HAL
open science

Functional encryption for inner-product evaluations

Florian Bourse

To cite this version:

Florian Bourse. Functional encryption for inner-product evaluations. Cryptography and Security [cs.CR]. Université Paris sciences et lettres, 2017. English. NNT : 2017PSLEE067 . tel-01665276v2

HAL Id: tel-01665276

https://theses.hal.science/tel-01665276v2

Submitted on 17 Jul 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'École normale supérieure

Functional Encryption for Inner-Product Evaluations

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité Informatique

COMPOSITION DU JURY

M. FERREIRA ABDALLA Michel
CNRS, École normale supérieure
Directeur de thèse

M. CATALANO DARIO
University of Catania
Président du jury

M. GENNARO Rosario
City University of New York
Rapporteur

Mme ROUX-LANGLOIS Adeline
CNRS, Institut de recherche en informatique et systèmes aléatoires
Examinatrice

M. LIBERT Benoit
CNRS, École normale supérieure de Lyon
Rapporteur

M. POINTCHEVAL David
CNRS, École normale supérieure
Directeur de thèse

Soutenue par
Florian Bourse
le 13 décembre 2017

Dirigée par
Michel FERREIRA ABDALLA
et **David POINTCHEVAL**

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

Functional Encryption for Inner-Product Evaluations

Florian Bourse

Thèse de doctorat dirigée par
Michel Ferreira Abdalla et David Pointcheval

Résumé

Le chiffrement fonctionnel est une technique émergente en cryptographie dans laquelle une autorité toute puissante est capable de distribuer des clés permettant d'effectuer des calculs sur des données chiffrées de manière contrôlée. La mode dans ce domaine est de construire des schémas qui sont aussi expressifs que possible, c'est-à-dire du chiffrement fonctionnel qui permet l'évaluation de n'importe quel circuit. Ces contributions délaissent souvent l'efficacité ainsi que la sécurité. Elles reposent sur des hypothèses fortes, très peu étudiées, et aucune construction n'est proche d'être pratique.
Le but de cette thèse est d'attaquer ce défi sous un autre angle: nous essayons de construire des schémas de chiffrement fonctionnel les plus expressifs que nous le pouvons en se basant sur des hypothèses classiques, tout en conservant la simplicité et l'efficacité des constructions. C'est pourquoi nous introduisons la notion de chiffrement fonctionnel pour l'évaluation de produits scalaires, où les messages sont des vecteurs $\vec{x}$, et l'autorité peut transmettre des clés correspondants à des vecteurs $\vec{y}$ qui permettent l'évaluation du produit scalaire $\langle \vec{x}, \vec{y} \rangle$. Cette fonctionnalité possède immédiatement des applications directes, et peut aussi être utilisée dans d'autres constructions plus théoriques, le produit scalaire étant une opération couramment utilisée.
Enfin, nous présentons deux structures génériques pour construire des schémas de chiffrement fonctionnels pour le produit scalaire, ainsi que des instanciations concrètes dont la sécurité repose sur des hypothèses classiques, telles que DDH, DCR, et LWE. Nous comparons aussi les avantages et inconvénients de chacune d'entre elles.

Abstract

Functional encryption is an emerging framework in which a master authority can distribute keys that allow some computation over encrypted data in a controlled manner. The trend on this topic is to try to build schemes that are as expressive as possible, i.e., functional encryption that supports any circuit evaluation. These results are at the cost of efficiency and security. They rely on recent, not very well studied assumptions, and no construction is close to being practical.
The goal of this thesis is to tackle this challenge from a different angle: we try to build the most expressive functional encryption scheme we can get from standard assumptions, while keeping the constructions simple and efficient.
To this end, we introduce the notion of functional encryption for inner-product evaluations, where plaintexts are vectors $\vec{x}$, and the trusted authority delivers keys for vectors $\vec{y}$ that allow the evaluation of the inner-product $\langle \vec{x}, \vec{y} \rangle$. This functionality already offers some direct applications, and it can also be used for theoretical constructions, as inner-product is a widely used operation.
Finally, we present two generic frameworks to construct inner-product functional encryption schemes, as well as some concrete instantiations whose security relies on standard assumptions such as DDH, DCR, and LWE. We also compare their pros and cons.

Remerciements

<('-'<) ~^('-')^ ~(>'-')>

"I don't know half of you half as well as I should like; and I like less than half of you half as well as you deserve."

— Bilbo Baggins, *The Lord of the Rings*

Je tiens d'abord à m'excuser auprès de tous les non-francophones qui liront cette thèse pour ces remerciements que j'écris en français. J'ai choisi de réserver mes remerciements pour toutes les personnes qui ont pu rendre ces travaux possibles. C'est pourquoi je souhaiterais avant de les partager, exprimer une pensée pour toutes les personnes que j'ai choisi de ne pas inclure dans cette partie. Une pensée pour toutes les personnes qui ont fait de moi qui je suis, qui m'ont appris, avec qui j'ai discuté, partagé un bon moment, réfléchi, ou philosophé : toutes celles et ceux que je considère mes amis (virtuels ou non), passés, présents, et futurs, et qui j'espère se reconnaîtront ; les professeurs et enseignants de toute ma scolarité qui s'achève par cet ouvrage ; les élèves qui ont dû me supporter comme enseignant à l'université ; toutes les personnes que j'ai pu affronter lors de joutes orales (ou écrites, sur internet) ou de débats — lors d'un débat, il n'y a pas de perdant, tous les participants en ressortent plus éclairés ; enfin, comme la productivité est toujours liée à la bonne humeur, tous ces inconnus qui, par un sourire ou un simple bonjour, ont pu ensoleiller ma journée, ou tout du moins mon trajet. Une pensée aussi à tous les auteurs des livres que j'ai lus, tous les compositeurs des musiques et chansons que j'ai entendus, tous les réalisateurs et acteurs des films que j'ai vus, tous les vidéastes qui ont pu me distraire ou m'instruire en ligne — et tout particulièrement pannenkoek, pour son modèle de rigueur scientifique.

Mes premiers remerciements vont bien évidemment à mes deux directeurs de thèse, qui ont su être disponibles pour moi tout au long de ces dernières années, tout en me laissant l'autonomie nécessaire afin d'apprendre à apprendre seul. Ils ont certainement été de bien meilleurs professeurs que je n'ai été un élève, et je me vois grandi par leur enseignement.

Ensuite, je me dois de remercier mes rapporteurs, qui ont accepté de relire cet ouvrage et qui m'ont permis d'en améliorer la qualité jusqu'au dernier moment. Les finitions de cette thèse n'ont été possibles que grâce à vous. De manière plus générale, j'aimerais remercier tous les reviewers qui ont relu mes travaux, et qui m'ont apporté des commentaires constructifs, qui m'ont permis de mieux comprendre, ou de mieux expliquer dans mes papiers de recherche. Et, encore plus généralement, j'aimerais remercier tous les lecteurs de cette thèse, et des mes travaux de recherche, sans qui ceux-ci auraient été futiles.

J'aimerais aussi remercier tous mes co-auteurs. J'ai pris beaucoup de plaisir à travailler avec vous et j'apprécie énormément les résultats de nos recherches.

La plus grande partie de mon temps passé à travailler sur cette thèse a été passée au laboratoire de crypto de l'ENS. C'est pourquoi je tiens particulièrement à remercier toutes les personnes qui y ont vécu avec moi. Venir à l'ENS a toujours été pour moi un plaisir et j'y ai rencontré des personnes formidables, attachantes, et intelligentes, avec qui j'ai partagé des

moments de joie et de réflexion que je n'oublierai pas. Sachez que vous resterez tous gravés dans ma mémoire. Je remercie toutes les personnes qui m'ont remercié, et toutes celles qui me remercieront. Je remercie aussi de la même manière les membres de CryptoExperts, qui m'ont invité à travailler avec eux, et à qui j'ai eu la joie de rendre visite plusieurs fois ces derniers mois.

Mes travaux ont d'autre part été beaucoup facilités par l'efficacité du personnel administratif de l'ENS et du CNRS, efficacité qui m'a permis de me consacrer entièrement à mon travail de recherche et à l'enseignement. Je remercie aussi le Service de Prestations Informatiques de l'ENS et leur compétence.

J'arrive à la partie de mes remerciements dédiée à mes proches. Je remercie donc du fond du cœur tous mes amis, ainsi que ma famille. Je remercie mes beaux-parents de m'avoir accueilli à bras ouverts. Je remercie mes parents pour m'avoir appris tout ce qu'ils m'ont appris, et sans qui je ne saurais toujours rien. Je remercie mes frères. Chaque moment passé avec vous est une joie pour moi, et tous les instants passés avec vous sont inévitablement trop courts.

Enfin, je termine mes remerciements avec le plus important. Je remercie l'élue de mon cœur, qui partageait déjà sa vie avec moi avant le commencement de cette thèse, qui a continué pendant toute sa durée, et qui j'espère, la partagera encore longtemps avec moi. Aucun mot de mon vocabulaire n'est assez puissant pour exprimer ce que je ressens pour toi, mais je continuerai d'essayer : je t'aime.

Contents

Résumé iii

Abstract v

Remerciements vii

1 Introduction 1
- 1.1 Personal Contributions . . . 3
  - 1.1.1 Contributions in this Thesis . . . 3
  - 1.1.2 Other Contributions . . . 4
- 1.2 Organization of this Thesis . . . 4
- Personal Publications . . . 5

2 Preliminaries 7
- 2.1 Notation and Preliminaries . . . 8
  - 2.1.1 Mathematical Notions . . . 8
  - 2.1.2 Provable Security . . . 10
- 2.2 Basic Cryptographic Primitives . . . 10
  - 2.2.1 One-Time Signature and Collision Resistant Hash Functions . . . 10
  - 2.2.2 Public Key Encryption . . . 11
- 2.3 Projective Hash Functions . . . 12
  - 2.3.1 Subset Membership Problems . . . 13
  - 2.3.2 Projective Hash Functions . . . 13
- 2.4 Additional Preliminaries for Concrete Instantiations . . . 14
  - 2.4.1 Basic Number Theory . . . 14
  - 2.4.2 Gaussians and Lattices . . . 14

3 Inner-Product Functional Encryption 17
- 3.1 Definition and Security Model . . . 18
  - 3.1.1 Definition . . . 18
  - 3.1.2 Indistinguishability-Based Security . . . 19
- 3.2 Alternative Security Models . . . 22
  - 3.2.1 Selectivity vs. Adaptivity . . . 22
  - 3.2.2 Simulation-Based Security . . . 25
- 3.3 Indistinguishability under Chosen-Ciphertext Attack . . . 27
  - 3.3.1 Tag-Based Inner-Product Functional Encryption . . . 28
  - 3.3.2 Removing the Tag . . . 30
- 3.4 Function Hiding . . . 31
  - 3.4.1 Function Hiding in the Public Key Setting . . . 32
  - 3.4.2 Secret Key Inner-Product Functional Encryption . . . 33
  - 3.4.3 Partial Function Hiding . . . 34

4 Generic Construction from Public Key Encryption 41
4.1 Overview . . . 42
4.2 Additional Properties of Public Key Encryption . . . 43
4.2.1 Structural Properties . . . 43
4.2.2 Homomorphic Properties . . . 44
4.2.3 Security Properties . . . 45
4.3 Generic Selectively Secure Inner-Product Functional Encryption . . . 46
4.3.1 Construction and Correctness . . . 46
4.3.2 A Simpler Case: Randomness Reuse . . . 47
4.3.3 Security in the General Case . . . 51
4.4 Generic Adaptively Secure Inner-Product Functional Encryption . . . 56
4.4.1 Construction and Correctness . . . 56
4.4.2 Adaptive Security . . . 58

5 Generic Construction from Projective Hash Functions 63
5.1 Overview . . . 64
5.2 IPFE-CPA Friendly Projective Hash Functions . . . 67
5.2.1 Key Homomorphism . . . 67
5.2.2 Projection Key Homomorphism . . . 68
5.2.3 Strong Diversity . . . 68
5.2.4 Translation Indistinguishability . . . 69
5.2.5 IPFE-CPA Friendliness . . . 69
5.3 Inner-Product Functional Encryption Secure Against Chosen-Plaintext Attacks 69
5.3.1 Construction and Correctness . . . 70
5.3.2 Security Against Chosen-Plaintext Attacks . . . 72
5.4 IPFE-CCA Friendly Projective Hash Functions . . . 74
5.4.1 Tag-Based Projective Hash Function . . . 75
5.4.2 2-Universality . . . 76
5.4.3 Universal Translation Indistinguishability . . . 76
5.4.4 IPFE-CCA Friendliness . . . 77
5.5 Inner-Product Functional Encryption Secure Against Chosen-Ciphertext Attacks 77
5.5.1 Construction and Correctness . . . 78
5.5.2 Security Against Chosen-Ciphertext Attacks . . . 80

6 Instantiations based on the Decisional Diffie-Hellman Assumption 85
6.1 The Decisional Diffie Hellman Assumption and Subset Membership Problem 86
6.1.1 The Decisional Diffie Hellman Assumption . . . 86
6.1.2 The Matrix Decisional Diffie-Hellman Assumption . . . 86
6.2 ElGamal Public Key Encryption Scheme . . . 87
6.2.1 Additively Homomorphic ElGamal . . . 87
6.2.2 Additional Properties of the Additive ElGamal Public Key Encryption Scheme . . . 88
6.3 Projective Hash Functions Based on the Decisional Diffie-Hellman Assumptions 88
6.3.1 IPFE-CPA-Friendly Projective Hash Functions . . . 88
6.3.2 IPFE-CCA-Friendly Projective Hash Functions . . . 90

6.4 Inner-Product Functional Encryption Schemes Based on the Decisional Diffie-Hellman Assumption . . . 92
6.4.1 Scheme Secure Against Selective Chosen-Plaintext Attacks . . . 92
6.4.2 Scheme Secure Against Adaptive Chosen-Plaintext Attacks . . . 94
6.4.3 Scheme Secure Against Chosen-Ciphertext Attacks . . . 95

7 Instantiations based on the Decisional Composite Residuosity Assumption 97
7.1 The Decisional Composite Residuosity Assumption and Subset Membership Problem . . . 98
7.1.1 The Decisional Composite Residuosity Assumption . . . 98
7.1.2 Subset Membership Problem Based on the Decisional Composite Residuosity Assumption . . . 98
7.2 Projective Hash Functions Based on the Decisional Composite Residuosity Assumptions . . . 100
7.2.1 IPFE-CPA-Friendly Projective Hash Function . . . 100
7.2.2 IPFE-CCA-Friendly Projective Hash Function . . . 101
7.3 Inner-Product Functional Encryption Schemes Based on the Decisional Composite Residuosity Assumption . . . 103
7.3.1 Scheme Secure Against Chosen-Plaintext Attacks . . . 103
7.3.2 Scheme Secure Against Chosen-Ciphertext Attacks . . . 105

8 Instantiations based on the Learning With Errors Assumption 107
8.1 The Learning With Errors Assumption . . . 108
8.1.1 The Learning With Errors Assumption . . . 108
8.1.2 Hardness Evidences . . . 108
8.2 Public Key Encryption Schemes Secure under the Learning With Error Assumption . . . 109
8.2.1 Regev Public Key Encryption Scheme . . . 109
8.2.2 Additional Properties of Regev Public Key Encryption Scheme . . . 110
8.3 Inner-Product Functional Encryption Schemes Based on the Learning With Errors Assumption . . . 116
8.3.1 Scheme Selectively Secure Against Chosen-Plaintext Attacks . . . 116
8.3.2 Scheme Adaptively Secure Against Chosen-Plaintext Attacks . . . 119

9 Conclusion and Open Questions 121
9.1 Conclusion . . . 121
9.2 Open Questions . . . 122

Notation 123

Abbreviations 125

List of Illustrations 127
Figures . . . 127
Tables . . . 128

Bibliography 129

Chapter 1

Introduction

"All warfare is based on deception."

— Sun Tzu, *The Art of War*

Throughout history, many generals have noticed that *information* was the key to victory: whoever knows the most about the enemy often triumphs in battle. To this end, the use of spies was capital. However, what use is a spy of, if his information can be intercepted or, even worse, faked? This is the reason why cryptography was first invented: to protect – with secure encryption – to authenticate – with signatures – and to hide – with steganography techniques – the information obtained by a spy behind the enemy lines so that it can be safely used as a weapon.
At this time, the use of symmetric cryptography was enough, because the spy could agree with his master upon a key before his departure. It was not until the advances of technology with the development of the Internet that public key cryptography was needed.

"With great power comes great responsibility."

