1\}$ *and outputs a ballot encoding the message* $\mu$*.*

***SubmitBallot***$(\mathsf{VPK}, \mathsf{B}, \mathsf{BB})$*: The ballot submitting algorithm takes a public key VPK, a ballot B and a bulletin board BB, it updates the bulletin board with submitted ballot B.*

***CheckBoard***$(\mathsf{VPK}, \mathsf{VVK}, \mathsf{BB})$*: The bulletin board checking algorithm CheckBoard takes the public key VPK, the verification key VVK and the bulletin board BB, it removes all ballots which can not pass the verification from the bulletin board, then it makes the bulletin board visible.*

***Tally***$(\mathsf{BB}, \mathsf{VSK}) \rightarrow result$*: The tally algorithm takes the bulletin board BB and the secret key VSK, then it reveals the voting result.*

We also request the voting scheme to verify the correctness and the ballot privacy property.

The scheme is correct if every vote submitted by an honest voter is counted correctly and if a malicious voter can not influence the voting result more than behave as an honest one.

**Definition 5.5** (Correctness)**.** *We define an experiment as in Figure 5.3. A voting scheme is correct if, and only if, for all PPT adversary, the following probability is negligible*

$$\Pr[(result, sum, nb_{malicious}) \leftarrow \mathsf{Exp}^{Correctness}_{\pi,\mathcal{A}}(\lambda) : result \leq sum \ \mathsf{OR} \ result \geq sum + nb_{malicious}].$$

The ballot privacy [CG15, Definition 10] property ensures that even if the verification key of the voting scheme becomes public at the same time as the result of the vote, the voters can not get any other information about the messages included in the ballot submitted by other voters.

## 5.4.2. Our Voting Scheme

We build our voting scheme using a $\mathsf{DVNIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ as described in Section 5.3.2 and the related BV encryption/commitment scheme Com = (Com.Setup, Com.Commit, Com.Decrypt) as described in Section 5.2.4.

**Setup**$(1^\lambda)$**:** The decryption party $\mathcal{P}_{\mathsf{Dec}}$ uses Com.Setup to produce the public key $\mathsf{PK}_{\mathsf{Com}}$ and the secret key $\mathsf{SK}_{\mathsf{Com}}$. The verification party $\mathcal{P}_{\mathsf{Verify}}$ uses DVNIZK.Setup to produce the public parameter $\mathsf{PP}_{\mathsf{DVNIZK}}$ and the secret verification key $\mathsf{VK}_{\mathsf{DVNIZK}}$. The setup algorithm initializes a bulletin board BB which is hidden and it outputs: $\mathsf{VSK} = \mathsf{SK}_{\mathsf{Com}}, \mathsf{VPK} = (\mathsf{PK}_{\mathsf{Com}}, \mathsf{PP}_{\mathsf{DVNIZK}}), \mathsf{VVK} = \mathsf{VK}_{\mathsf{DVNIZK}}$.

**Vote**$(\mathbf{VPK}, \mu \in \{0,1\})$**:** Given the public key $\mathsf{VPK} = (\mathsf{PK}_{\mathsf{Com}}, \mathsf{PP}_{\mathsf{DVNIZK}})$, the voter uses $\mathsf{PK}_{\mathsf{Com}}$ to compute an encryption $\mathbf{c}$ of his vote $\mu$ using a random $\mathbf{r}$. He computes also a DVNIZK proof $\pi$ that $\mathbf{c}$ is a well formed commitment of $\mu \in \{0,1\}$ with randomness $\mathbf{r}$. The voter outputs $\mathsf{B} = (\mathbf{c}, \pi)$.

**SubmitBallot**$(\mathbf{VPK}, \mathbf{B}, \mathbf{BB})$**:** This algorithm submits the ballot to the bulletin board.

**CheckBoard**$(\mathbf{VPK}, \mathbf{VVK}, \mathbf{BB})$**:** The verification party $\mathcal{P}_{\mathsf{Verif}}$ uses the verification key VVK to check that all the ballots are commitments of $0$ or $1$. After the tally algorithm finished, we also output VVK to allow everyone to check the bulletin board BB.

**Tally**$(\mathbf{BB}, \mathbf{VSK})$**:** The decryption party $\mathcal{P}_{\mathsf{Dec}}$, using the decryption key VSK, computes the result of the vote $result = \sum_{\mathsf{B}=(\mathbf{c}_B,\pi_B)\in\mathsf{BB}} \mathsf{Decrypt}(\mathsf{SK}_{\mathsf{Com}}, \mathbf{c}_B)$. It outputs $result$ of the vote.

**Theorem 5.11.** *Our voting scheme* $\Pi$ *constructed as above is correct, if the DVNIZK argument is culpable sound.*

*Proof.* We construct an adversary $\mathcal{B}$ against the culpable soundness of the underlying DVNIZK with the adversary $\mathcal{A}$ against the correctness experiment of the voting scheme.