— Uncle Ben, *Spider-Man*

On the Internet, anyone is able to contact everybody. However, anybody can pretend to be anyone, and can read everything. It is really hard to protect users' privacy if this ability is not used with caution. Unlike the previous example of the spy, it is not possible for a search engine or an online encyclopedia for example to meet physically with all of its users, and agree about a key with each of them, and remember all the keys. This is where public key cryptography comes into play. For example, the use of a key exchange [DH76] allows the user and the server to share a secret key, which allows them to rely on previously known techniques. However, this has a flaw: how can the user know that he communicated with the server he wanted to speak to, and not to another user trying to trick him? This explains the need of public-key encryption and digital signatures [RSA78], and authenticated variants of those primitives.
Another aspect of the Internet, is that the technical knowledge can be shared amongst everybody, and it is very hard to restrict its spread. So security has to hold even if the eventual attackers know the algorithm used to encrypt or sign messages.

"Power! Unlimited power!"

— Palpatine, *Star Wars*

With the emergence of cloud computing, the Internet is now a place where everybody puts all data online, and have servers do the work instead of their own computers. This sounds like a dream world, where everybody would have equally easy access to a lot of computing power when they need it. However, on this gigantic web, it is very hard to trust anybody with your private information. Thus, this dream seems to be unreachable: since encryption shouldn't reveal anything about the message they contain, how could someone use those data to provide you with a service? In the recent years, this challenge hasn't remained untouched. There have been many proposals that aim at bringing this dream closer to reality. For example, secure multi-party computation [Yao82; Yao86; Yun84; GMW87] would allow different users to compute together some function of their data, while only learning the output of said function. Another approach is *fully homomorphic encryption* [Gen09; BV11b; GSW13; AP14], which would allow anybody to perform some computation on one user's encrypted data, which could then be decrypted by that user in order to reveal the result of the computation. The last example we give, but not the least, is the paradigm of *functional encryption* [BSW11; GVW12; BO13; AGVW13; BF13; GGH+13].

It all started with identity-based encryption [BF01; BB04; Wat09; ABB10a; ABB10b], an encryption scheme which allows all users to use the same public key, and have each a secret key associated to their identity. One can encrypt using the public key and an identity, and only the user with this identity can recover the message. Even if multiple adversaries possessing different secret keys collude, they cannot recover anything at all. Then started the generalization process. Identity-based encryption was generalized with fuzzy identity-based encryption [SW05; ABV+12], where it is possible to encrypt a message for an identity, and decryption is possible for users whose identity is almost the same. This allows constructions of scheme where the sender can chose a set of recipients to its message, as in broadcast encryption [FN94]. Then attribute-based encryption [GPSW06; OSW07; GJPS08; LOS+10; Wat11; GVW13; BV16] was developped as a generalization of fuzzy identity-based encryption, where the choice of recipients can be given by any predicate. The next step was to hide the actual set of recipient, by hiding the identities, or attributes, used to encrypt the message, in what we call predicate encryption [KSW08; KY09; OT09; GMW15; GVW15]. The problem with all these definition is that the decryption is always all-or-nothing. A secret key either recovers completely the message or learns nothing at all.

Hence the generalization to functional encryption. In this encryption scheme, it is possible to derive secret keys for different functions, that allow for computation over the message. Then, with a secret key associated with a function, and a ciphertext, it is only possible to recover the result of this function on the encrypted message. Thus, it allows for computation on encrypted data, and to recover the result in clear. Achieving functional encryption for any computation would make our dream world come true. Some constructions have been proposed, but they are relying on existence of really complex primitives like indistinguishability obfuscation or multilinear maps. These construction are also extremely inefficient to the point where even a proof of concept seems to be infeasible for the moment.

This is why this thesis will be focusing on designing efficient schemes for the special case of *inner-product functional encryption* [ABDP15b], one of the smallest and yet powerful example of functional encryption that is non-trivial (at least superpolynomial set of possible functions) and that is not all-or-nothing – it allows partial computation on the data.

Functional Encryption for Inner-Product Evaluations. In an inner-product functional encryption scheme (IPFE), the messages are vectors, and it is possible to generate keys for vectors $\mathbf{y}$ such that given an encryption of $\mathbf{x}$, the secret key only reveals the partial information $\langle \mathbf{x}, \mathbf{y} \rangle$ about the vector $\mathbf{x}$. At first glance, this functionality might seem like a step back, from the previous definitions and constructions of identity-based encryption and attribute-based encryption. Indeed, inner-product functional encryption cannot be used to construct those kind of cryptosystems. However, neither can identity-based encryption nor attribute-based encryption be used to build IPFE. So this notion is completely separated from previous work, in that its expressivity is somewhat diminished, but the fact that it provides partial information and not all-or-nothing looks interesting. And in fact, IPFE directly finds many natural applications, in theory as well as in practice.
First, it can be used directly to compute hamming distances between two words, and more generally, the inner-product is already a distance for vectors of fixed norm. So this could find some application for example in biometric identification. IPFE could also be used in machine learning, where support vector machine evaluation can be just an inner-product between a test vector and the data. In this case, a secret key could allow classifying the data with regard to one criterion, and learn nothing about the rest of it. It also allows to make statistical analysis, like weighted means, or by encrypting the cross-products between the entries, it can even be used for variance computations or other degree 2 polynomial in the input.
Theoretically, the inner-product functionality is interesting because it allows for all computation in NC^0 by encrypting the powers of x, and using polynomial representation. Moreover, as stated in [ALS16] and refined in [AR17], it can be used to construct bounded collusion resistant functional encryption for any function. Note that this line of work has the upside of building functional encryption with succint ciphertexts, which was not reached by previous approaches. Another line of work [LV16; Lin16a; Lin16b; LT17; AS17] shows that a slightly more expressive functional encryption scheme can be used to construct indistinguishability obfuscation and functional encryption for all functions. It has also been used to construct efficient public trace and revoke schemes [ABP+17].

All these reasons made the line of work very interesting and a lot of improvements have been made to the first proposal. For example, hiding the vector $\mathbf{y}$ in the secret key [BJK15; DDM16; TAO16], using multiple vectors $\mathbf{x}_i$ to be concatenated as input [AGRW17] (we note that those parts of vectors could be encrypted separately), or using pairings to compute bilinear functions without having to increase the size of the vectors [BCFG17].

1.1 Personal Contributions

The detailed list of my personal publications with full names, authors, and conferences is given at the end of the introduction. Here we briefly describe the goal of each of these papers, as well as the results they achieve.

1.1.1 Contributions in this Thesis

[ABDP15b] In this paper, we introduce the notion of inner-product functional encryption. We give the first very simple construction from the DDH assumption, and develop a framework to generically construct IPFE schemes from public key encryption schemes.

We also include a lattice-based instantiation whose security relies on the LWE assumption. The details for the security proof of the lattice-based scheme can be found in the full version [ABDP15a]. All of the constructions of this paper reach selective security against chosen-plaintext attacks.

[ABCP16] In this paper, we refine the security of our previous construction in order to get security against adaptive adversaries. We also include a new instantiation based on the DCR assumption. As an added contribution, we also show the equivalence between non-adaptive simulation-based security and adaptive indistinguishability. This paper wasn't published in any conference because the independent concurrent work [ALS16] was accepted before, and therefore our paper was of little interest (the contributions left are the genericity of our framework, and the comparison between simulation-based security and indistinguishability). [ALS16] achieves slightly better parameters under the same assumptions and also presents a construction of functional encryption for any circuits secure against bounded collusions from their IPFE constructions by transforming their non-modular IPFE schemes into stateful modular IPFE schemes. This second result can be used together with our framework for building IPFE schemes.

[BBL17] In this paper, we further refine our construction to reach security against chosen-ciphertext attacks. In order to do so, we rely on another building block: projective hash functions. The downside of this technique is that it gives up on the lattice-based concrete constructions, but we are able to show CCA security of our scheme based on the plain DDH assumption, without the use of pairings, as well as a generalization to any MDDH assumption. We also give another instantiation based on the DCR assumption to avoid the problem of efficiency for the decryption.

1.1.2 Other Contributions

[BPMW16] In this paper, we look at the circuit privacy property for fully homomorphic encryption (FHE). This property states that computing over encrypting data using an FHE scheme shouldn't reveal anything about the computation that has been performed but the result. In this paper we show a new technique of noise analysis for lattice-based fully homomorphic encryption schemes (we show the result for one particular scheme, but we believe the same analysis could be valuable in many other instances) in order to reach this property basically for free, adding just a really small noise to the output of the computation. Our result also applies to somewhat homomorphic encryption schemes, as it removes the necessity of bootstrapping (read: homomorphically evaluating the decryption circuit on an encryption of the secret key) the ciphertext in order to have circuit privacy.

1.2 Organization of this Thesis

In Chapter 2, we first define the notations that will be used throughout the thesis, and recall preliminaries that will be necessary for proving the security of our constructions.
In Chapter 3, we formally define the notion of inner-product functional encryption and its security model. We also show alternative models of security and compare them. We also show a way to reach function hiding almost for free against bounded collusions of adversaries via

partial function hiding inner-product functional encryption, which we think is an interesting security notion in itself.
In Chapter 4, we combine the results of [ABDP15b; ABCP16] that generically builds inner-product functional encryption from public key encryption with additional properties that are defined in that chapter.
In Chapter 5, we present the construction of [BBL17] which uses projective hash functions in order to generically build IPFE schemes secure against chosen-ciphertext attacks.
We then proceed to present concrete instantiations of our generic constructions based on standard assumptions in Chapters 6, 7, and 8, where we show constructions from the decisional Diffie-Hellman assumption, the decisional composite residuosity assumption, and the learning with errors assumptions. Each of these constructions have their upsides and their downsides that are discussed.
Finally, we conclude in Chapter 9 with a brief summary of what we achieved and open questions that remain to be answered after this work.

Personal Publications

[ABCP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. *Better Security for Functional Encryption for Inner Product Evaluations*. Cryptology ePrint Archive, Report 2016/011. http://eprint.iacr.org/2016/011. 2016 (cit. on pp. 4, 5, 17, 41, 107).

[ABDP15a] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. *Simple Functional Encryption Schemes for Inner Products*. Cryptology ePrint Archive, Report 2015/017. http://eprint.iacr.org/2015/017. 2015 (cit. on pp. 4, 107).

[ABDP15b] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. "Simple Functional Encryption Schemes for Inner Products". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33 (cit. on pp. 2, 3, 5, 17, 41, 85, 107).

[BBL17] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. "CCA-Secure Inner-Product Functional Encryption from Projective Hash Functions". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Heidelberg, Mar. 2017, pp. 36–66 (cit. on pp. 4, 5, 17, 63, 85, 97).

[BPMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. "FHE Circuit Privacy Almost for Free". In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 62–89. DOI: 10.1007/978-3-662-53008-5_3 (cit. on p. 4).

Chapter 2

Preliminaries

In this chapter we present the notations that will be used throughout the thesis. Then, we recall some standard definitions adapted to our notations, and we recall some known results or lemmas that will be used in our proofs. The definitions of Section 2.2.1 will be used in Section 3.3.2 in order to transform tag-based schemes into standard tagless schemes. The definitions of Section 2.2.2 will be the basic building blocks for our first generic construction, in Chapter 4. Additional properties will be required and defined in that chapter. The definitions of Section 2.3 will be the basic building blocks for our second generic construction, in Chapter 5. Once again, additional properties will be defined in that chapter. Section 2.4.1 recalls some definitions and lemmas that will be useful for our concrete instantiations of Chapter 7, while Section 2.4.2 recalls definitions and lemmas that will be useful for the instantiations of Chapter 8.

Contents

2.1 Notation and Preliminaries **8**
- 2.1.1 Mathematical Notions 8
- 2.1.2 Provable Security 10

2.2 Basic Cryptographic Primitives **10**
- 2.2.1 One-Time Signature and Collision Resistant Hash Functions 10
- 2.2.2 Public Key Encryption 11

2.3 Projective Hash Functions **12**
- 2.3.1 Subset Membership Problems 13
- 2.3.2 Projective Hash Functions 13

2.4 Additional Preliminaries for Concrete Instantiations **14**
- 2.4.1 Basic Number Theory 14
- 2.4.2 Gaussians and Lattices 14

2.1 Notation and Preliminaries

2.1.1 Mathematical Notions

Integers, Sets. We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{Z}$ the set of integers, and by $\mathbb{R}$ the set of real numbers. The curly brackets $\{\}$ denotes a set, and $:$ is used as an abreviation for "such that". If $n \in \mathbb{N}$, then $[n]$ denotes the set $\{1, \ldots, n\}$ of positive integers up to n, $\mathrm{spf}(n)$ denotes its smallest prime factor. The size of an integer $\log n$ is the number of bits used to represent it, where log will be used to denote the base-2 logarithm. The logarithm in natural base will be denoted ln. $\{0,1\}^n$ denotes the set of n-bit strings, and if x is a string then $|x|$ denotes its length. More generally, if S is a set, then S^n is the set of n-tuples of elements of S and $|S|$ denotes the cardinality of S. We denote by $U(S)$ the uniform distribution on S, and we use the notation $x \xleftarrow{\$} S$ to say that x is sampled uniformly at random from S. For any operation $\circ$ (e.g., $+$ or $\cdot$), we extend it to sets in the following way: if t is an element and S is a set, $t \circ S = \{t \circ s \,:\, s \in S\}$.

Vectors and Matrices. Let $\mathcal{R}$ be a commutative ring. We denote the set of d-dimensional column vectors over $\mathcal{R}$ by $\mathcal{R}^d$, the set of d-dimensional row vectors by $\mathcal{R}^{1\times d}$, and the set of $\ell \times d$ matrices by $\mathcal{R}^{\ell\times d}$. Unless explicitly said otherwise, each vector is a column vector. We denote vectors by using either boldface lower-case letters or lower-case letters with an arrow over it as in $\boldsymbol{b}$ and $\overrightarrow{b}$. We denote matrices by using boldface upper-case letters like in $\boldsymbol{A}$. We have two possible notations for vectors, as we sometimes need to consider vectors of vectors ($\overrightarrow{\boldsymbol{b}}$) and vectors of matrices ($\overrightarrow{\boldsymbol{A}}$). We denote the concatenation of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ by $(\boldsymbol{a}, \boldsymbol{b})$. The transpose of a matrix is denoted $\boldsymbol{A}^\intercal$, and the inner-product between two vectors by $\langle \cdot, \cdot \rangle$. The ith coefficient of a vector $\boldsymbol{b}$ or $\overrightarrow{b}$ is denoted by b_i, while the ith coefficient of a vector of vectors $\overrightarrow{\boldsymbol{b}}$ is a vector and is denoted by $\boldsymbol{b}_i$. The jth coefficient of this latter vector is $b_{i,j}$. The same convention is used with coefficients of matrices and coefficients of vectors of matrices. Unless otherwise stated, the norm $\|\cdot\|$ considered in this paper is the ℓ_2 norm.

Abelian Groups. We extensively use Abelian groups, especially when dealing with projective hash functions. In particular, in our concrete instantiations of PHFs in Chapters 6 and 7, we use prime-order cyclic groups over an elliptic curve or subgroups of the (multiplicative) group $\mathbb{Z}_N^*$, for some positive integer N. We denote the elements of such groups by using the Fraktur script like in $\mathfrak{g}$ or $\mathfrak{b}$. By extension, even in our generic constructions and definitions in Chapter 5, we also use this font to indicate values which, in our concrete instantiations, are group elements in such group $\mathbb{G}$ or vectors of such elements. However, we are also considering other Abelian groups (e.g., the group $\mathcal{K}$ of hashing keys of a key-homomorphic PHF in Definition 5.2.1) that are not related to cryptographic assumptions and for which group elements are not denoted using the Fraktur script.

Except if explicitly stated otherwise, we use *additive notation* for all our Abelian groups, even when this is not usual (as in the case of subgroups of $\mathbb{Z}_N^*$).

Let $\mathbb{G}$ be an Abelian group. We recall that if $\mathfrak{g}$ is a group element of order M, then we have a canonical monomorphism $w \in \mathbb{Z}_M \mapsto w \cdot \mathfrak{g} \in \mathbb{G}$. If $\mathbb{G}$ is a multiplicative group, this monomorphism corresponds to exponentiation. Hence, we denote the inverse of this monomorphism by $\log_{\mathfrak{g}}$. That is, if $\mathfrak{b} = w \cdot \mathfrak{g}$, then $\log_{\mathfrak{g}} \mathfrak{b} = w$.