$\mathcal{B}$ simulates the correctness experiment for $\mathcal{A}$ with the $\mathsf{PP}_{\mathsf{DVNIZK}}$ given by the security game of the culpable soundness instead of generate his own $\mathsf{PP}_{\mathsf{DVNIZK}}$ and $\mathsf{VK}_{\mathsf{DVNIZK}}$. The fact that $result \leq sum$ OR $result \geq sum + nb_{malicious}$ implies that at least one of the ballot submitted for $\mathcal{O}_{MaliciousVoter}$ is either a not well formed commitment, either a well formed commitment of $\mu \notin \{0,1\}$. $\mathcal{B}$ uniformly chooses a ballot submitted to the oracle $\mathcal{O}_{MaliciousVoter}$ and submit it together with the decryption key $\mathsf{SK}_{\mathsf{BV}}$ as the witness of the culpable language $w_{guilt}$. Since the number of requests for the oracle $\mathcal{O}_{MaliciousVoter}$ is polynomial, the advantage of $\mathcal{B}$ against the culpable soundness is non-negligible. □ □

**Theorem 5.12.** *Our voting scheme* $\Pi$ *constructed as above verifies the ballot privacy property.*

Theorem 5.12 is based on the zero-knowledge property of the underlying DVNIZK and the IND-CPA security of the underlying BV encryption scheme. As we follow the framework of [CG15], the proof is the same as in [CG15, Theorem 5].

# 6 Conclusion

This thesis presents several efficiency improvement of existing scheme, as well as some new primitives.

## 6.1. Summary of our contributions

In this thesis, we focused on improving the efficiency of existing cryptosystems, as well as providing new functionalities based on various assumptions.

In the first part, we have studied two variants of lossy trapdoor functions. LTFs can been seen as a generalization of trapdoor one-way function. More specifically, the trapdoor one-way function is the starting point of the asymmetric cryptography. However the direct construction of encryption scheme using trapdoor one-way function only satisfies the one-wayness (the adversary cannot easily find the message from the ciphertext). For more advanced indistinguishability based security, the lossy trapdoor function is a more suitable modelization. Using its injectif mode, the receiver can decrypt the ciphertext with the trapdoor. On the other side, in the security proof, we can use the indistinguishability type assumptions to switch the LTF into lossy mode, in which the message is statistically hiding from the adversary's point of view. Therefore the lossy trapdoor functions are essential building blocks for many cryptographic protocols. In this thesis, we studied two variants of lossy trapdoor functions.

Firstly, We proposed the first lossy algebraic filter (LAF) with linear-size tag. For the second variant of the LTF, we also proposed different efficient constructions of $\mathcal{R}$-LTF using different assumptions as the Decision Composite Residuosity assumption (DCR), the Decision Diffie-Hellman assumption (DDH), or the semi-smooth RSA subgroup assumption (ss-RSA). The new $\mathcal{R}$-LTFs can be used to construct more efficient deterministic encryption scheme.

In the second part, we have focused on the zero-knowledge proof systems and their application. Despite importance of zero-knowledge proof in constructing many cryptographic protocols including e-cash, e-voting, CCA (Chosen Ciphertext Attack) encryption schemes, most the constructions of zero-knowledge proof suffered from the using of the random oracle model. The disadvantages is two-fold, on the one hand the random oracle model is plausible. On the other hand, many construction using the random oracle model lacks tight security proof (typically the constructions involving Fiat-Shamir transformation). For these proposes, we studied the zero-knowledge proof in two ways. We try to make some construction tightly secure and try to build zero-knowledge proof systems

without random oracle. By combining the Groth-Kholweiss $\Sigma$-protocol and the dual mode encryption scheme, we achieve the first tightly secure ring-signature scheme. On the other side, we are also interested in the construction of non-interactive zero-knowledge argument system, we proposed a construction of a weaker version: designated-verifier zero-knowledge argument in the standard model and as an application we use our DVNIZK to construct a post-quantum voting scheme in the standard model.

## 6.2. Open Problems

Question 1 Can we build lattice-based KDM-CCA encryption scheme?

In this thesis, we give an efficient construction of lossy algebraic filter (LAF) which allows us to construct an efficient KDM-CCA encryption scheme based on DDH assumption. However, even if our construction of LAF can be easily constructed based on lattices, the framework of the KDM-CCA proposed by [Hof13] is not generic, it can not be constructed using lattice assumptions.

Question 2 Can we construct DVNIZK in the standard model without size of witness loss?

In this thesis, we give a construction of a lattice-based DVNIZK in the standard model. However, there is always a gap between the size of the witness and what the verifier is convinced of. Concretely, in lattice related problems, we need to prove the knowledge of a small vector $x$. In the state-of-the-art of lattice based non-interactive proof system in the standard model, we can only convince the verifier that we know a small vector $y$ such that $y$ is strictly larger than $x$. By using the lattice interactive proofs like Stern's protocol, we can prove the statement, but to prove the statement with an efficient non-interactive proof system without random oracle remains as an open problem.

Question 3 Can we construct a tightly secure logarithmic-size ring signature in the standard model?

In this thesis, we give a construction of a tightly secure logarithmic-size ring signature. Despite the fact that, we bypass the security loss in the Fiat-Shamir transformation used in previous works, we still need a random oracle to construct the primitive. The tightly secure construction of a such primitive in the standard model remains as an open problem.

Question 4 Can we an efficient e-voting scheme in the standard model that profit from the homomorphism of the lattice-based encryption scheme?

Through this thesis, we give an post-quantic e-voting scheme. However, due to the fact that we can not prove the

# Bibliography


[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. *Tightly-Secure Signatures from Lossy Identification Schemes*. In: *EUROCRYPT*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 572–590 (cit. on pp. 58, 59, 61).

[AFQ+14] Diego F. Aranha, Pierre-Alain Fouque, Chen Qian, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. *Binary Elligator Squared*. In: *Selected Areas in Cryptography*. Vol. 8781. Lecture Notes in Computer Science. Springer, 2014, pp. 20–37 (cit. on pp. xii, 7).

[Ajt96] Miklós Ajtai. *Generating Hard Instances of Lattice Problems (Extended Abstract)*. In: *STOC*. ACM, 1996, pp. 99–108 (cit. on p. 78).

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. *1-out-of-n Signatures from a Variety of Keys*. In: *ASIACRYPT*. Vol. 2501. Lecture Notes in Computer Science. Springer, 2002, pp. 415–432 (cit. on p. 60).

[Bab86] László Babai. "On Lovász' lattice reduction and the nearest lattice point problem". In: *Combinatorica* 6.1 (1986), pp. 1–13 (cit. on p. 84).

[Ban93] Wojciech Banaszczyk. "New bounds in some transference theorems in the geometry of numbers". In: *Mathematische Annalen* 296.1 (1993), pp. 625–635 (cit. on p. 81).

[BB04] Dan Boneh and Xavier Boyen. *Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles*. In: *EUROCRYPT*. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 223–238 (cit. on pp. 21, 29).

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. *Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits*. In: *CRYPTO (2)*. Vol. 10992. Lecture Notes in Computer Science. Springer, 2018, pp. 669–699 (cit. on p. 78).

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. *Short Group Signatures*. In: *CRYPTO*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55 (cit. on p. 19).

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. *Short Accountable Ring Signatures Based on DDH*. In: *ESORICS (1)*. Vol. 9326. Lecture Notes in Computer Science. Springer, 2015, pp. 243–265 (cit. on p. 61).

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. *Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures*. In: *ASIACRYPT, Part I*. 2014, pp. 551–572 (cit. on p. 78).

[BDLN16] Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. *How to Prove Knowledge of Small Secrets*. In: *CRYPTO (3)*. Vol. 9816. LNCS. Springer, 2016, pp. 478–498 (cit. on p. 78).

[BDOP16] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris Peikert. "Efficient Commitments and Zero-Knowledge Protocols from Ring-SIS with Applications to Lattice-based Threshold Cryptosystems". In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 997 (cit. on p. 78).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. *Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)*. In: *STOC*. ACM, 1988, pp. 103–112 (cit. on pp. 52, 77).

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. *On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles*. In: *CRYPTO*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 335–359 (cit. on p. 16).

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. *Circular-Secure Encryption from Decision Diffie-Hellman*. In: *CRYPTO*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 108–125 (cit. on p. 22).

[BHJ+13] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Jae Hong Seo, and Christoph Striecks. *Practical Signatures from Standard Assumptions*. In: *EUROCRYPT*. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 461–485 (cit. on p. 28).

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. *Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening*. In: *EUROCRYPT*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 1–35 (cit. on pp. 22, 59, 60).

[BK10] Zvika Brakerski and Yael Tauman Kalai. "A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model". In: *IACR Cryptology ePrint Archive* 2010 (2010), p. 86 (cit. on p. 60).

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. *Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings*. In: *ESORICS*. 2015, pp. 305–325 (cit. on pp. 78, 79, 84).

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. *Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles*. In: *TCC*. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 60–79 (cit. on pp. 60, 61).

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. *(Hierarchical) Identity-Based Encryption from Affine Message Authentication*. In: *CRYPTO (1)*. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 408–425 (cit. on pp. 20, 21, 23, 35, 36).