Furthermore, let $\mathcal{R}$ be $\mathcal{R} = \mathbb{Z}$ or $\mathcal{R} = \mathbb{Z}_M$ with M being such that the order of any group element in $\mathbb{G}$ divides M. Then $\mathbb{G}$ can be seen as a $\mathcal{R}$-module. This means that for any $w \in \mathcal{R}$ and $\mathfrak{g} \in \mathbb{G}$, $w \cdot \mathfrak{g}$ is well defined. Importantly, by using additive notation, we can use the

standard "matrix-vector" notation without prior explanation.

Asymptotic Behaviors. For two sequences of real numbers $f = (f_n)_{n\in\mathbb{N}}, g = (g_n)_{n\in\mathbb{N}}$ we use the following notations:

- $f = O(g)$ if $\exists M > 0, n_0 \in \mathbb{N}$ such that $\forall n > n_0, |f_n| \leq M|g_n|$;
- $f = o(g)$ if $\forall \epsilon > 0, \exists n_0 \in \mathbb{N}$ such that $\forall n > n_0, |f_n| \leq \epsilon|g_n|$;
- $f = \Omega(g)$ if $g = O(f)$;
- $f = \omega(g)$ if $g = o(f)$.

We also often abuse the notation by writing directly an expression, for example $f = O(n^2)$ instead of $f = O(n \mapsto n^2)$, or using those notations as functions, for example we write $f = n^{\omega(1)}$ if there exists a sequence g such that $f = n^g$ and $g = \omega(1)$. We say that a function is polynomial in n if it is $O(n^c)$ for some constant c. We say that it is superpolynomial if it is $n^{\omega(1)}$. We say that it is constant if it is $O(1)$.

Probability. If X is a probability distribution, we will write $v \xleftarrow{\$} X$ when v is sampled from the distribution X. Given two distributions X, Y over a finite or countable domain D, their statistical distance is defined as

$$\Delta(X, Y) = \frac{1}{2} \sum_{v\in D} |X(v) - Y(v)|. \tag{2.1}$$

When defining a probability distribution, we use the symbol $\propto$ to denote proportionality, the distribution can then be obtained by taking into account the normalizing factor.

We will often implicitly use the following lemmas.

Lemma 2.1.1. *Let S_1 and S_2 be two finite sets. If $S_1 \subseteq S_2$, we have $\Delta(U(S_1), U(S_2)) = 1 - |S_1|/|S_2|$. In particular, if $|S_2| = (1 + 1/t) \cdot |S_1|$ for some positive integer t, then $\Delta(U(S_1), U(S_2)) = 1/(t + 1)$.*

Proof. $\Delta(U(S_1), U(S_2)) = \frac{1}{2}\left(|S_2 \setminus S_1|/|S_2| + |S_1| \cdot (1/|S_1| - 1/|S_2|)\right) = 1 - |S_1|/|S_2|$. □

Lemma 2.1.2. *Let $S \subseteq \mathbb{Z}$ be an interval and t be an integer. Then $\Delta(U(S), U(S + t)) = |t|/|S|$.*

Proof. In the sum in Equation 2.1, exactly $2|t|$ terms are non-zero: the ones corresponding to y in $(S \setminus (S + t)) \cup ((S + t) \setminus S)$. And these terms are equal to $1/|S|$. □

Algorithms. Unless otherwise stated, all the algorithms considered in this paper are randomized, and thus $\mathscr{A}(x)$ the result of running algorithm $\mathscr{A}$ with input x can be seen as a probability distribution. We sometimes explicit the random tape of $\mathscr{A}$: $\mathscr{A}(x; r)$ denotes running algorithm $\mathscr{A}$ on input x with randomness r. If $\mathscr{A}$ is an algorithm, we will write $v \xleftarrow{\$} \mathscr{A}(x)$ to denote running $\mathscr{A}$ on input x and assigning the result to the variable v. We will instead write $v \leftarrow \mathscr{A}(x)$ if $\mathscr{A}$ is deterministic. We also use this definition for any computation, e.g., $v \leftarrow x + y$, thus we reserve the notation $=$ for equality testing and definitions (or assignments without computation). An algorithm is called polynomial, or efficient if its running time is polynomial in the size of its inputs. We sometimes allow algorithms to have oracle access to probability distributions or other algorithms and write $\mathscr{A}^{\mathcal{O}}$ the fact that the algorithm $\mathscr{A}$ can use oracle $\mathcal{O}$.

2.1.2 Provable Security

Security Parameter. In public key cryptography, any problem can be solved if given enough time and efforts. The goal of provable security is to measure the quantity of time and efforts required to solve a problem. To do so, we implicitely have a certain security parameter κ, and every integers, vectors, matrices, functions that we define will implicetely be defined with regard to that security parameter, and we often omit this in the definitions to make everything more readable. We explicitely remind this sometimes, for example our algorithms Setup will take as input the security parameter in unary representation 1^κ. The reason it is represented in unary is because we want the algorithm to run in a time that is polynomial in κ, so its input must have size κ. We usually state that an algorithm is efficient if its running time is polynomial in κ, and inefficient if not. Similarly, we say that a function f is negligible if it is $o(1/\kappa^c)$ for any constant $c \in \mathbb{N}$, and we say that a probability p is overwhelming if $1 - p$ is negligible. We consider a scheme secure if there is no efficient algorithm that has a non-negligible probability of "breaking the scheme".
Given two distributions X, Y over a finite or countable domain D, we say that they are equal if their statistical distance $\Delta(X, Y)$ is 0. We say that they are statistically indistinguishable if $\Delta(X, Y)$ is negligible. We say that they are computationally indistinguishable if $\Delta(\mathscr{A}^X, \mathscr{A}^Y)$ is negligible for all probabilistic polynomial time (i.e., efficient) algorithms $\mathscr{A}$. As an analogy with the applications, we often call adversary an algorithm designed to attack the security of a scheme.

Code-Based Games. We use the code-based game-playing [BR06] to define the security notions. In such games, there exist procedures for initialization (**Initialize**) and finalization (**Finalize**) and procedures to respond to adversary oracle queries. A game G is executed with an adversary $\mathscr{A}$ as follows. First, **Initialize** executes and its outputs are the inputs to $\mathscr{A}$. Then $\mathscr{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathscr{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted $\mathrm{G}(\mathscr{A})$, is called the output of the game, and "$\mathrm{G}(\mathscr{A}) = y$" denotes the event that the output takes a value y. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true.

2.2 Basic Cryptographic Primitives

2.2.1 One-Time Signature and Collision Resistant Hash Functions

One-Time Signatures. The purpose of a one-time signature is to prevent messages to be tampered with. Basically, it allows to sign a message and give out the verification key that guarantees that the message was generated as a whole, and that no parts of it has been replaced, added, or removed.

Definition 2.2.1 (One-Time Signature)**.** *A* one-time signature scheme *is a tuple* $\mathcal{OTS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *of three probabilistic polynomial time algorithms:*

- $\mathsf{Setup}(1^\kappa) \mapsto (\mathsf{sk}, \mathsf{vk})$*. On input security parameter* κ *outputs a signing key* sk *and a verification key* vk*;*
- $\mathsf{Sign}(\mathsf{sk}, m) \mapsto (\sigma)$*. On input signature key* sk *and message* m *returns a signature* σ*;*

- $\mathsf{Verify}(\mathsf{vk}, m, \sigma) \mapsto \emptyset$ *or* $\perp$. *On input verification key* vk, *message* m *and signature* σ *returns nothing if the signature is valid, and* $\perp$ *otherwise.*

$\mathcal{OTS}$ must be *complete*, in the sense that if m is in the message space, then for all $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$, $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$, it holds that $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ passes the check.

Definition 2.2.2 (One-Time Unforgeability)**.** *A one-time signature scheme* $\mathcal{OTS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *is* strongly unforgeable*, if no probabilistic polynomial time adversary* $\mathscr{A}$ *has a non-negligible advantage in the following game:*

1. *The challenger sets* $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{Setup}$ *and sends* vk *to* $\mathscr{A}$.
2. $\mathscr{A}$ *chooses adaptively messages queries* m_i *and sends them to the challenger. The challenger sends back* $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$ *to* $\mathscr{A}$ *for each* i.
3. $\mathscr{A}$ *outputs a forgery* $(m', \sigma') \notin \{(m_i, \sigma_i)\}$ *and wins if* $\mathsf{Verify}(\mathsf{vk}, m', \sigma')$.

Collision Resistant Hash Functions. We call family of hash functions a family of functions $(\mathsf{H}_k)_{k \in \mathcal{K}}$ that maps arbitraty bitstrings to fixed-length bitstrings: $\mathsf{H}_k : \times\{0,1\}^* \to \{0,1\}^\kappa$.

Definition 2.2.3 (Collision Resistance)**.** *A family of hash functions* $(\mathsf{H}_k)_{k \in \mathcal{K}}$ *is* collision resistant*, if no probabilistic polynomial time adversary* $\mathscr{A}$ *can output a pair* (m, m') *such that* $\mathsf{H}_k(m) = \mathsf{H}_k(m')$ *with non-negligible probability for a randomly chosen key* k.

This property is not only useful for cryptographic purposes, but it can also be used to check if two users have the same data for example, they just have to compare the fixed-length hash of their database.

2.2.2 Public Key Encryption

Some of our generic construction of inner-product functional encryption are using public key encryption scheme as building blocks. We state here the formal definition of PKE, as well as the security notion we require for our constructions.

Definition 2.2.4 (Public Key Encryption)**.** *A* public key encryption *(PKE) scheme* $\mathcal{E}$ *consists of 4 probabilistic polynomial time algorithms:*

- $\mathsf{Setup}(1^\kappa) \mapsto (pp, sk, pk)$. *On input security parameter* κ *outputs public parameters* pp, *secret key* sk, *and public key* pk;
- $\mathsf{Encrypt}(pp, pk, m) \mapsto ct$. *On input public parameter* pp, *public key* pk, *and plaintext* m, *outputs ciphertext* ct;
- $\mathsf{Decrypt}(pp, sk, ct) \mapsto m$ *or* $\perp$. *On input public parameters* pp, *secret key* sk, *and ciphertext* ct, *outputs a message* m *or an error symbol* $\perp$.

Definition 2.2.5 (Indistinguishability under Selective Chosen-Plaintext Attacks)**.** *For a PKE scheme* $\mathcal{E}$, *we define* selective security against chosen-plaintext attacks *(*s-IND-CPA security*) via the security game depicted on Figure 2.1: we define the advantage of an adversary* $\mathscr{A}$ *to be*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}cpa}}_{\mathcal{E},\kappa}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}cpa\text{-}b}}_{\mathcal{E},\kappa}(\mathscr{A}) = 1] - 1 \right|.$$

Then we say that $\mathcal{E}$ *is* secure against chosen-plaintext attacks *(*s-IND-CPA secure*) if* $\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}cpa}}_{\mathcal{E},\kappa}(\mathscr{A})$ *is negligible for all probabilistic polynomial time* $\mathscr{A}$.

Game $\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}cpa\text{-}b}}_{\mathcal{E},\kappa}(\mathscr{A})$

proc Initialize(κ, m_0^*, m_1^*)	**proc Encrypt**$()$
$(pp, sk, pk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$	$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, pk, m_b^*)$
Return (pp, pk)	Return ct
	proc Finalize(b')
	Return $(b' = b)$

Figure 2.1: Game $\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}cpa\text{-}b}}_{\mathcal{E},\kappa}(\mathscr{A})$ defines s-IND-CPA security of $\mathcal{E}$.

The more standard security notion would be adaptibe security against chosen-plaintext attacks, where the challenge messages m_0^* and m_1^* would be chosen when calling **Encrypt**. We chose to only use a selective notion because it is a weaker assumption, and it was sufficient for our construction.

Functional Encryption. Functional Encryption has been introduced to generalize the concept of public key encryption and identity-based encryption. We briefly recall the definition for completeness. We don't elaborate on the security definitions, since we will discuss that matter on the special case of inner-product functional encryption in Chapter 3.

A *functionality* $\mathcal{F}$ defined over $(\mathcal{M}_y, \mathcal{M}_x)$ is a function $\mathcal{M}_y \times \mathcal{M}_x \to \Sigma \cup \{\perp\}$, where $\mathcal{M}_y$ is a key space, $\mathcal{M}_x$ is a message space, and Σ is an output space that does not contain the special symbol $\perp$.

A *functional encryption scheme for functionality* $\mathcal{F}$ [ONe10; BSW11] is a tuple $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ of four probabilistic polynomial time algorithms:

- $\mathsf{Setup}(1^\kappa) \mapsto (pp, msk, mpk)$. On input security parameter κ outputs public parameters pp, master secret key msk, and master public key mpk;
- $\mathsf{KeyDer}(pp, msk, y) \mapsto sk_y$. On input public parameter pp, master secret key msk, and key $y \in \mathcal{M}_y$, outputs user secret key sk_y;
- $\mathsf{Encrypt}(pp, mpk, x) \mapsto ct_x$. On input public parameter pp, master public key mpk, and plaintext $x \in \mathcal{M}_x$, outputs ciphertext ct_x;
- $\mathsf{Decrypt}(pp, sk_y, ct_x) \mapsto m$ or $\perp$. On input public parameters pp, user secret key sk_y, and ciphertext ct_x, outputs a message m or an error symbol $\perp$.

We say that a FE scheme is correct if $m = \mathcal{F}(y, x)$. The special case of *inner-product functional encryption* is $\mathcal{F}(\mathbf{y}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$.

2.3 Projective Hash Functions

Before being able to define projective hash functions, we have to define what is a subset membership problem, which will be used in the definition of PHF.

2.3.1 Subset Membership Problems

Our framework uses *subset membership problems*, which were originally defined in [CS02]. Basically, a subset membership problem defines an NP language $\mathcal{L} \subset \mathcal{X}$, in which a random word in $\mathcal{L}$ is hard to distinguish from a random word in $\mathcal{X} \setminus \mathcal{L}$. In this thesis, we consider a slight extension, where we instead require a random word in $\mathcal{L}$ to be hard to distinguish from a random word in a given set $\bar{\mathcal{L}} \subseteq \mathcal{X} \setminus \mathcal{L}$.

More formally, a subset membership problem $\mathbf{P}$ specifies an ensemble $(I_\kappa)_{\kappa \geq 0}$ of distributions. For every value of a security parameter $\kappa \geq 0$, I_κ is a probability distribution of instance descriptions. An instance description $\Lambda = \Lambda[\mathcal{X}, \mathcal{L}, \mathcal{W}, \varrho, \bar{\mathcal{L}}]$ specifies the following: (a) finite, non-empty sets $\mathcal{X}$, $\mathcal{L}$, $\mathcal{W}$, and $\bar{\mathcal{L}}$, such that $\mathcal{L}$ is a proper subset of $\mathcal{X}$ and $\bar{\mathcal{L}}$ is a non-empty subset of $\mathcal{X} \setminus \mathcal{L}$, (b) a binary relation $\varrho \subset \mathcal{X} \times \mathcal{W}$. For $\mathfrak{b} \in \mathcal{X}$ and $w \in \mathcal{W}$, we say that w is a witness for $\mathfrak{b}$ if $(\mathfrak{b}, w) \in \varrho$. We require that instance descriptions and elements of $\mathcal{X}$ and $\mathcal{W}$ can be uniquely encoded as bitstrings of length $\mathrm{poly}(\kappa)$.

A subset membership problem satisfies the following properties:

1. I_κ is efficiently samplable, which means that there exists a probabilistic polynomial time instance sampling algorithm that on input 1^κ samples an instance Λ according to the distribution I_κ;

2. ϱ is efficiently samplable, which means that there exists a probabilistic polynomial time subset sampling algorithm that on input Λ outputs a random $\mathfrak{b} \in \mathcal{L}$ together with a witness $w \in \mathcal{W}$ for $\mathfrak{b}$; the distribution over ϱ implicitly defines a distribution over $\mathcal{L}$;

3. $\bar{\mathcal{L}}$ is efficiently samplable;

4. $\mathcal{X}$ is efficiently recognizable, which means that there exists a deterministic polynomial algorithm that on input (Λ, ζ) checks whether ζ is a valid binary encoding of an element of $\mathcal{X}$;

5. ϱ is efficiently recognizable;

6. $(\mathcal{L}, \bar{\mathcal{L}})$-*indistinguishability:* a sample from $\mathcal{L}$ is computationally indistinguishable from a sample from $\bar{\mathcal{L}}$.

We do not require the distributions over ϱ, $\mathcal{L}$, and $\bar{\mathcal{L}}$ to be uniform. However, when we do not specify these distributions, we implicitly use the uniform distributions.

2.3.2 Projective Hash Functions

We are now ready to state the formal definition of projective hash functions.