[BL17] Carsten Baum and Vadim Lyubashevsky. "Simple Amortized Proofs of Shortness for Linear Relations over Polynomial Rings". In: *IACR ePrint Archive* 2017 (2017), p. 759 (cit. on p. 78).

[BLL00] Ahto Buldas, Peeter Laud, and Helger Lipmaa. *Accountable certificate management using undeniable attestations*. In: *ACM Conference on Computer and Communications Security*. ACM, 2000, pp. 9–17 (cit. on p. 60).

[BM93] Josh Cohen Benaloh and Michael de Mare. *One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract)*. In: *EUROCRYPT*. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 274–285 (cit. on p. 60).

[Boy04] Xavier Boyen. *Reusable cryptographic fuzzy extractors*. In: *ACM Conference on Computer and Communications Security*. ACM, 2004, pp. 82–91 (cit. on p. 22).

[Boy07] Xavier Boyen. *Mesh Signatures*. In: *EUROCRYPT*. Vol. 4515. Lecture Notes in Computer Science. Springer, 2007, pp. 210–227 (cit. on p. 60).

[BR93] Mihir Bellare and Phillip Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*. In: *ACM Conference on Computer and Communications Security*. ACM, 1993, pp. 62–73 (cit. on pp. 23, 57).

[BR96] Mihir Bellare and Phillip Rogaway. *The Exact Security of Digital Signatures - HOw to Sign with RSA and Rabin*. In: *EUROCRYPT*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 399–416 (cit. on pp. 23, 61).

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. *Encryption-Scheme Security in the Presence of Key-Dependent Messages*. In: *Selected Areas in Cryptography*. Vol. 2595. Lecture Notes in Computer Science. Springer, 2002, pp. 62–75 (cit. on p. 19).

[BS11] Zvika Brakerski and Gil Segev. *Better Security for Deterministic Public-Key Encryption: The Auxiliary-Input Setting*. In: *CRYPTO*. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 543–560 (cit. on p. 16).

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. *Threshold Ring Signatures and Applications to Ad-hoc Groups*. In: *CRYPTO*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 465–480 (cit. on p. 60).

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. *Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages*. In: *CRYPTO 2011*. Vol. 6841. LNCS. Springer, 2011, pp. 505–524 (cit. on pp. 78, 82).

[BW10] Xavier Boyen and Brent Waters. *Shrinking the Keys of Discrete-Log-Type Lossy Trapdoor Functions*. In: *ACNS*. Vol. 6123. Lecture Notes in Computer Science. 2010, pp. 35–52 (cit. on pp. 20, 21, 25).

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*. In: *CRYPTO*. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 174–187 (cit. on pp. 60, 63).

[CDXY17] Ronald Cramer, Ivan Damgård, Chaoping Xing, and Chen Yuan. *Amortized Complexity of Zero-Knowledge Proofs Revisited: Achieving Linear Soundness Slack*. In: *EUROCRYPT (1)*. Vol. 10210. Lecture Notes in Computer Science. 2017, pp. 479–500 (cit. on p. 91).

[CG15] Pyrros Chaidos and Jens Groth. *Making Sigma-Protocols Non-interactive Without Random Oracles*. In: *PKC*. Vol. 9020. LNCS. Springer, 2015, pp. 650–670 (cit. on pp. xi, 7, 77, 79, 80, 85, 87, 91–93).

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. *A Homomorphic LWE Based E-voting Scheme*. In: *PQCrypto*. Vol. 9606. Lecture Notes in Computer Science. Springer, 2016, pp. 245–265 (cit. on p. 80).

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. *Ring Signatures of Sub-linear Size Without Random Oracles*. In: *ICALP*. Vol. 4596. Lecture Notes in Computer Science. Springer, 2007, pp. 423–434 (cit. on p. 60).

[CH91] David Chaum and Eugène van Heyst. *Group Signatures*. In: *EUROCRYPT*. Vol. 547. Lecture Notes in Computer Science. Springer, 1991, pp. 257–265 (cit. on pp. 52, 57).

[CHKO08] Philippe Camacho, Alejandro Hevia, Marcos A. Kiwi, and Roberto Opazo. *Strong Accumulators from Collision-Resistant Hashing*. In: *ISC*. Vol. 5222. Lecture Notes in Computer Science. Springer, 2008, pp. 471–486 (cit. on p. 60).

[CL06] Melissa Chase and Anna Lysyanskaya. *On Signatures of Knowledge*. In: *CRYPTO*. Vol. 4117. Lecture Notes in Computer Science. Springer, 2006, pp. 78–96 (cit. on p. 60).

[Cor00] Jean-Sébastien Coron. *On the Exact Security of Full Domain Hash*. In: *CRYPTO*. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 229–235 (cit. on p. 61).