Definition 2.3.1 (Projective Hash Function)**.** *Let* $\mathbf{P}$ *be a subset membership problem, specifying an ensemble* $(I_\kappa)_\kappa$ *of instance distributions. A* projective hash function *for* $\mathbf{P}$ *is a tuple* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *of four probabilistic polynomial time algorithms:*

- $\mathsf{hashkg}(\Lambda)$ *generates a hashing key* hk *in some set* $\mathcal{K}$ *for the instance* $\Lambda = \Lambda[\mathcal{X}, \mathcal{L}, \mathcal{W}, \varrho]$,

- $\mathsf{projkg}(\mathsf{hk})$ *(deterministically) derives from the hashing key* hk *a projection key* $\mathfrak{hp}$,

- $\mathsf{hash}(\mathsf{hk}, \mathfrak{b})$ *(deterministically) computes the hash value* $\mathfrak{H}$ *(in some efficiently recognizable set* Π*) of* $\mathfrak{b} \in \mathcal{X}$ *under* $\mathsf{hk} \in \mathcal{K}$,

Chapter 2

- $\mathsf{projhash}(\mathfrak{hp}, \mathfrak{b}, w)$ *(deterministically) computes the projected hash value* $\mathfrak{pH}$ *of* $\mathfrak{b} \in \mathcal{L}$ *using a witness* $w \in \mathcal{W}$.

A PHF must be complete*, in the following sense:*

- *For any instance* Λ*, for any* $\mathfrak{b} \in \mathcal{X}$ *and* $w \in \mathcal{W}$*, such that* $(\mathfrak{b}, w) \in \varrho$*, for any hashing key* $\mathsf{hk} \in \mathcal{K}$*, if* $\mathfrak{hp} \leftarrow \mathsf{projkg}(\mathsf{hk})$*, then*

$$\mathsf{hash}(\mathsf{hk}, \mathfrak{b}) = \mathsf{projhash}(\mathfrak{hp}, \mathfrak{b}, w) \ .$$

The instance Λ is implicitly included in the hashing key hk and the projection key $\mathfrak{hp}$.

The standard statistical requirement for a projective hash function is to be smooth. The smoothness property essentially states that the hash of a word not in the language is close to uniform, and so unguessable. In our proof, we will require other properties that we define in the beginning Chapter 5.

2.4 Additional Preliminaries for Concrete Instantiations

We now recall some additional preliminaries that will be required for our concrete instantiations, starting with preliminaries that will be used for the instantiations based on DCR, and then ones that will be used for the instantiations based on LWE.

2.4.1 Basic Number Theory

Let N be a positive integer. Let $\varphi(N)$ be the Euler totient function. For any integer a and an odd prime q, the Legendre symbol $\left(\frac{a}{q}\right)$ is defined as

$$\left(\frac{a}{q}\right) = \begin{cases} 0 & \text{If } a \equiv 0 \pmod{q} \\ +1 & \text{If } \quad a \not\equiv 0 \pmod{q} \text{ and for some integer } y,\ a \equiv y^2 \pmod{q} \\ -1 & \text{If } a \not\equiv 0 \pmod{q} \text{ and there is no such } y \end{cases}$$

For any integer a and any positive odd integer N, the Jacobi symbol is defined as the product of the Legendre symbols corresponding to the prime factors of N, $\left(\frac{a}{N}\right) := \prod_{i=1}^{t} \left(\frac{a}{p_i}\right)^{\alpha_i}$, where $N = \prod_{i=1}^{t} p_i^{\alpha_i}$ for distinct primes p_i. Let $\mathbb{J}_N = \{a \in \mathbb{Z}_N \ : \ \left(\frac{a}{N}\right) = 1\}$; clearly $\mathbb{J}_N$ is a subgroup of $\mathbb{Z}_N^*$. The Jacobi symbol can be computed in polynomial time, given only a and N [JOA96, Algorithm 2.149].

2.4.2 Gaussians and Lattices

We first recall the definition of a lattice before defining discrete Gaussian distribution over a lattice. We also recall some lemmas that will be used in Chapter 8.

Lattices. A m-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$ (e.g., $\mathbb{Z}^m$). For an integer $k < m$ and a rank k matrix $\boldsymbol{B} \in \mathbb{R}^{m \times k}$, $\Lambda(\boldsymbol{B}) = \{\boldsymbol{B}\mathbf{x} \in \mathbb{R}^m \ : \ \mathbf{x} \in \mathbb{Z}^k\}$ is the lattice generated by the columns of $\boldsymbol{B}$.

Gaussians. The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to [0, 1]$ is defined as

$$\rho(\mathbf{x}) = \exp(-\pi \cdot ||\mathbf{x}||^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle) \ .$$

Applying a linear transformation given by a (not necessarily square) matrix $\boldsymbol{B}$ with linearly independent columns yields the (possibly degenerate) Gaussian function

$$\rho_{\boldsymbol{B}}(\mathbf{x}) = \begin{cases} \rho(\boldsymbol{B}^{-1}\mathbf{x}) = \exp(-\pi \cdot \mathbf{x}^\intercal \Sigma^{-\intercal}\mathbf{x}) & \text{If } \mathbf{x} \in \mathrm{Span}(\boldsymbol{B}) = \mathrm{Span}(\Sigma) \\ 0 & \text{otherwise} \end{cases}$$

where $\Sigma = \boldsymbol{B}\boldsymbol{B}^\intercal \leq 0$. Because $\rho_{\boldsymbol{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

We say that a distribution D is subgaussian with parameter σ if there exists a constant $k \in \mathbb{R}$ such that $D(x) \leq k \cdot \rho_\sigma(x)$.

Definition 2.4.1 (Smoothing parameter)**.** *For a lattice $\Lambda \subseteq \mathbb{Z}^m$ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$, where $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^m : \mathbf{x}^\intercal \Lambda \subseteq \mathbb{Z}\}$.*

The next result gives a bound on the smoothing parameter of a generic lattice.

Lemma 2.4.2 ([MR07, Lemma 3.3])**.** *Let Λ be any rank-m lattice and ϵ be any positive real. Then*

$$\eta_\epsilon(\Lambda) \leq \lambda_m(\Lambda) \cdot \sqrt{\frac{\ln(2m(1+1/\epsilon))}{\pi}}$$

where $\lambda_m(\Lambda)$ is the smallest R such that the ball $\mathcal{B}_R$ centered in the origin and with radius R contains m linearly independent vectors of Λ.

Corollary 2.4.3.

$$\eta_\epsilon(\mathbb{Z}^\ell) \leq \sqrt{\frac{\ln(2\ell(1+1/\epsilon))}{\pi}}$$

The following lemma gives a bound on values drawn from subgaussian distributions.

Lemma 2.4.4 ([AP14, Lemma 2.1])**.** *There exists a universal constant $C > 0$, such that*

$$\Pr\left[\, \|\mathbf{x}\| > C\sigma\sqrt{m} \,\right] \leq 2^{-\Omega(m)}$$

where $\mathbf{x}$ is drawn from a subgaussian distribution over $\mathbb{Z}^m$ with parameter σ.

Lemma 2.4.5. *Let $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ be the probability that a gaussian random variables of covariance matrix Σ is equal to $\mathbf{x}$. Then, for any invertible matrix β,*

$$\rho_{\sqrt{\Sigma}}(\beta^{-1}\mathbf{x}) = \rho_{\beta\sqrt{\Sigma}}(\mathbf{x}) \ .$$

Lemma 2.4.6. *Let $\Lambda \subset \mathbb{R}^n$ be a lattice. For any $\Sigma \geq \mathbf{0}$ and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \leq \rho_{\sqrt{\Sigma}}(\Lambda)$. Moreover, if $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ for some $\epsilon > 0$ and $\mathbf{c} \in \mathrm{Span}(\Lambda)$, then $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \geq \frac{1-\epsilon}{1+\epsilon} \cdot \rho_{\sqrt{\Sigma}}(\Lambda)$.*

Generalized Leftover Hash Lemma. The following text and lemma are taken verbatim from [DRS04]. The predictability of a random variable A is $\max_a \Pr[A = a]$, and its min-entropy $\mathbf{H}_\infty(A)$ is $-\log(\max_a \Pr[A = a])$. The min-entropy of a distribution tells us how many nearly uniform random bits can be extracted from it.

Chapter 2

Consider now a pair of (possibly correlated) random variables A, B. If the adversary finds out the value b of B, then predictability of A becomes $\max_a \Pr[A = a|B = b]$. On average, the adversary's chance of success in predicting A is then $\mathbb{E}_{b \xleftarrow{\$} B}[\max_a \Pr[A = a|B = b]]$. Note that we are taking the average over B (which is not under adversarial control), but the worst case over A (because prediction of A is adversarial once b is known). Again, it is convenient to talk about security in log-scale, which is why we define the average min-entropy of A given B as simply the logarithm of the above:

$$\tilde{\mathbf{H}}_\infty(A|B) = -\log\left(\mathbb{E}_{b \xleftarrow{\$} B}[\max_a \Pr[A = a|B = b]]\right) = -\log\left(\mathbb{E}_{b \xleftarrow{\$} B}\left[2^{\mathbf{H}_\infty(A|B=b)}\right]\right) .$$

Lemma 2.4.7. *Let A, B, C be random variables. Then If B has at most 2^κ possible values, then*

$$\tilde{\mathbf{H}}_\infty(A|(B, C)) \geq \tilde{\mathbf{H}}_\infty((A, B)|C) - \kappa \geq \tilde{\mathbf{H}}_\infty(A|C) - \kappa ,$$

In particular, $\tilde{\mathbf{H}}_\infty(A|B) \geq \mathbf{H}_\infty((A, B)) - \kappa \geq \mathbf{H}_\infty(A) - \kappa$.

Lemma 2.4.8. *[Generalized Leftover Hash Lemma [DRS04]] Assume $\{H_x : \{0,1\}^n \to \{0,1\}^\ell\}_{x \in X}$ is a family of universal hash functions. Then, for any random variables W and I,*

$$\Delta((H_X(W), X, I), (U_\ell, X, I)) \leq \frac{1}{2}\sqrt{2^{-\tilde{H}_\infty(W|I)} 2^\ell} .$$

Chapter 3

Inner-Product Functional Encryption

In this Chapter, we provide the formal definition of inner-product functional encryption and discuss the multiple ways to model its security, as well as the desirable properties it can have. We start off with the definition as we introduced in [ABDP15b] and explain the functionality it provides together with the standard security notion that we will use: indistinguishability under chosen-plaintext attack.
Then we compare this model with the weaker model of selective security, and with the other intuitive way to model security: simulation-based security. We show that adaptive indistinguishability-based security is equivalent to the non-adaptive simulation-based security (result from our paper [ABCP16]).
We also introduce the security against chosen-ciphertext attacks. We define the related tag-based inner-product functional encryption, which is a variant of IPFE that is easier to analyze in the case of CCA security, and we show how to translate it into a CCA secure standard inner-product functional encryption scheme, as done in our contribution [BBL17].
Finally, we discuss the function hiding hiding property, another desirable security property for an inner-product encryption scheme which means that a secret key doesn't reveal the computation it allows. We also show how to obtain weak versions of this property without any additional assumption. This work is exclusive to this thesis.

Contents

3.1 Definition and Security Model **18**
3.1.1 Definition 18
3.1.2 Indistinguishability-Based Security 19
3.2 Alternative Security Models **22**
3.2.1 Selectivity vs. Adaptivity 22
3.2.2 Simulation-Based Security 25
3.3 Indistinguishability under Chosen-Ciphertext Attack **27**
3.3.1 Tag-Based Inner-Product Functional Encryption 28
3.3.2 Removing the Tag 30
3.4 Function Hiding **31**
3.4.1 Function Hiding in the Public Key Setting 32
3.4.2 Secret Key Inner-Product Functional Encryption 33
3.4.3 Partial Function Hiding 34

3.1 Definition and Security Model

3.1.1 Definition

The definition of an inner-product functional encryption scheme is best illustrated with an example of what it allows to do. A diagram showing the functionality provided by an IPFE scheme can be found on Figure 3.1. In this case, a user Bob transmits to another user Alice the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ between a vector $\mathbf{x}$ that he knows and any vector $\mathbf{y}$ for which Alice has a certified secret key:

- The trusted authority generates a pair (mpk, msk) of master public key and master secret key. The former can be made public and retrieved by Bob, while the later will be kept secret by the authority.
- Bob uses the master public key mpk to encrypt his vector $\mathbf{x}$. He then makes public the resulting ciphertext $ct_{\mathbf{x}}$.
- Alice makes a request for a secret key $sk_{\mathbf{y}}$ for her vector $\mathbf{y}$, which the authority derives using the master secret key msk, and privately sends to Alice if she is entitled.
- Alice uses her key $sk_{\mathbf{y}}$ to decrypt the ciphertext $ct_{\mathbf{x}}$ that she received from Bob and receives the value $\langle \mathbf{x}, \mathbf{y} \rangle$.

We say that an inner-product functional encryption scheme is correct if the result is indeed $\langle \mathbf{x}, \mathbf{y} \rangle$.

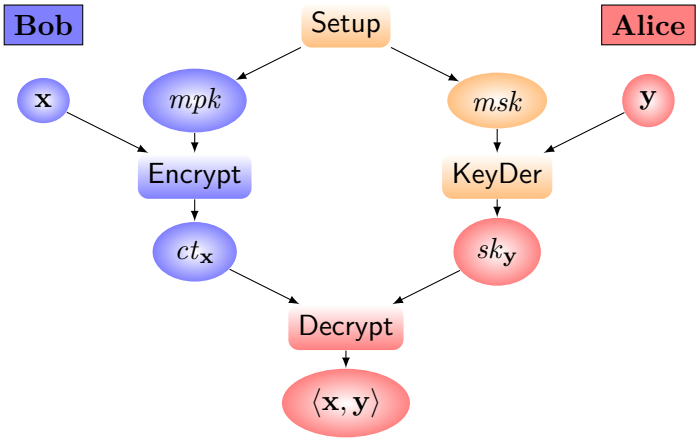

Figure 3.1: Illustration of an Inner-Product Functional Encryption scheme. Colors are used as follows: in orange are the actions performed and the data owned by the trusted authority; in red are the actions performed and the data owned by some user Alice; and in blue are the actions performed and the data owned by another user Bob.

More formally, we have the following definition and correctness property.

Definition 3.1.1 (Inner-Product Functional Encryption)**.** *An* inner-product functional encryption *(IPFE) scheme $\mathcal{FE}$ is defined with regard to a some ring $\mathcal{R}$, and consists of 4 probabilistic polynomial time algorithms:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp, msk, mpk)$. *On input security parameter κ and functionality parameter ℓ outputs public parameters pp, master secret key msk, and master public key mpk;*

- $\mathsf{KeyDer}(pp, msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input public parameter pp, master secret key msk, and key $\mathbf{y} \in \mathcal{R}^\ell$, outputs user secret key $sk_{\mathbf{y}}$;*

- $\mathsf{Encrypt}(pp, mpk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input public parameter pp, master public key mpk, and plaintext $\mathbf{x} \in \mathcal{R}^\ell$, outputs ciphertext $ct_{\mathbf{x}}$;*

- $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input public parameters pp, user secret key $sk_{\mathbf{y}}$, and ciphertext $ct_{\mathbf{x}}$, outputs a message $m \in \mathcal{R}$ or an error symbol $\perp$.*

To lighten the notations, we sometimes omit the public parameters pp or assume they are included in the public key. We also sometimes implicitely assume that $\mathbf{y}$ is included in clear in the user secret key $sk_{\mathbf{y}}$. We will focus on the rings $\mathbb{Z}_q$ for some prime number q (we also sometimes denote this prime number p to avoid confusions), and $\mathbb{N}$. We will sometimes refer to the first case as modular inner-product functional encryption, and to the second as non-modular inner-product functional encryption. There are some major differences between the two: modular IPFE is more friendly for security analysis and also for applications. However, the only instantiations we know of requires the computation of a discrete logarithm in order to decrypt, which means that we can only decrypt inner-products in a fixed polynomial range, thus the whole plaintext space is not used. In the case of non-modular IPFE, the computation is usually done modulo some integer, but we require the inner-produt result to be smaller than the modulus for security issues. [ALS16] describes a technique to build modular IPFE schemes from non-modular one, the downside being that the key generation algorithm KeyDer has to be stateful. The idea is to remember the queries that were answered, and change the queries that would allow a break into queries that give the same information without threatening the security of the scheme.

Correctness. To account for the previous remarks, we allow the Decrypt algorithm to output $\perp$ if $\langle \mathbf{x}, \mathbf{y} \rangle$ is outside of a polynomial sized set $\mathcal{P}$ fixed at setup. We also denote the plaintext space $\mathcal{M}_x$ and the key space $\mathcal{M}_y$, and they may be smaller than the full ring $\mathcal{R}$. Hence the following *correctness* property:
An IPFE scheme is correct if for any $\mathbf{x} \in \mathcal{M}_x, \mathbf{y} \in \mathcal{M}_y, (pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$, $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$, $ct_{\mathbf{x}} \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x})$. Then if $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{P}$, it holds that $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) = \langle \mathbf{x}, \mathbf{y} \rangle$.

3.1.2 Indistinguishability-Based Security

We will use a left-or-right indistinguishability based definition of security also called security against chosen-plaintext attacks. Intuitively, it means that an eavesdropping adversary cannot tell appart between encryptions of messages of his chosing. For example, this implies that the scheme has no weak plaintexts, otherwise the adversary could choose to attack this particular message, and it also means that the scheme is non-deterministic, because this would lead to a trivial attack in the model. It has been used for many years in the context of public key encryption as well as symmetric key encryption, and has been adapted to identity-based encryption and functional encryption as the most intuitive security property.

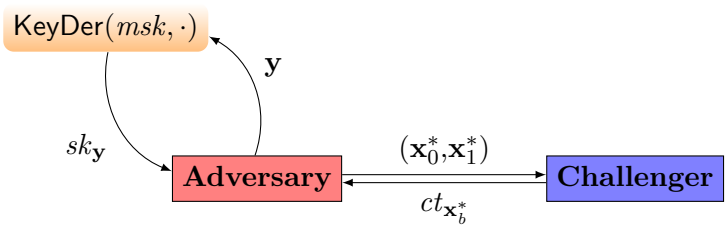

Figure 3.2: Illustration of the IND-IPFE-CPA security game. **Adversary** wins if $(b' = b) \wedge \forall \mathbf{y}, \langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$.

As for the definition of IPFE, we give an informal illustration of the security model for intuition on Figure 3.2. Here we only pictured a single challenge couple $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ because as we show later, it is equivalent.

Definition 3.1.2 (Indistinguishability under Chosen-Plaintext Attacks)**.** *For an inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *defined over* $\mathcal{R}$*, we define* security against chosen-plaintext attacks *(*IND-IPFE-CPA security*) via the security game depicted on Figure 3.3: we define the advantage of an adversary* $\mathscr{A}$ *to be*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1 \right|.$$

Then we say that $\mathcal{FE}$ *is* secure against chosen-plaintext attacks *(*IND-IPFE-CPA secure*) if* $\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A})$ *is negligible for all probabilistic polynomial time* $\mathscr{A}$*.*

The event bad corresponds to the adversary making queries that trivially allows him to find b. Those are attack that we cannot be secure against, by definition of the functionality. The goal of the adversary is to find b without having access to those queries.

Game $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ)
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$
$V_{\mathbf{y}} \leftarrow \emptyset$
$V_{\mathbf{x}} \leftarrow \emptyset$
Return (pp, mpk)

proc KeyDer$(\mathbf{y})$
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{\mathbf{y}\}$
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$
Return $sk_{\mathbf{y}}$

proc Encrypt$(\mathbf{x}_0, \mathbf{x}_1)$
$V_{\mathbf{x}} \leftarrow V_{\mathbf{x}} \cup \{(\mathbf{x}_0, \mathbf{x}_1)\}$
$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_b)$
Return ct

proc Finalize(b')
if bad
  then return false
Return $(b' = b)$

Figure 3.3: Game $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines IND-IPFE-CPA security of $\mathcal{FE}$. The event bad corresponds to $\exists \mathbf{y} \in V_{\mathbf{y}}, \exists (\mathbf{x}_0, \mathbf{x}_1) \in V_{\mathbf{x}}, \langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$.

Because we are dealing with public key encryption schemes, we can without loss of generality only consider the adversaries $\mathscr{A}$ that only makes one encryption query $(\mathbf{x}_0^*, \mathbf{x}_1^*)$. In particular, we have the following theorem:

Theorem 3.1.3. *For any adversary $\mathscr{A}$ making q encryption queries $\{(\mathbf{x}_0^j, \mathbf{x}_1^j)\}_{j\in[q]}$, there exists an adversary $\mathscr{B}$ making only 1 encryption query which runs in roughly the same time as $\mathscr{A}$ and such that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B}) \geq \frac{1}{q} \cdot \mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A})$$

Proof. We prove this theorem by using a standard hybrid argument [GM84]. The adversary $\mathscr{B}$ is as follows: It first picks a random number $i \in \{0, \ldots, q\}$. It forwards $\mathscr{A}$'s queries to **Initialize**, **KeyDer** and **Finalize**. However, it behaves differently with **Encrypt** on input $(\mathbf{x}_0^j, \mathbf{y}_1^j)$:

- If i<j, it sets $ct = \mathsf{Encrypt}(pp, mpk, \mathbf{x}_0)$;
- If i=j, it invokes **Encrypt**;
- If i>j, it sets $ct = \mathsf{Encrypt}(pp, mpk, \mathbf{x}_1)$.

Now, to analyze the advantage of $\mathscr{B}$, we define the following hybrid experiment:

Game $\mathbf{H}_i$

proc Initialize(κ, ℓ)	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	**if** $\lvert V\rvert < i$
$V_{\mathbf{y}} \leftarrow \emptyset$	**then** $ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_0)$
$V_{\mathbf{x}} \leftarrow \emptyset$	**if** $\lvert V\rvert \geq i$
Return (pp, mpk)	**then** $ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_1)$
proc KeyDer$(\mathbf{y})$	$V_{\mathbf{x}} \leftarrow V_{\mathbf{x}} \cup \{(\mathbf{x}_0, \mathbf{x}_1)\}$
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{\mathbf{y}\}$	Return ct
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$	**proc Finalize**(b')
Return $sk_{\mathbf{y}}$	**if** bad
	then return false
	Return $(b' = b)$

Notice that we have:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A}) \\
&= \left|2 \cdot \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1\right| \\
&= \left|\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}0}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] + \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}1}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1\right| \\
&= \left|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\right| \\
&= \left|\Pr_{\mathrm{H}_1}[b' = 1] - \Pr_{\mathrm{H}_{q+1}}[b' = 1]\right| \\
&= \left|\sum_{i\in[q]} \Pr_{\mathrm{H}_i}[b' = 1] - \Pr_{\mathrm{H}_{i+1}}[b' = 1]\right| \\
&\leq \sum_{i\in[q]} \left|\Pr_{\mathrm{H}_i}[b' = 1] - \Pr_{\mathrm{H}_{i+1}}[b' = 1]\right| \\
&= q \cdot \mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B})
\end{aligned}$$

□

Remark 3.1.4. *We note that this reduction is not tight in the general case, but with most instantiations, one can use random self-reducibility of the underlying assumption to have a better reduction in the case of multiple encryption challenges. A good example would be the resulting scheme based on the decisional Diffie-Hellman assumption presented in Chapter 6.*

In the rest of this chapter, we will present alternative security notions and discuss their relations with IND-IPFE-CPA security.

3.2 Alternative Security Models

3.2.1 Selectivity vs. Adaptivity

A natural weakening of IND-IPFE-CPA security is security against selective chosen-plaintext attacks. The difference with the previous definition is that in this new model, the adversary has to chose the challenge messages beforehand. It cannot see the public key or make any key query before he choses the messages. This means that the security will be easier to prove: for example, in the special case of IPFE, this means that the adversary has to chose the challenge vectors $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$, so that its queries are restricted to a known set $(\mathbf{x}_1^* - \mathbf{x}_0^*)^\perp$. To avoid ambiguity, we sometimes refer to IND-IPFE-CPA as adaptive security in contrast with selective security. We now define s-IND-IPFE-CPA security and compare it to IND-IPFE-CPA. We also show that a modular IPFE scheme that satisfy s-IND-IPFE-CPA security can be slightly modified to be IND-IPFE-CPA secure if it satisfies a very natural homomorphic property.

Definition 3.2.1 (Indistinguishability under Selective Chosen-Plaintext Attacks)**.** *For an IPFE scheme $\mathcal{FE}$, we define* selective security against chosen-plaintext attacks *(*s-IND-IPFE-CPA security*) via the security game depicted on Figure 3.4: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left|2 \cdot \Pr[\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1\right|.$$

Then we say that $\mathcal{FE}$ is secure against chosen-plaintext attacks *(*s-IND-IPFE-CPA secure*) if $\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.*

The following theorem states the relation between those two security notions. Basically, if the adversary is restricted in his queries, he can choose randomly beforehand, and abort if it decides it was not the query he wanted. Its advantage will drop by the probability of guessing correctly. This technique is called complexity leveraging, and is widely used when dealing with more complex tools such as functional encryption for all circuits, or indistinguishability obfuscation.

Theorem 3.2.2. *For any IPFE scheme $\mathcal{FE}$ with message space $\mathcal{M}_x$, if $\mathcal{M}_x$ is polynomial in κ and $\mathcal{FE}$ is* s-IND-IPFE-CPA *secure, then it is also* IND-IPFE-CPA *secure. More precisely, for any adversary $\mathscr{A}$, there exist an adversary $\mathscr{B}$ which runs in roughly the same time as $\mathscr{A}$, and such that:*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B}) \geq \frac{1}{|\mathcal{M}_x|^2} \cdot \mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{A})$$

Game $\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize$(\kappa, \ell, \mathbf{x}_0^*, \mathbf{x}_1^*)$	**proc Encrypt**$()$
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_b^*)$
$V_{\mathbf{y}} \leftarrow \emptyset$	Return ct
Return (pp, mpk)	**proc Finalize**(b')
proc KeyDer$(\mathbf{y})$	**if** bad
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{\mathbf{y}\}$	**then return** false
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$	Return $(b' = b)$
Return $sk_{\mathbf{y}}$	

Figure 3.4: Game $\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines s-IND-IPFE-CPA security of $\mathcal{FE}$. The event bad corresponds to $\exists \mathbf{y} \in V_{\mathbf{y}},\ \langle \mathbf{x}_0^*, \mathbf{y}\rangle \neq \langle \mathbf{x}_1^*, \mathbf{y}\rangle$.

Proof. $\mathscr{B}$ just guesses the challenge messages $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ of $\mathscr{A}$ and does the same as $\mathscr{A}$. If $\mathscr{B}$ guessed wrongly, it just aborts and output a uniformly random guess b', otherwise, he is correct when $\mathscr{A}$ is correct. The result follows from the fact that $\mathscr{B}$ guesses correctly with probability $\frac{1}{|\mathcal{M}_x|^2}$. □

This implies that an IPFE scheme with polynomial message space is necessarily IND-IPFE-CPA secure if it is s-IND-IPFE-CPA secure. Moreover, this is true if the message space is independent of, or logarithmic in the security parameter κ. However, this factor really impacts the concrete security of the scheme, which is why adaptive security is an appealing property. It has been shown that if a functional encryption scheme supports all polynomial functions, then one can use the functionality provided by the scheme to build a tightly adaptively secure scheme [ABSV15]. However, this is not as simple in the case of functional encryption limited to inner-product evaluations. We proceed to show a tight (independent of the size of the message space) reduction from IND-IPFE-CPA security to s-IND-IPFE-CPA security, which only works for certains modular IPFE but the same idea will be used in Chapter 4 to build directly IND-IPFE-CPA secure schemes. First, we need to define the additively homomorphic property that will be needed for this transformation.

Definition 3.2.3 (Additive Homomorphism)**.** *We say that an inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is additively homomorphic if it is possible to add encryptions:* $\forall (\mathbf{x}_0, \mathbf{x}_1) \in \mathcal{M}_x^2, (pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$,

$$\mathsf{Encrypt}(pp, mpk, \mathbf{x}_0) + \mathsf{Encrypt}(pp, mpk, \mathbf{x}_1) = \mathsf{Encrypt}(pp, mpk, \mathbf{x}_0 + \mathbf{x}_1)$$

for some operation $+$ *on the ciphertexts.*

Note that we are talking about probability distributions. Moreover, we don't actually need rigorous equality for our reduction as computational inditinguishability would be enough. However, this definition makes the notations much simpler and the proof easier to read.

We now move onto the construction of our IND-IPFE-CPA scheme. We construct an IPFE scheme using another IPFE scheme, so for the sake of readability and simplicity, we add a

prime to elements that corresponds to the new scheme and use notations without primes for the building block.

Construction 3.2.4 (Inner-Product Functional Encryption Secure against Adaptive Chosen–Plaintext Attacks)**.** *Let us assume the existence of a modular IPFE scheme* $\mathcal{FE} = (\mathcal{FE}.\mathsf{Setup}, \mathcal{FE}.\mathsf{KeyDer}, \mathcal{FE}.\mathsf{Encrypt}, \mathcal{FE}.\mathsf{Decrypt})$*. We define our improved IPFE scheme* $\mathcal{FE}'$ *as follows:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp', msk', mpk')$*. On input security parameter* κ *and functionality parameter* ℓ*, samples* $(pp, msk, mpk) \xleftarrow{\$} \mathcal{FE}.\mathsf{Setup}(1^\kappa, 1^{\ell+1})$*, and a random vector* $\mathbf{t} \xleftarrow{\$} \mathcal{R}^\ell$*, and outputs public parameters* $pp' = pp$*, master secret key* $msk' = (msk, \mathbf{t})$*, and master public key* $mpk' = mpk$*;*
- $\mathsf{KeyDer}(pp', msk', \mathbf{y}) \mapsto {sk_\mathbf{y}}'$*. On input public parameter* pp'*, master secret key* msk'*, and key* $\mathbf{y} \in \mathcal{R}^\ell$*, samples* $sk_\mathbf{y} \xleftarrow{\$} \mathcal{FE}\mathsf{KeyDer}(pp, msk, (\mathbf{y}, -\langle \mathbf{t}, \mathbf{y}\rangle))$ *and outputs user secret key* ${sk_\mathbf{y}}' = sk_\mathbf{y}$*;*
- $\mathsf{Encrypt}(pp', mpk', \mathbf{x}) \mapsto {ct_\mathbf{x}}'$*. On input public parameter* pp'*, master public key* mpk'*, and plaintext* $\mathbf{x} \in \mathcal{R}^\ell$*, samples* $ct_\mathbf{x} \xleftarrow{\$} \mathcal{FE}.\mathsf{Encrypt}(pp, mpk, (\mathbf{x}, 0))$ *and outputs ciphertext* ${ct_\mathbf{x}}' = ct_\mathbf{x}$*;*
- $\mathsf{Decrypt}(pp', {sk_\mathbf{y}}', {ct_\mathbf{x}}') \mapsto m$ *or* $\perp$*. On input public parameters* pp'*, user secret key* ${sk_\mathbf{y}}'$*, and ciphertext* ${ct_\mathbf{x}}'$*, computes* $m \leftarrow \mathcal{FE}.\mathsf{Decrypt}(pp, sk_\mathbf{y}, ct_\mathbf{x})$ *and outputs message* m*.*

Correctness. The correctness of $\mathcal{FE}'$ directly follows from the correctness of $\mathcal{FE}$, because $-\langle \mathbf{t}, \mathbf{y}\rangle \cdot 0 = 0$. Notice that we require $\mathcal{R}$ to be finite in order to sample uniform vectors $\mathbf{t}$.

Security. If $\mathcal{FE}$ is s-IND-IPFE-CPA secure, then $\mathcal{FE}'$ is IND-IPFE-CPA secure. More precisely, we have the following theorem.

Theorem 3.2.5. *Let* $\mathcal{FE}$ *and* $\mathcal{FE}'$ *be defined as in Construction 3.2.4. If* $\mathcal{FE}$ *is* s-IND-IPFE-CPA *secure, then* $\mathcal{FE}'$ *is* IND-IPFE-CPA *secure. More precisely, for any adversary* $\mathscr{A}$*, there exists and adversary* $\mathscr{B}$ *which runs in roughly the same time as* $\mathscr{A}$ *and such that:*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B}) = \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE}',\kappa}(\mathscr{A})$$

Proof. We prove this theorem by showing such an adversary $\mathscr{B}$. $\mathscr{B}$ first picks a random $\mathbf{t} \xleftarrow{\$} \mathcal{R}^\ell$, and a random bit $b^* \xleftarrow{\$} \{0, 1\}$. Then, it acts as follows on $\mathscr{A}$'s queries:

- **Initialize** (κ, ℓ): $\mathscr{B}$ calls **Initialize** $(\kappa, \ell, \mathbf{0}, (\mathbf{t}, 1))$;
- **KeyDer**$(\mathbf{y})$: $\mathscr{B}$ appends $-\langle \mathbf{t}, \mathbf{y}\rangle$ to $\mathbf{y}$, and calls **KeyDer**$((\mathbf{y}, -\langle \mathbf{t}, \mathbf{y}\rangle))$;
- **Encrypt**$(\mathbf{x}_0^*, \mathbf{x}_1^*)$: $\mathscr{B}$ calls **Encrypt**$()$ to get ct', and then sets and returns $ct = ct' + \mathcal{FE}.\mathsf{Encrypt}(pp, mpk, (\mathbf{x}_{b^*}, 0))$;
- **Finalize**(b'): $\mathscr{B}$ checks if $\mathscr{A}$ has triggered the bad event, and trigger it in this case. Otherwise, it calls **Finalize**$(b' \neq b^*)$.