[Cra96] Ronald Cramer. *Modular Design of Secure, yet Practical Cryptographic Protocols*. In: *Doctoral thesis*. 1996 (cit. on p. 11).

[CW13] Jie Chen and Hoeteck Wee. *Fully, (Almost) Tightly Secure IBE and Dual System Groups*. In: *CRYPTO (2)*. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 435–460 (cit. on pp. 20, 23).

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. *Non-interactive Zero-Knowledge from Homomorphic Encryption*. In: *TCC*. Vol. 3876. LNCS. Springer, 2006, pp. 41–59 (cit. on p. 77).

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. *Public-Key Encryption Schemes with Auxiliary Inputs*. In: *TCC*. Vol. 5978. LNCS. Springer, 2010, pp. 361–381 (cit. on p. 82).

[DH76] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Trans. Information Theory* 22.6 (1976), pp. 644–654. Link. (Cit. on pp. viii, 4, 12).

[DJ01] Ivan Damgård and Mads Jurik. *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*. In: *Public Key Cryptography*. Vol. 1992. Lecture Notes in Computer Science. Springer, 2001, pp. 119–136 (cit. on p. 52).

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. *Anonymous Identification in Ad Hoc Groups*. In: *EUROCRYPT*. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 609–626 (cit. on p. 60).

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. *Magic Functions*. In: *FOCS*. IEEE Computer Society, 1999, pp. 523–534 (cit. on p. 22).

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*. In: *EUROCRYPT*. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 523–540 (cit. on p. 22).

[FGK+10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. *More Constructions of Lossy and Correlation-Secure Trapdoor Functions*. In: *Public Key Cryptography*. Vol. 6056. Lecture Notes in Computer Science. Springer, 2010, pp. 279–295 (cit. on p. 16).

[FQ16] Pierre-Alain Fouque and Chen Qian. *Fault Attacks on Efficient Pairing Implementations*. In: *AsiaCCS*. ACM, 2016, pp. 641–650 (cit. on pp. xii, 7).

[FS86] Amos Fiat and Adi Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*. In: *CRYPTO*. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194 (cit. on pp. 61, 64).

[Fuj14] Eiichiro Fujisaki. *All-But-Many Encryption - A New Framework for Fully-Equipped UC Commitments*. In: *ASIACRYPT (2)*. Vol. 8874. Lecture Notes in Computer Science. Springer, 2014, pp. 426–447 (cit. on p. 22).

[Gen09] Craig Gentry. *Fully homomorphic encryption using ideal lattices*. In: *STOC*. ACM, 2009, pp. 169–178 (cit. on pp. vii, 3, 78).

[GHK17] Romain Gay, Dennis Hofheinz, and Lisa Kohl. *Kurosawa-Desmedt Meets Tight Security*. In: *CRYPTO (3)*. Vol. 10403. Lecture Notes in Computer Science. Springer, 2017, pp. 133–160 (cit. on p. 23).

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. *Tightly CCA-Secure Encryption Without Pairings*. In: *EUROCRYPT (1)*. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 1–27 (cit. on pp. 23, 77).

[GJ03] Eu-Jin Goh and Stanislaw Jarecki. *A Signature Scheme as Secure as the Diffie-Hellman Problem*. In: *EUROCRYPT*. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 401–415 (cit. on p. 61).

[GK15] Jens Groth and Markulf Kohlweiss. *One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin*. In: *EUROCRYPT (2)*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 253–280 (cit. on pp. xi, 6, 10, 57–65, 74).

[Gon17] Alonso González. "A Ring Signature of size Θ(sqrt[3]{n}) without Random Oracles". In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 905 (cit. on p. 60).

[GS08] Jens Groth and Amit Sahai. *Efficient Non-interactive Proof Systems for Bilinear Groups*. In: *EUROCRYPT*. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 415–432 (cit. on p. 59).

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based*. In: *CRYPTO (1)*. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 75–92 (cit. on pp. 78, 80, 84).

[HJ12] Dennis Hofheinz and Tibor Jager. *Tightly Secure Signatures and Public-Key Encryption*. In: *CRYPTO*. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 590–607 (cit. on p. 23).

[HK08] Dennis Hofheinz and Eike Kiltz. *Programmable Hash Functions and Their Applications*. In: *CRYPTO*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 21–38 (cit. on p. 33).

[HN19] Dennis Hofheinz and Ngoc Khanh Nguyen. *On Tightly Secure Primitives in the Multi-instance Setting*. In: *Public Key Cryptography (1)*. Vol. 11442. Lecture Notes in Computer Science. Springer, 2019, pp. 581–611 (cit. on p. 23).

[HO12] Brett Hemenway and Rafail Ostrovsky. *Extended-DDH and Lossy Trapdoor Functions*. In: *Public Key Cryptography*. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 627–643 (cit. on p. 16).