Notice that in both experiments $\mathsf{Exp}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B})$ and $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE}',\kappa}(\mathscr{A})$, bad is the same event because $\mathscr{B}$ never triggers it unless $\mathscr{A}$ did: for any $\mathbf{y} \in \mathcal{R}^\ell$, $\langle (\mathbf{t}, 1), (\mathbf{y}, -\langle \mathbf{t}, \mathbf{y}\rangle)\rangle = 0$. We now analyze the advantage of $\mathscr{B}$.

- If $b = 0$, then $ct' = \mathcal{FE}.\mathsf{Encrypt}(pp, mpk, \mathbf{0})$, and $\mathscr{A}$ receives $\mathcal{FE}'.\mathsf{Encrypt}(pp, mpk, \mathbf{x}_b^*)$. This means that in this case, $\mathscr{B}$ wins whenever $\mathscr{A}$ wins.

- Now, if $b = 1$, then $ct' = \mathcal{FE}.\mathsf{Encrypt}(pp, mpk, (\mathbf{t}, 1))$, and thus $\mathscr{A}$ receives $\mathcal{FE}.\mathsf{Encrypt}(pp, mpk, (\mathbf{x}_b^* + \mathbf{t}, 1))$. We want to show that in this case, $\mathscr{A}$ cannot correctly guess b with probability more than $\frac{1}{2}$, because its view is independent of b. Indeed, we show that replacing $\mathbf{t}$ by $\mathbf{t}' = \mathbf{t} + \mathbf{x}_{1-b}^* - \mathbf{x}_b^*$ yields the same keys $sk_{\mathbf{y}}$, while changing the message from $\mathbf{x}_b^*$ to $\mathbf{x}_{1-b}^*$. The important thing to notice is just that for any query $\mathbf{y}$ that doesn't trigger bad:

$$\langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle \implies \langle \mathbf{t}', \mathbf{y} \rangle = \langle \mathbf{t}, \mathbf{y} \rangle.$$

Thus, the advantage of $\mathscr{B}$ is

$$\mathsf{Adv}_{\mathcal{FE},\kappa}^{\mathsf{s\text{-}ind\text{-}ipfe\text{-}cpa}}(\mathscr{B}) = \frac{1}{2} \cdot \mathsf{Adv}_{\mathcal{FE}',\kappa}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}(\mathscr{A}).$$

□

3.2.2 Simulation-Based Security

Another very intuitive way to model the security is what we call simulation-based security. In this setting, in order to prove security, we show that it is possible to generate the same view (ciphertexts and keys), or at least a view that fools any adversary, only knowing the outputs of the functionality, and not the inputs. In our context, this means generating a ciphertext, as well as secret keys without knowing the underlying plaintext vector $\mathbf{x}$, only using the results $\langle \mathbf{x}, \mathbf{y} \rangle$. This is formalized by a simulator that generates a view given the inner-products values, and the adversary tries to tell apart if it's interacting with the simulator, or if it has been given normally generated keys and ciphertext.
Simulation-based security is a very strong notion of security. Its relation to indistinguishability-based security has been well studied [ONe10; AGVW13; DIJ+13], and it has been shown to be impossible to reach in the case of functional encryption for very simple functionalities like IBE [BSW11], and even for any non-trivial functionality [BO13].
It is easy to see that in general, simulation-based security implies indistinguishability-based security, because as the simulator doesn't know the plaintext, it yields the same view whether it was $\mathbf{x}_0^*$ or $\mathbf{x}_1^*$, as long as the adversary only makes allowed queries.

Definition 3.2.6 (Non-Adaptive Simulation-Based Security.)**.** *For an inner-product functional encryption scheme $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined over $\mathcal{R}$, we define* non-adaptive simulation security *(*NA-SIM security*, for short) via the security game depicted on Figure 3.5: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}_{\mathcal{FE},\kappa}^{\mathsf{na\text{-}sim\text{-}ipfe}}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}_{\mathcal{FE},\kappa}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}b}}(\mathscr{A}, \mathscr{S}) = 1] - 1 \right|.$$

Then, we say that $\mathcal{FE}$ is simulation secure against non-adaptive adversaries *(*NA-SIM secure*, for short) if there exists a probabilistic polynomial time simulator $\mathscr{S}$ such that $\mathsf{Adv}_{\mathcal{FE},\kappa}^{\mathsf{na\text{-}sim\text{-}ipfe}}(\mathscr{A}, \mathscr{S})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.*

Game $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$

proc Initialize(κ,ℓ)	**proc Encrypt**$(\mathbf{x}^*)$
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	$ct_0^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}^*)$
$V \leftarrow \emptyset$	$ct_1^* \xleftarrow{\$} \mathscr{S}(mpk, \{\langle \mathbf{x}^*, \mathbf{y}\rangle), \mathbf{y}\}_{\mathbf{y}\in V})$
Return (pp, mpk)	Return ct_b^*
proc KeyDer$(\mathbf{y})$	**proc Finalize**(b')
$V \leftarrow V \cup \{\mathbf{y}\}$	Return $(b = b')$
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$	
Return $sk_{\mathbf{y}}$	

Figure 3.5: Game $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ define NA-SIM security of $\mathcal{FE}$. The procedure **KeyDer** can only be invoked before invoking **Encrypt**.

In this section, we compare the two a-priori incomparable security notions of NA-SIM security and IND-IPFE-CPA security. It is a known result that simulation security implies indistinguishability for any functionality. The next theorem shows that IND-IPFE-CPA security notion lies somewhere between SIM security and NA-SIM security for the inner-product functionality. Although we prove this result from scratch for completeness, we remark that it follows from the work of [ONe10], because the inner-product functionality is preimage sampleable, as noted in [ALS15].

Theorem 3.2.7. *Let $\mathcal{FE}$ be any inner-product functional encryption scheme. If $\mathcal{FE}$ is* IND-IPFE-CPA *secure, then it is* NA-SIM *secure.*

Proof. We prove this statement by exhibiting a simulator $\mathscr{S}$ and showing that if $\mathcal{FE}$ is IND-IPFE-CPA secure, then $\mathscr{S}$ satisfies the properties needed to prove NA-SIM security of $\mathcal{FE}$.
$\mathscr{S}$ is given as input mpk and the set of values $\{\langle \mathbf{x}, \mathbf{y}\rangle, \mathbf{y}\}_{\mathbf{y}\in V}$ for some unknown $\mathbf{x}$. It then finds a vector $\mathbf{x}'$ such that $\langle \mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}', \mathbf{y}\rangle$ for all $\mathbf{y} \in V$ and encrypts it using mpk. $\mathscr{S}$ returns this new formed ciphertext.
If an adversary $\mathscr{A}$ wins Game $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$, it wins Game $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ with challenge messages $\mathbf{x}$ and $\mathbf{x}'$ and without key queries after the challenge.

□

The next question is whether NA-SIM security implies IND-IPFE-CPA security or not. We answer this question in a non-black box manner: We require that the functional encryption scheme supports key delegation in order to prove equivalence of NA-SIM security and IND-IPFE-CPA security.

Definition 3.2.8 (Key delegation)**.** *We say that an inner-product functional encryption scheme $\mathcal{FE}$ supports key delegation if $sk_{\mathbf{y}}$ can be obtained from any set $\{sk_{\mathbf{z}}\}_{\mathbf{z}\in V}$ where $\mathbf{y} \in Span(V)$.*

Note that with this definition, any functional encryption scheme for the inner-product functionality with a deterministic key derivation algorithm supports key delegation.

Theorem 3.2.9. *Let $\mathcal{FE}$ be any inner-product functional encryption scheme. If $\mathcal{FE}$ is* NA-SIM *secure and supports key delegation, then it is* IND-IPFE-CPA *secure.*

Proof. If there weren't any secret key queries after the challenge ciphertext has been received, we could use the same argument as for proving that SIM security implies IND-IPFE-CPA security. This is a simple argument on games defining the security: the experiment $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}0}}_{\mathcal{FE},\kappa}(\mathscr{A})$ with challenges $\mathbf{x}_0$ and $\mathbf{x}_1$ is the same experiment as $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}0}}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ with challenge $\mathbf{x}_0$, which is indistinguishable from the experiment $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}1}}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ with challenge $\mathbf{x}_0$, which is exactly the same as $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}1}}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ with challenge $\mathbf{x}_1$, which is in turn indistinguishable from $\mathsf{Exp}^{\mathsf{na\text{-}sim\text{-}ipfe\text{-}0}}_{\mathcal{FE},\kappa}(\mathscr{A},\mathscr{S})$ with challenge $\mathbf{x}_1$, or equivalently $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}1}}_{\mathcal{FE},\kappa}(\mathscr{A})$ with challenges $\mathbf{x}_0$ and $\mathbf{x}_1$. From which IND-IPFE-CPA security follows.

We now show that given $\mathbf{x}_0$ and $\mathbf{x}_1$, one can compute a basis $\{\mathbf{z}_i\}_{i\in[\ell]}$ of the orthogonal $(\mathbf{x}_1-\mathbf{x}_0)^\perp$. Given secret keys for this basis, key delegation allows to compute a secret key for any $\mathbf{y}$ such that $\langle\mathbf{x}_0,\mathbf{y}\rangle=\langle\mathbf{x}_1,\mathbf{y}\rangle$.

This is easy for modular IPFE schemes where $\mathcal{R}=\mathbb{Z}_q$ because $\mathbb{Z}_q$ is a field. For non-modular IPFE schemes where $\mathcal{R}=\mathbb{Z}$, we want to find a basis $\{\mathbf{z}_i\}_{i\in[\ell-1]}$ of the lattice orthogonal to a given vector $\mathbf{x}$. We construct one recursively:
First, if the gcd of all coordinates of $\mathbf{x}$ is not 1, set $\mathbf{x}=\frac{1}{\gcd(x_i)}\mathbf{x}$.

- If $\ell=2$: $\mathbf{z}=(-x_2,x_1)$ is a basis of $(x_1,x_2)^\perp$
- If $\ell>2$: Let $\{\mathbf{z}_i\}_{i\in[\ell-2]}$ be a basis of $(x_1,\cdots,x_{\ell-1})^\perp$. We set $\mathbf{z}_{\ell-1}=(-x_\ell\times a_1,-x_\ell\times a_2,\ldots,-x_\ell\times a_{\ell-1},\gcd(x_1,\ldots,x_{\ell-1}))$ where the a_i's come from Bezout's Identity.

This basis does not generate a sub-lattice of $\mathbf{x}^\perp$. Indeed, a vector $\mathbf{y}$ orthogonal to $\mathbf{x}$ has a last coordinate which is a multiple of $\gcd(x_1,\ldots,x_{\ell-1})$, because $x_\ell y_\ell=-\sum_{i\in[\ell-1]}x_iy_i$.

To conclude the proof, suppose there is an adversary $\mathscr{A}$ that breaks the IND-IPFE-CPA security. A simulator $\mathscr{S}$ can use this adversary to break the NA-SIM security: Upon reception of $\mathbf{x}_0$ and $\mathbf{x}_1$, $\mathscr{S}$ queries secret keys for the $\mathbf{z}_i$ before asking for a challenge ciphertext, thus avoiding the non-adaptive queries. $\mathscr{S}$ can still answer $\mathscr{A}$'s adaptive queries using key delegation.

□

3.3 Indistinguishability under Chosen-Ciphertext Attack

Although IND-IPFE-CPA security is a very attractive notion, because it provides really strong security of the plaintext vectors, it only protects against passive attacks. In order to be secure against active adversaries, we need to consider a stronger notion of security: IND-IPFE-CCA security [BDPR98; Ble98](In this work, we only consider CCA2 security, not the weaker CCA1 security or security against lunchtime attacks). In this model, the adversary tries to attack a ciphertext while having access to a decryption oracle. IND-CCA security is nowadays the go-to security notion for public key encryption. It is thus very natural to extend it to functional encryption, as has been done in [NP15]. However, this result doesn't directly give any IPFE scheme secure against chosen-ciphertext attacks.
As noted in [BL17], there are multiple possibilities to be considered for this decryption oracle for handling the private keys. However, since our schemes have deterministic key derivation,

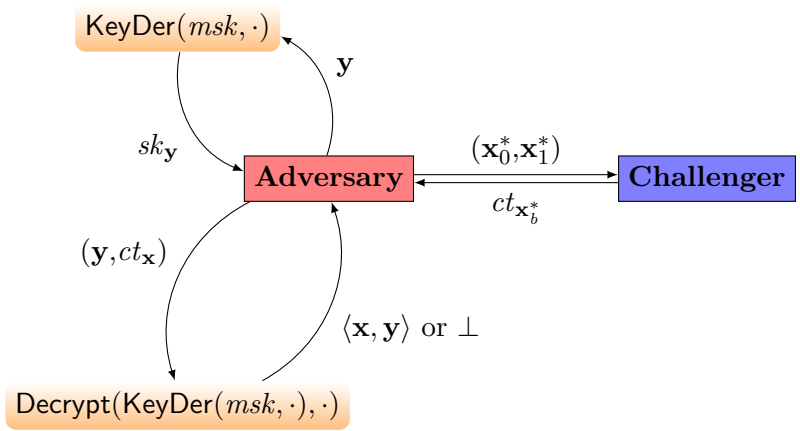

Figure 3.6: Illustration of the IND-IPFE-CCA security game. **Adversary** wins if $(b' = b) \wedge \forall \mathbf{y}, \langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$.

these definitions are equivalent to us. Noticing this, we chose to prove security of our schemes in the one time key model, where a new secret key is generated for each decryption query. We will present a generic construction of IND-IPFE-CCA secure schemes in Chapter 5. In order to lighten the notations in this future construction, we define in this section the notion of *tag-based inner product functional encryption*, and we show how to convert a IND-TBIPFE-CCA secure scheme into a tagless IND-IPFE-CCA secure scheme.

In order to emphasize the difference between IND-IPFE-CCA security and IND-IPFE-CPA security, we present on Figure 3.6 the Figure 3.2 updated with the new decryption oracle.

Definition 3.3.1 (Indistinguishability against Chosen-Ciphertext Attacks)**.** *For an inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *defined over* $\mathcal{R}$*, we define* security against chosen-ciphertext attacks *(*IND-IPFE-CCA security*) via the security game depicted on Figure 3.7: we define the advantage of an adversary* $\mathscr{A}$ *to be*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cca}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cca\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1 \right|.$$

Then we say that $\mathcal{FE}$ *is* secure against chosen-ciphertext attacks *(*IND-IPFE-CCA secure*) if* $\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cca}}_{\mathcal{FE},\kappa}(\mathscr{A})$ *is negligible for all probabilistic polynomial time* $\mathscr{A}$*.*

As in the case of IND-IPFE-CPA security, we only consider the case of a single challenge vector pair $(\mathbf{x}_0^*, \mathbf{x}_1^*)$. This is without loss of generality, because the same hybrid argument proves security in the case of multiple challenge.

Remark 3.3.2. *As in the case of public key encryption, if an inner-product functional encryption scheme is additively homomorphic as in Definition 3.2.3, it cannot be* IND-IPFE-CCA *secure. This is because one can always add an encryption of* $\mathbf{0}$ *to change the ciphertext without changing the plaintext and submit this new ciphertext to the decryption oracle.*

3.3.1 Tag-Based Inner-Product Functional Encryption

Tags are a really useful tool to construct schemes secure against active adversaries. The reason is that this tag can be anything from a hash value to verification keys for signature

Game $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cca\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ)	**proc KeyDer**$(\mathbf{y})$
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{\mathbf{y}\}$
$V_{\mathbf{y}} \leftarrow \emptyset$	$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$
$V_{ct} \leftarrow \emptyset$	Return $sk_{\mathbf{y}}$
Return (pp, mpk)	**proc Encrypt**$(\mathbf{x}_0^*, \mathbf{x}_1^*)$
proc Dec$(ct, \mathbf{y})$	$ct^* \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_b)$
$V_{ct} \leftarrow V_{ct} \cup \{ct\}$	Return ct^*
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$	**proc Finalize**(b')
$m \leftarrow \mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct)$	**if** bad
return m	**then return** false
	Return $(b' = b)$

Figure 3.7: Game $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cca\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines IND-IPFE-CCA security of $\mathcal{FE}$. The event bad corresponds to $\exists \mathbf{y} \in V_{\mathbf{y}},\ \langle \mathbf{x}_0^*, \mathbf{y} \rangle \neq \langle \mathbf{x}_1^*, \mathbf{y} \rangle \vee ct^* \in V_{ct}$.

schemes. This allows the scheme to be protected against an adversary trying to tamper with the ciphertexts. The security in a tag-based scheme is only required as long as tags are differents. This allows for easier analysis and more elegant proofs. A very speaking example is the original IND-CCA secure scheme of Cramer and Shoup [CS98], which use hashes of parts of the ciphertext as tags in order to ensure its authenticity. Our definition is an adaptation from the concept of tag-based encryption in [MRY04].