[Hof12] Dennis Hofheinz. *All-But-Many Lossy Trapdoor Functions*. In: *EUROCRYPT*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 209–227 (cit. on pp. 16, 22, 23, 34).

[Hof13] Dennis Hofheinz. *Circular Chosen-Ciphertext Security with Compact Ciphertexts*. In: *EUROCRYPT*. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 520–536 (cit. on pp. x, 6, 19, 20, 22–25, 96).

[Hof16] Dennis Hofheinz. *Algebraic Partitioning: Fully Compact and (almost) Tightly Secure Cryptography*. In: *TCC (A1)*. Vol. 9562. Lecture Notes in Computer Science. Springer, 2016, pp. 251–281 (cit. on p. 23).

[Hof17] Dennis Hofheinz. *Adaptive Partitioning*. In: *EUROCRYPT (3)*. Vol. 10212. Lecture Notes in Computer Science. 2017, pp. 489–518 (cit. on p. 23).

[JR13] Charanjit S. Jutla and Arnab Roy. *Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces*. In: *ASIACRYPT (1)*. Vol. 8269. Lecture Notes in Computer Science. Springer, 2013, pp. 1–20 (cit. on p. 28).

[KGV16] Alhassan Khedr, P. Glenn Gulak, and Vinod Vaikuntanathan. "SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers". In: *IEEE Trans. Computers* 65.9 (2016), pp. 2848–2858 (cit. on p. 84).

[KK12] Saqib A. Kakvi and Eike Kiltz. *Optimal Security Proofs for Full Domain Hash, Revisited*. In: *EUROCRYPT*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 537–553 (cit. on p. 61).

[KP06] Sébastien Kunz-Jacques and David Pointcheval. *About the Security of MTI/C0 and MQV*. In: *SCN*. Vol. 4116. Lecture Notes in Computer Science. Springer, 2006, pp. 156–172 (cit. on pp. 12, 13, 21).

[KR00] Hugo Krawczyk and Tal Rabin. *Chameleon Signatures*. In: *NDSS*. The Internet Society, 2000 (cit. on p. 10).

[KW03] Jonathan Katz and Nan Wang. *Efficiency improvements for signature schemes with tight security reductions*. In: *ACM Conference on Computer and Communications Security*. ACM, 2003, pp. 155–164 (cit. on pp. 58, 59, 61).

[LJYP14] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. *Concise Multi-challenge CCA-Secure Encryption and Signatures with Almost Tight Security*. In: *ASIACRYPT (2)*. Vol. 8874. Lecture Notes in Computer Science. Springer, 2014, pp. 1–21 (cit. on p. 23).

[LLNW16] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. *Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors*. In: *EUROCRYPT (2)*. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 1–31 (cit. on pp. xi, 6, 57, 60, 61).

[LM06] Vadim Lyubashevsky and Daniele Micciancio. *Generalized Compact Knapsacks Are Collision Resistant*. In: *ICALP (2)*. 2006, pp. 144–155 (cit. on p. 81).

[LN17] Vadim Lyubashevsky and Gregory Neven. *One-Shot Verifiable Encryption from Lattices*. In: *EUROCRYPT (1)*. 2017, pp. 293–323 (cit. on p. 78).

[LN86] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and Their Applications*. NY, USA: Cambridge University Press, 1986. ISBN: 0-521-30706-6 (cit. on p. 81).

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. *Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications*. In: *PKC*. 2013, pp. 107–124 (cit. on p. 78).

[LPJY15] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. *Compactly Hiding Linear Spans - Tightly Secure Constant-Size Simulation-Sound QA-NIZK Proofs and Applications*. In: *ASIACRYPT (1)*. Vol. 9452. Lecture Notes in Computer Science. Springer, 2015, pp. 681–707 (cit. on p. 23).

[LPQ17] Benoît Libert, Thomas Peters, and Chen Qian. *Structure-Preserving Chosen-Ciphertext Security with Shorter Verifiable Ciphertexts*. In: *Public Key Cryptography (1)*. Vol. 10174. Lecture Notes in Computer Science. Springer, 2017, pp. 247–276 (cit. on pp. xii, 7).

[LPQ18] Benoît Libert, Thomas Peters, and Chen Qian. *Logarithmic-Size Ring Signatures with Tight Security from the DDH Assumption*. In: *ESORICS (2)*. Vol. 11099. Lecture Notes in Computer Science. Springer, 2018, pp. 288–308 (cit. on pp. xi, 7, 52).

[LPR13a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *A Toolkit for Ring-LWE Cryptography*. In: *EUROCRYPT*. Vol. 7881. LNCS. Springer, 2013, pp. 35–54 (cit. on p. 82).