Basically, tags are bitstrings that we will denote τ and which will be added as inputs to Encrypt and Decrypt. We denote the set of all possible tags $\mathcal{T}$ and assume that it is easy to sample one at random. In order to preserve correctness, the tag τ used to decrypt a ciphertext should match the one that has been used to encrypt it.

Definition 3.3.3 (Tag-Based Inner-Product Functional Encryption)**.** *Formally, a* tag-based inner-product functional encryption *(TBIPFE) scheme $\mathcal{TBFE}$ is defined with regard to a some ring $\mathcal{R}$, and consists of 4 probabilistic polynomial time algorithms:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp, msk, mpk)$. *On input security parameter κ and functionality parameter ℓ outputs public parameters pp, master secret key msk, and master public key mpk;*

- $\mathsf{KeyDer}(pp, msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input public parameter pp, master secret key msk, and key $\mathbf{y} \in \mathcal{R}^\ell$, outputs user secret key $sk_{\mathbf{y}}$;*

- $\mathsf{Encrypt}(pp, \tau, mpk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input public parameter pp, master public key mpk, and plaintext $\mathbf{x} \in \mathcal{R}^\ell$, outputs ciphertext $ct_{\mathbf{x}}$;*

- $\mathsf{Decrypt}(pp, \tau, sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input public parameters pp, user secret key $sk_{\mathbf{y}}$, and ciphertext $ct_{\mathbf{x}}$, outputs a message $m \in \mathcal{R}$ or an error symbol $\perp$.*

Correctness. As in the case of standard inner-product functional encryption, we allow the Decrypt algorithm to output $\perp$ if $\langle \mathbf{x}, \mathbf{y} \rangle$ is outside of a polynomial sized set $\mathcal{P}$ fixed at setup. We also denote the plaintext space $\mathcal{M}_x$ and the key space $\mathcal{M}_y$, and they may be smaller than the full ring $\mathcal{R}$. Hence the following *correctness* property:
A tag-based inner-product functional encryption scheme is correct if for any tag τ, for any $\mathbf{x} \in \mathcal{M}_x, \mathbf{y} \in \mathcal{M}_y$, $(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$, $sk_\mathbf{y} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$, $ct_\mathbf{x} \xleftarrow{\$} \mathsf{Encrypt}(pp, \tau, mpk, \mathbf{x})$. Then if $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{P}$, it holds that $\mathsf{Decrypt}(pp, \tau, sk_\mathbf{y}, ct_\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$.

We also define IND-TBIPFE-CCA security for TBIPFE schemes, which is similar, but the decryption oracle only allows queries that don't have the same tag as the challenge ciphertext.

Definition 3.3.4 (Indistinguishability against Chosen-Ciphertext Attacks)**.** *For a tag-based inner-product functional encryption scheme $\mathcal{TBFE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined over $\mathcal{R}$, we define* security against chosen-ciphertext attacks *(*IND-TBIPFE-CCA security*) via the security game depicted on Figure 3.8: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}^{\mathsf{ind\text{-}tbipfe\text{-}cca}}_{\mathcal{TBFE},\kappa}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}tbipfe\text{-}cca\text{-}b}}_{\mathcal{TBFE},\kappa}(\mathscr{A}) = 1] - 1 \right| .$$

Then we say that $\mathcal{FE}$ is secure against chosen-ciphertext attacks *(*IND-TBIPFE-CCA secure*) if $\mathsf{Adv}^{\mathsf{ind\text{-}tbipfe\text{-}cca}}_{\mathcal{TBFE},\kappa}(\mathscr{A})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.*

Game $\mathsf{Exp}^{\mathsf{ind\text{-}tbipfe\text{-}cca\text{-}b}}_{\mathcal{TBFE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ, τ^*)
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$
$V_\mathbf{y} \leftarrow \emptyset$
$V_{ct} \leftarrow \emptyset$
Return (pp, mpk)

proc Dec$(\tau, ct, \mathbf{y})$
$V_\tau \leftarrow V_\tau \cup \{\tau\}$
$sk_\mathbf{y} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$
$m \leftarrow \mathsf{Decrypt}(pp, \tau, sk_\mathbf{y}, ct)$
return m

proc KeyDer$(\mathbf{y})$
$V_\mathbf{y} \leftarrow V_\mathbf{y} \cup \{\mathbf{y}\}$
$sk_\mathbf{y} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$
Return $sk_\mathbf{y}$

proc Encrypt$(\mathbf{x}_0^*, \mathbf{x}_1^*)$
$ct^* \xleftarrow{\$} \mathsf{Encrypt}(pp, \tau^*, mpk, \mathbf{x}_b)$
Return ct^*

proc Finalize(b')
if bad
  then return false
Return $(b' = b)$

Figure 3.8: Game $\mathsf{Exp}^{\mathsf{ind\text{-}tbipfe\text{-}cca\text{-}b}}_{\mathcal{TBFE},\kappa}(\mathscr{A})$ defines IND-TBIPFE-CCA security of $\mathcal{TBFE}$. The event bad corresponds to $\exists \mathbf{y} \in V_\mathbf{y},\ \langle \mathbf{x}_0^*, \mathbf{y} \rangle \neq \langle \mathbf{x}_1^*, \mathbf{y} \rangle \vee \tau^* \in V_\tau$.

3.3.2 Removing the Tag

In order to get a IND-IPFE-CCA secure inner-product functional encryption scheme from a tag-based one, we use a standard technique following the work of [Kil06] to transform tag-based PKE schemes to IND-CCA secure PKE schemes. The tag is the hash of a fresh verification key for a one-time signature scheme, used to sign the ciphertext. This signature prevents attacks using the malleability of the scheme. This way, if the tag is reused, the ciphertext

cannot be signed, and if the tag isn't reused, the security is guaranteed by the security of the tag-based IPFE. In order to transform our tag-based inner-product functional encryption into a tagless inner-product functional encryption and preserve the security properties, we are going to need: a strongly unforgeable one-time signature (see Definitions 2.2.1 and 2.2.2), as well as a collision resistant hash function (see Definition 2.2.3). We now proceed with our inner-product functional encryption construction.

Construction 3.3.5 (Inner-Product Functional Encryption secure against Chosen-Ciphertext Attacks)**.** *Let us assume the existence of a tag-based inner-product functional encryption scheme* $\mathcal{TBFE} = (\mathcal{TBFE}.\mathsf{Setup}, \mathcal{TBFE}.\mathsf{KeyDer}, \mathcal{TBFE}.\mathsf{Encrypt}, \mathcal{TBFE}.\mathsf{Decrypt})$*, a strongly unforgeable one-time signature* $\mathcal{OTS}$ *and a family of collision resistant hash functions* $(\mathsf{H}_k)_{k\in\mathcal{K}}$*. We define our secret key inner-product functional encryption scheme* $\mathcal{FE}$ *as follows:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp, msk, mpk)$*. On input security parameter* κ *and functionality parameter* ℓ*, samples* $(pp, msk, mpk) \xleftarrow{\$} \mathcal{TBFE}.\mathsf{Setup}(1^\kappa, 1^\ell)$*, a key* $k \xleftarrow{\$} \mathcal{K}$*, and outputs public parameters* $pp' = (pp, k)$*, master secret key* msk*, and master public key* mpk*;*
- $\mathsf{KeyDer}(pp', msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$*. On input public parameter* pp*, master secret key* msk*, and key* $\mathbf{y} \in \mathcal{R}^\ell$*, samples* $sk_{\mathbf{y}} \xleftarrow{\$} \mathcal{TBFE}.\mathsf{KeyDer}(pp, msk, \mathbf{y})$ *and outputs user secret key* $sk_{\mathbf{y}}$*;*
- $\mathsf{Encrypt}(pp', mpk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$*. On input public parameter* pp*, master public key* mpk*, and plaintext* $\mathbf{x} \in \mathcal{R}^\ell$*, samples* $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathcal{OTS}.\mathsf{Setup}(1^\kappa)$*,* $\tau \leftarrow \mathsf{H}_k(\mathsf{vk})$*,* $ct_{\mathbf{x}} \xleftarrow{\$} \mathcal{TBFE}.\mathsf{Encrypt}(pp, \tau, mpk, \mathbf{x})$*, and* $\sigma \xleftarrow{\$} \mathcal{OTS}.\mathsf{Sign}(\mathsf{sk}, ct_{\mathbf{x}})$ *and outputs ciphertext* $ct_{\mathbf{x}}' = (ct_{\mathbf{x}}, \mathsf{vk}, \sigma)$*;*
- $\mathsf{Decrypt}(pp', sk_{\mathbf{y}}, ct_{\mathbf{x}}') \mapsto m$ *or* $\perp$*. On input public parameters* pp*, user secret key* $sk_{\mathbf{y}}$*, and ciphertext* $ct_{\mathbf{x}}'$*, verifies that* $\mathcal{OTS}.\mathsf{Verify}(\mathsf{vk}, ct_{\mathbf{x}})$ *and returns* $\perp$ *if it fails. Otherwise, computes* $\tau \leftarrow \mathsf{H}_k(\mathsf{vk})$ *and* $m \leftarrow \mathcal{FE}.\mathsf{Decrypt}(pp, \tau, sk_{\mathbf{y}}, ct_{\mathbf{x}})$ *and outputs message* m*.*

Correctness. The correctness of $\mathcal{FE}$ directly follows from the correctness of $\mathcal{TBFE}$ and the completeness of $\mathcal{OTS}$. □

Security. We have the following security theorem.

Theorem 3.3.6. *Let* $\mathcal{TBFE}$*,* $\mathcal{OTS}$*,* $(\mathsf{H}_k)_{k\in\mathcal{K}}$*, and* $\mathcal{FE}$ *be defined as in Construction 3.3.5. If* $\mathcal{TBFE}$ *is* IND-TBIPFE-CCA *secure,* $\mathcal{OTS}$ *is strongly unforgeable, and* $(\mathsf{H}_k)_{k\in\mathcal{K}}$ *is collision resistant, then* $\mathcal{FE}$ *is* IND-IPFE-CCA *secure.*

Proof. If for any decryption query $(\mathbf{y}, (ct_{\mathbf{x}}, \mathsf{vk}, \sigma))$ made by the adversary $\mathscr{A}$, $\mathsf{H}(\mathsf{vk}) = \mathsf{H}(\mathsf{vk}^*)$ for the challenge vk^*, and the check $\mathcal{OTS}.\mathsf{Verify}(\mathsf{vk}, ct_{\mathbf{x}}, \sigma)$ passes. Then either $\mathscr{A}$ found a collision of the hash function (if $\mathsf{vk} \neq \mathsf{vk}^*$), or it did a forgery σ for the message $ct_{\mathbf{x}}$. Otherwise, the security game of IND-TBIPFE-CCA security for $\mathcal{TBFE}$ ensures that the advantage of $\mathscr{A}$ is negligible. □

3.4 Function Hiding

Although inner-product functional encryption provides the best possible security for the plaintext vector $\mathbf{x}$, it doesn't protect the key vector $\mathbf{y}$ at all. This however might be desirable for many applications. For example, it prevents an adversary with many keys to learn the

plaintext vector if it doesn't know the actual key vectors associated with its user secret keys. We call this property function hiding. In the following, we show that this property cannot be met in the public key setting, because the adversary can learn the key vectors by encrypting many different plaintexts and decrypting them with its keys. This is why many works have studied the function hiding property in the secret key setting. Function hiding has been previously studied in the case of functional encryption for larger class of functions [BS15], where it is easy to use the functionality provided by the scheme in order to hide the computation allowed by the key. In the particular case of inner-product functional encryption, the first scheme was proposed by [BJK15], but was only reaching a weak form of function hiding. This was later improved to reach the best possible security we can hope for [DDM16; TAO16; KLM+16; KKS17]. All these constructions use the same underlying ideas and require the use of pairings. In this work, we relax a bit the definition of function hiding property to show what we can achieve with modular IPFE schemes without additional assumptions, and show that it implies the standard function hiding security property if the adversary only has access to a bounded number of key queries.

3.4.1 Function Hiding in the Public Key Setting

In this section, we first define the function hiding property in the public key setting before showing its impossibility. The intuition behind the definition is that we adapt the IND-IPFE-CPA security experiment, but now the **KeyDer** procedure takes 2 inputs $\mathbf{y}_0$ and $\mathbf{y}_1$, and it returns $\mathsf{KeyDer}(pp, msk, \mathbf{y}_b)$. Notice that the hidden bit is shared with the **Encrypt** procedure.

Definition 3.4.1 (Function Hiding)**.** *For an inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *defined over* $\mathcal{R}$*, we define* function hiding *via the security game depicted on Figure 3.9: we define the advantage of an adversary* $\mathscr{A}$ *to be*

$$\mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left|2 \cdot \Pr[\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1\right|.$$

Then we say that $\mathcal{FE}$ *is* function hiding *if* $\mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A})$ *is negligible for all probabilistic polynomial time* $\mathscr{A}$*.*

As usual, the event bad aims at removing queries that allows $\mathscr{A}$ to distinguish trivially between $b = 0$ and $b = 1$. However, in this case, it won't be enough, because the restriction is only on the inputs to **Encrypt**, and $\mathscr{A}$ can computes Encrypt directly using mpk.

Theorem 3.4.2. *No public key inner-product functional encryption scheme is function hiding.*

Proof. We exhibit an adversary $\mathscr{A}$ that guesses correctly b for any IPFE $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathscr{A}$ calls **Initialize** $(\kappa, 1)$ to receive pp and mpk.
- $\mathscr{A}$ calls **KeyDer**$(0, 1)$ to receive sk.
- $\mathscr{A}$ computes $ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, 1)$ and learns $b = \mathsf{Decrypt}(pp, ct, sk)$.

Notice that $\mathscr{A}$ didn't trigger bad because $V_{\mathbf{x}}$ is empty. So $\mathscr{A}$ completely breaks the function hiding of $\mathcal{FE}$. □

Game $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ)	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$
$(pp, msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	$V_{\mathbf{x}} \leftarrow V_{\mathbf{x}} \cup \{(\mathbf{x}_0, \mathbf{x}_1)\}$
$V_{\mathbf{y}} \leftarrow \emptyset$	$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, mpk, \mathbf{x}_b)$
$V_{\mathbf{x}} \leftarrow \emptyset$	Return ct
Return (pp, mpk)	
proc KeyDer$(\mathbf{y}_0, \mathbf{y}_1)$	**proc Finalize**(b')
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{(\mathbf{y}_0, \mathbf{y}_1)\}$	**if** bad
$sk \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y}_b)$	**then return** false
Return sk	Return $(b' = b)$

Figure 3.9: Game $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines function hiding of $\mathcal{FE}$. The event bad corresponds to $\exists(\mathbf{y}_0, \mathbf{y}_1) \in V_{\mathbf{y}}$, $\exists(\mathbf{x}_0, \mathbf{x}_1) \in V_{\mathbf{x}}$, $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle \neq \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$.

Due to this impossibility result, in order to continue our discussion on function hiding, we are forced to switch to the context of secret key inner-product funtional encryption.

3.4.2 Secret Key Inner-Product Functional Encryption

This setting is almost the same as the standard IPFE setting we discussed previously, the only difference being that *mpk* doesn't exist anymore, and Encrypt takes as input *msk* instead. We start this section by the formal definition of secret key IPFE and the security model of function hiding, before comparing this new tool with standard modular IPFE.