[LPR13b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *J. ACM* 60.6 (2013), 43:1–43:35 (cit. on pp. 79, 82).

[LQ19] Benoît Libert and Chen Qian. *Lossy Algebraic Filters with Short Tags*. In: *Public Key Cryptography (1)*. Vol. 11442. Lecture Notes in Computer Science. Springer, 2019, pp. 34–65 (cit. on pp. xi, 7).

[LS17] Vadim Lyubashevsky and Gregor Seiler. "Partially Splitting Rings for Faster Lattice-Based Zero-Knowledge Proofs". In: *IACR ePrint* 2017 (2017), p. 523 (cit. on p. 78).

[LSSS17] Benoît Libert, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. *All-But-Many Lossy Trapdoor Functions and Selective Opening Chosen-Ciphertext Security from LWE*. In: *CRYPTO (3)*. Vol. 10403. Lecture Notes in Computer Science. Springer, 2017, pp. 332–364 (cit. on pp. 23, 34).

[LSW10] Allison B. Lewko, Amit Sahai, and Brent Waters. *Revocation Systems with Very Small Private Keys*. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2010, pp. 273–285 (cit. on pp. 21, 25).

[LV08] Benoît Libert and Damien Vergnaud. *Multi-use unidirectional proxy re-signatures*. In: *ACM Conference on Computer and Communications Security*. ACM, 2008, pp. 511–520 (cit. on p. 13).

[MOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN: 0-8493-8523-7 (cit. on pp. vii, 3).

[MP12] Daniele Micciancio and Chris Peikert. *Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller*. In: *EUROCRYPT*. Vol. 7237. LNCS. Springer, 2012, pp. 700–718 (cit. on pp. 80, 84).

[MV03] Daniele Micciancio and Salil P. Vadhan. *Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More*. In: *CRYPTO*. 2003, pp. 282–298 (cit. on p. 78).

[NR97] Moni Naor and Omer Reingold. *Number-theoretic Constructions of Efficient Pseudo-random Functions*. In: *FOCS*. IEEE Computer Society, 1997, pp. 458–467 (cit. on pp. 14, 35, 46).

[Oka92] Tatsuaki Okamoto. *Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes*. In: *CRYPTO*. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 31–53 (cit. on p. 60).

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. *A New Public-Key Cryptosystem as Secure as Factoring*. In: *EUROCRYPT*. Vol. 1403. LNCS. Springer, 1998, pp. 308–318 (cit. on p. 77).

[Pai99] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. In: *EUROCRYPT*. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238 (cit. on pp. viii, 4).

[PL17] Rafaël del Pino and Vadim Lyubashevsky. *Amortization with Fewer Equations for Proving Knowledge of Small Secrets*. In: *CRYPTO*. 2017, pp. 365–394 (cit. on pp. 78, 91).

[PLNS17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. *Practical Quantum-Safe Voting from Lattices*. In: *CCS*. ACM, 2017, pp. 1565–1581 (cit. on pp. 80, 91).

[PR06] Chris Peikert and Alon Rosen. *Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices*. In: *TCC*. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 145–166 (cit. on p. 82).

[PS96] David Pointcheval and Jacques Stern. *Security Proofs for Signature Schemes*. In: *EUROCRYPT*. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 387–398 (cit. on pp. 58, 61, 78).

[PV05] Pascal Paillier and Damien Vergnaud. *Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log*. In: *ASIACRYPT*. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 1–20 (cit. on p. 61).

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. *A Framework for Efficient and Composable Oblivious Transfer*. In: *CRYPTO*. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 554–571 (cit. on p. 59).

[PW08] Chris Peikert and Brent Waters. *Lossy trapdoor functions and their applications*. In: *STOC*. ACM, 2008, pp. 187–196 (cit. on pp. ix, 5, 9, 16, 22).

[QTG18] Chen Qian, Mehdi Tibouchi, and Rémi Géraud. *Universal Witness Signatures*. In: *IWSEC*. Vol. 11049. Lecture Notes in Computer Science. Springer, 2018, pp. 313–329 (cit. on pp. xi, 7).

[Reg05] Oded Regev. *On lattices, learning with errors, random linear codes, and cryptography*. In: *STOC*. ACM, 2005, pp. 84–93 (cit. on pp. viii, 4).

[Reg09] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *J. ACM* 56.6 (2009), 34:1–34:40 (cit. on p. 78).

[RS91] Charles Rackoff and Daniel R. Simon. *Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*. In: *CRYPTO*. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 433–444 (cit. on p. 22).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (1978), pp. 120–126 (cit. on pp. viii, 4).

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. *How to Leak a Secret*. In: *ASIACRYPT*. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 552–565 (cit. on pp. xi, 6, 52, 57, 60).

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. *Deterministic Public-Key Encryption for Adaptively Chosen Plaintext Distributions*. In: *EUROCRYPT*. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 93–110 (cit. on p. 16).

[RSW18] Miruna Rosca, Damien Stehlé, and Alexandre Wallet. *On the Ring-LWE and Polynomial-LWE Problems*. In: *EUROCRYPT (1)*. Vol. 10820. Lecture Notes in Computer Science. Springer, 2018, pp. 146–173 (cit. on p. 82).

[San99] Tomas Sander. *Efficient Accumulators without Trapdoor Extended Abstracts*. In: *ICICS*. Vol. 1726. Lecture Notes in Computer Science. Springer, 1999, pp. 252–262 (cit. on p. 60).

[Sho94] Peter W. Shor. *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. In: *FOCS*. IEEE Computer Society, 1994, pp. 124–134 (cit. on pp. viii, 4).

[SW07] Hovav Shacham and Brent Waters. *Efficient Ring Signatures Without Random Oracles*. In: *Public Key Cryptography*. Vol. 4450. Lecture Notes in Computer Science. Springer, 2007, pp. 166–180 (cit. on p. 60).

[Wat05] Brent Waters. *Efficient Identity-Based Encryption Without Random Oracles*. In: *EUROCRYPT*. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 114–127 (cit. on pp. 20, 21, 28).

[Wee12] Hoeteck Wee. *Dual Projective Hashing and Its Applications - Lossy Trapdoor Functions and More*. In: *EUROCRYPT*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 246–262 (cit. on p. 16).

[WL18] Yunhua Wen and Shengli Liu. *Robustly Reusable Fuzzy Extractor from Standard Assumptions*. In: *ASIACRYPT (3)*. Vol. 11274. Lecture Notes in Computer Science. Springer, 2018, pp. 459–489 (cit. on pp. x, 6, 19, 20, 22–24).

[Zha16] Mark Zhandry. *The Magic of ELFs*. In: *CRYPTO (1)*. Vol. 9814. Lecture Notes in Computer Science. Springer, 2016, pp. 479–508 (cit. on p. 16).

# List of Figures

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

**Titre :** Les primitives *lossy trapdoor*, preuve à divulgation nulle de connaissance et applications.

**Mot clés :** *Lossy Trapdoor Function*, preuve à divulgation nulle de connaissance, preuve de sécurité.

**Résumé :** Dans cette thèse, nous étudions deux primitives différentes : les *lossy trapdoor functions* (LTF) et les systèmes de preuve à divulgation nulle de connaissance.

Les LTFs sont des familles de fonctions dans lesquelles les fonctions injectives et les fonctions *lossy* sont calculatoirement indistinguables. Depuis leur introduction, elles se sont avérées utiles pour la construction de diverses primitives cryptographiques. Nous donnons dans cette thèse des constructions efficaces d'une variante de la LTF : le filtre algébrique *lossy*. Avec cette variante, nous pouvons améliorer l'efficacité du schéma de chiffrement KDM-CCA et extracteur flous.

Dans la deuxième partie de cette thèse, nous étudions les constructions de systèmes de preuve à divulgation nullle de connaissance. Nous donnons la première signature d'anneau de taille logarithmique avec la sécurité étroite en utilisant une variante de Groth-Kolhweiz $\Sigma$-protocole dans le modèle de l'oracle aléatoire. Nous proposons également une nouvelle construction d'arguments à divulgation nulle de connaissance non-intéractive et à vérifieur désigné (DVNIZK) sous l'hypothèse de réseaux Euclidiens. En utilisant cette nouvelle construction, nous construisons un système de vote basé sur les réseaux Euclidiens dans le modèle standard.

**Title:** Lossy trapdoor primitives, zero-knowledge proofs and applications.

**Keywords:** Lossy trapdoor function, zero-knowledge proof, security proof.

**Abstract:** In this thesis, we study two different primitives: lossy trapdoor functions and zero-knwoledge proof systems.

The lossy trapdoor functions (LTFs) are function families in which injective functions and lossy ones are computationally indistinguishable. Since their introduction, they have been found useful in constructing various cryptographic primitives. We give in this thesis efficient constructions of a variant of LTF: Lossy Algebraic Filter. Using this variant, we can improve the efficiency of the KDM-CCA (Key-Depended-Message Chosen-Ciphertext-Attack) encryption schemes and fuzzy extractors.

In the second part of this thesis, we investigate on constructions of zero-knowledge proof systems. We give the first logarithmic-size ring-signature with tight security using a variant of Groth-Kolhweiz $\Sigma$-protocol in the random oracle model. We also propose one new construction of lattice-based Designated-Verifier Non-Interactive Zero-Knowledge arguments (DVNIZK). Using this new construction, we build a lattice-based voting scheme in the standard model.