Definition 3.4.3 (Secret Key Inner-Product Functional Encryption)**.** *Formally, a* secret key inner-product functional encryption *(secret key IPFE) scheme $\mathcal{FE}$ is defined with regard to a some ring $\mathcal{R}$, and consists of 4 probabilistic polynomial time algorithms:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp, msk)$. *On input security parameter κ and functionality parameter ℓ outputs public parameters pp and master secret key msk;*
- $\mathsf{KeyDer}(pp, msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input public parameter pp, master secret key msk, and key $\mathbf{y} \in \mathcal{R}^\ell$, outputs user secret key $sk_{\mathbf{y}}$;*
- $\mathsf{Encrypt}(pp, msk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input public parameter pp, master secret key msk, and plaintext $\mathbf{x} \in \mathcal{R}^\ell$, outputs ciphertext $ct_{\mathbf{x}}$;*
- $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input public parameters pp, user secret key $sk_{\mathbf{y}}$, and ciphertext $ct_{\mathbf{x}}$, outputs a message $m \in \mathcal{R}$ or an error symbol $\perp$.*

Correctness. As in the public key setting, we allow the Decrypt algorithm to output $\perp$ if $\langle \mathbf{x}, \mathbf{y} \rangle$ is outside of a polynomial sized set $\mathcal{P}$ fixed at setup. We also denote the plaintext space $\mathcal{M}_x$ and the key space $\mathcal{M}_y$, and they may be smaller than the full ring $\mathcal{R}$. Hence the following *correctness* property:
An IPFE scheme is correct if for any $\mathbf{x} \in \mathcal{M}_x, \mathbf{y} \in \mathcal{M}_y$, $(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$,

$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$, $ct_{\mathbf{x}} \xleftarrow{\$} \mathsf{Encrypt}(pp, msk, \mathbf{x})$. Then if $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{P}$, it holds that $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) = \langle \mathbf{x}, \mathbf{y} \rangle$.

Note that any IPFE scheme can be seen as a secret key IPFE scheme, by merging the master public key and the master secret key, but not the other way around. The security that we require for secret key IPFE is also stronger than what we ask in the public key setting. For example, we cannot just prove security for one challenge message because the hybrid argument presented in Section 3.1.2 doesn't hold anymore. Hence the following function hiding definition, which is similar to the one in the public key setting:

Definition 3.4.4 (Function Hiding)**.** *For a secret key inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *defined over* $\mathcal{R}$*, we define* function hiding *via the security game depicted on Figure 3.10: we define the advantage of an adversary* $\mathscr{A}$ *to be*

$$\mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left| 2 \cdot \Pr[\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1 \right|.$$

Then we say that $\mathcal{FE}$ *is* function hiding *if* $\mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A})$ *is negligible for all probabilistic polynomial time* $\mathscr{A}$*.*

Game $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ)
$(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$
$V_{\mathbf{y}} \leftarrow \emptyset$
$V_{\mathbf{x}} \leftarrow \emptyset$
Return pp

proc KeyDer$(\mathbf{y}_0, \mathbf{y}_1)$
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{(\mathbf{y}_0, \mathbf{y}_1)\}$
$sk \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y}_b)$
Return sk

proc Encrypt$(\mathbf{x}_0, \mathbf{x}_1)$
$V_{\mathbf{x}} \leftarrow V_{\mathbf{x}} \cup \{(\mathbf{x}_0, \mathbf{x}_1)\}$
$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, msk, \mathbf{x}_b)$
Return ct

proc Finalize(b')
if bad
  then return false
Return $(b' = b)$

Figure 3.10: Game $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines function hiding of $\mathcal{FE}$. The event bad corresponds to $\exists (\mathbf{y}_0, \mathbf{y}_1) \in V_{\mathbf{y}}$, $\exists (\mathbf{x}_0, \mathbf{x}_1) \in V_{\mathbf{x}}$, $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle \neq \langle \mathbf{x}_1, \mathbf{y}_1 \rangle$.

Remark 3.4.5. *If a secret key IPFE scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is function hiding, then its dual* $\mathcal{FE}' = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyDer}, \mathsf{Decrypt})$*, obtained by switching plaintexts and keys is also function hiding, because of the symmetry in the definition. This implies that* $\mathcal{FE}'$ *is* IND-IPFE-CPA *secure, which is a really strong property. In most applications, fewer user secret keys than ciphertexts are used, so the user secret keys shouldn't need the same security as the one provided to ciphertexts. It is with this idea in mind that we now define the partial function hiding property, which means that the leakage obtained when getting multiple user secret keys is equivalent to the linear relations between the keys.*

3.4.3 Partial Function Hiding

For the remainder of this section, we will be focusing on secret key modular IPFE, that is the case where $\mathcal{R} = \mathbb{Z}_p$ for some prime number p. Note that all known constructions of

function hiding IPFE are modular (for small message space). As explained previously, we feel that some applications don't need as strong a security for secret keys as the function hiding property. Notice that only learning the linear equations that the keys satisfy don't allow the adversary to learn the plaintext vector $\mathbf{x}$ of a ciphertext $ct_{\mathbf{x}}$ even given user secret keys for ℓ independent unknown vectors $(\mathbf{y}_i)_{i\in[\ell]}$. This is because in this scenario, the adversary only learns $\mathbf{x}$ up to a change of basis, even if he has multiple ciphertexts.
Moreover, we observe that in some applications, no linearly dependent keys should be given, or the linear combinations don't really have a meaning. For example, if doing machine learning using support vector machine, the elements are embeded as vectors $\mathbf{x}$, and to classify the elements, one simply separates them using another vector $\mathbf{y}$: this separator defines two sets $\mathcal{S}_+ = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{y}\rangle \geq 0\}$ and $\mathcal{S}_- = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{y}\rangle < 0\}$. In this case, adding two vectors $\mathbf{y}$ doesn't really have a meaning as a new separator. For example, if we have two separators that learned to distinguish men from women, and children from adults, adding them doesn't give a separator that distinguishes young boys from grown up women.
We first give the formal definition of this security property, then we show how to obtain such a security property from any public key modular IPFE, using an intermediate tool that we call vector one-time pad, then we will show that partial function hiding implies full function hiding against bounded collusions of adversaries.

We now define our notion of partial function hiding. The only difference with function hiding resides in the definition of the event bad and thus, the set of queries the adversary is allowed in the security game.

Definition 3.4.6 (Partial Function Hiding)**.** *For a secret key inner-product functional encryption scheme $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined over $\mathcal{R}$, we define* partial function hiding *via the security game depicted on Figure 3.11: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}^{\mathsf{pfh\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A}) = \left|2 \cdot \Pr[\mathsf{Exp}^{\mathsf{pfh\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A}) = 1] - 1\right|.$$

Then we say that $\mathcal{FE}$ is partially function hiding *if $\mathsf{Adv}^{\mathsf{pfh\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{A})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.*

Notice that here we broke the symmetry between the secret keys and the ciphertexts. We also show an equivalent definition of bad (the first part stays the same):

$$\begin{aligned}&\exists \lambda_1, \ldots, \lambda_j \in \mathcal{R},\ \exists (\mathbf{y}_0^1, \mathbf{y}_1^1), \ldots, (\mathbf{y}_0^j, \mathbf{y}_1^j) \in V_{\mathbf{y}},\ \sum_{k\in[j]} \lambda_k \mathbf{y}_0^k = 0 \neq \sum_{k\in[j]} \lambda_k \mathbf{y}_1^k \\ \iff\ &\nexists \mathbf{M} \in \mathcal{R}^{\ell\times\ell} \text{ invertible s.t. } \forall (\mathbf{y}_0, \mathbf{y}_1) \in V_{\mathbf{y}},\ \mathbf{y}_0 = \mathbf{M}\mathbf{y}_1\end{aligned}$$

Remark 3.4.7. *We note that any adversary $\mathscr{A}$ that triggers the event bad in this experiment also triggers it in $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}b}}_{\mathcal{FE},\kappa}(\mathscr{A})$. Thus, if a scheme is function hiding, then it is also partially function hiding.*

3.4.3.1 Vector One-Time Pad.

We now present our tool that we call Vector One-Time Pad. It is a toy secret key inner-product functional encryption scheme that is secure as long as it is used only for one key and one plaintext, like the well known one-time pad that is the way to information theoretically

Game $\mathsf{Exp}^{\mathsf{pfh\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$

proc Initialize(κ, ℓ)
$(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$
$V_{\mathbf{y}} \leftarrow \emptyset$
$V_{\mathbf{x}} \leftarrow \emptyset$
Return pp

proc KeyDer$(\mathbf{y}_0, \mathbf{y}_1)$
$V_{\mathbf{y}} \leftarrow V_{\mathbf{y}} \cup \{(\mathbf{y}_0, \mathbf{y}_1)\}$
$sk \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y}_b)$
Return sk

proc Encrypt$(\mathbf{x}_0, \mathbf{x}_1)$
$V_{\mathbf{x}} \leftarrow V_{\mathbf{x}} \cup \{(\mathbf{x}_0, \mathbf{x}_1)\}$
$ct \xleftarrow{\$} \mathsf{Encrypt}(pp, msk, \mathbf{x}_b)$
Return ct

proc Finalize(b')
if bad
  then return false
Return $(b' = b)$

Figure 3.11: Game $\mathsf{Exp}^{\mathsf{pfh\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{A})$ defines function hiding of $\mathcal{FE}$. The event bad corresponds to $\exists(\mathbf{y}_0, \mathbf{y}_1) \in V_{\mathbf{y}}$, $\exists(\mathbf{x}_0, \mathbf{x}_1) \in V_{\mathbf{x}}$, $\langle \mathbf{x}_0, \mathbf{y}_0 \rangle \neq \langle \mathbf{x}_1, \mathbf{y}_1 \rangle \vee \exists \lambda_1, \ldots, \lambda_j \in \mathcal{R}$, $\exists(\mathbf{y}_0^1, \mathbf{y}_1^1), \ldots, (\mathbf{y}_0^j, \mathbf{y}_1^j) \in V_{\mathbf{y}}, \sum_{k\in[j]} \lambda_k \mathbf{y}_0^k = 0 \neq \sum_{k\in[j]} \lambda_k \mathbf{y}_1^k$.

hide a message with a key of same length. Here, the idea is to translate it to vectors by multiplying the plaintext vector by a random invertible matrix $\boldsymbol{B}$. Then, the key will be multiplied by the transpose of the inverse $\boldsymbol{B}^{-t}$ of $\boldsymbol{B}$. Thus, these new vectors will totally hide the plaintext and the key, while their inner-product will keep the same value. Now, looking more closely at this scheme, the leakage coming from reusing the same matrix $\boldsymbol{B}$ for multiple plaintexts $(\mathbf{x}_i)$ will be exactly the linear relations between those vectors. And the same can be said about the key vectors. So the total leakage is the linear relations between plaintexts and keys, as well as the inner-product values between each pair of plaintext and key.

Construction 3.4.8 (Vector One-Time-Pad)**.** *We define our functional encryption scheme for the inner-product functionality* $\mathcal{VOTP} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *as follows:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp, msk)$*. On input security parameter* κ *and functionality parameter* ℓ*, samples a random invertible matrix* $\mathbf{B} \in \mathbb{Z}_p^\ell$ *outputs public parameters* $pp = (p, \ell)$ *and master secret key* $msk = \mathbf{B}$*;*
- $\mathsf{KeyDer}(pp, msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$*. On input public parameter* pp*, master secret key* $msk = \mathbf{B}$*, and key* $\mathbf{y} \in \mathbb{Z}_p^\ell$*, outputs user secret key* $sk_{\mathbf{y}} = \mathbf{B}^{-t}\mathbf{y}$*;*
- $\mathsf{Encrypt}(pp, msk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$*. On input public parameter* pp*, master secret key* $msk = \mathbf{B}$*, and plaintext* $\mathbf{x} \in \mathbb{Z}_p^\ell$*, outputs ciphertext* $ct_{\mathbf{x}} = \mathbf{B}\mathbf{x}$*;*
- $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$*. On input public parameters* pp*, user secret key* $sk_{\mathbf{y}}$*, and ciphertext* $ct_{\mathbf{x}}$*, outputs* $\langle ct_{\mathbf{x}}, sk_{\mathbf{y}} \rangle$*.*

Correctness. For any $\kappa \in \mathbb{Z}$, $\ell \in \mathbb{Z}$, $\mathbf{y} \in \mathbb{Z}_p^\ell$, $\mathbf{x} \in \mathbb{Z}_p^\ell$, let $(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$, $sk_{\mathbf{y}} \leftarrow \mathsf{KeyDer}(pp, msk, \mathbf{y})$, $ct_{\mathbf{x}} \leftarrow \mathsf{Encrypt}(pp, msk, \mathbf{x})$. We have:

$$\langle ct_{\mathbf{x}}, sk_{\mathbf{y}} \rangle = {ct_{\mathbf{x}}}^t sk_{\mathbf{y}} = \mathbf{x}^t \mathbf{B}^t \mathbf{B}^{-t} \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle.$$

□

Remark 3.4.9. *All known function hiding IPFE use the same underlying idea of using a random invertible matrix to rerandomize plaintexts and keys. They do so by putting this new vector in the exponent in a group where the discrete logarithm is hard to compute, thus needing a pairing to compute the inner-product. This technique is also called dual pairing vector spaces [OT08; OT09] This means that at the end of the decryption, a discrete logarithm has to be computed. The plaintext space and the key space should be restricted because of this, but it seems that this is inherent to all constructions based on the discrete logarithm problem. Our way to proceed is to encrypt the resulting vectors with any modular IPFE, thus it could be used with IPFE schemes that supports bigger message spaces. For example the schemes in [ALS16] based on the DCR and LWE assumptions with stateful key derivations algorithms. This is one of the reasons that makes this variant of function hiding appealing.*

3.4.3.2 Generic Construction from Public Key Inner-Product Functional Encryption

We now present our generic construction from public key IPFE to partially function hiding secret key IPFE. The partial function hiding property of the resulting scheme will be inherited from the IND-IPFE-CPA security of the building block, and from the perfect security of the vector one-time pad.

Construction 3.4.10 (Inner-Product Functional Encryption Partially Function Hiding)**.** *Let us assume the existence of a modular IPFE scheme* $\mathcal{FE} = (\mathcal{FE}.\mathsf{Setup}, \mathcal{FE}.\mathsf{KeyDer}, \mathcal{FE}.\mathsf{Encrypt}, \mathcal{FE}.\mathsf{Decrypt})$*. We define our secret key IPFE scheme* $\mathcal{FE}'$ *as follows:*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (pp', msk')$*. On input security parameter* κ *and functionality parameter* ℓ*, samples* $(pp, msk, mpk) \xleftarrow{\$} \mathcal{FE}.\mathsf{Setup}(1^\kappa, 1^\ell)$*, and a random invertible matrix* $\boldsymbol{B} \xleftarrow{\$} \mathcal{R}^{\ell\times\ell}$*, and outputs public parameters* $pp' = pp$ *and master secret key* $msk' = (mpk, msk, \boldsymbol{B})$*;*

- $\mathsf{KeyDer}(pp', msk', \mathbf{y}) \mapsto {sk_\mathbf{y}}'$*. On input public parameter* pp'*, master secret key* msk'*, and key* $\mathbf{y} \in \mathcal{R}^\ell$*, samples* $sk_\mathbf{y} \xleftarrow{\$} \mathcal{FE}.\mathsf{KeyDer}(pp, msk, \boldsymbol{B}^{-t}\mathbf{y})$ *and outputs user secret key* ${sk_\mathbf{y}}' = sk_\mathbf{y}$*;*

- $\mathsf{Encrypt}(pp', msk', \mathbf{x}) \mapsto {ct_\mathbf{x}}'$*. On input public parameter* pp'*, master secret key* msk'*, and plaintext* $\mathbf{x} \in \mathcal{R}^\ell$*, samples* $ct_\mathbf{x} \xleftarrow{\$} \mathcal{FE}.\mathsf{Encrypt}(pp, mpk, \boldsymbol{B}\mathbf{x})$ *and outputs ciphertext* ${ct_\mathbf{x}}' = ct_\mathbf{x}$*;*

- $\mathsf{Decrypt}(pp', {sk_\mathbf{y}}', {ct_\mathbf{x}}') \mapsto m$ *or* $\perp$*. On input public parameters* pp'*, user secret key* ${sk_\mathbf{y}}'$*, and ciphertext* ${ct_\mathbf{x}}'$*, computes* $m \leftarrow \mathcal{FE}.\mathsf{Decrypt}(pp, sk_\mathbf{y}, ct_\mathbf{x})$ *and outputs message* m*.*

Correctness. The correctness of $\mathcal{FE}'$ directly follows from the correctness of $\mathcal{FE}$, because

$$\langle \boldsymbol{B}\mathbf{x}, \boldsymbol{B}^{-t}\mathbf{y}\rangle = \mathbf{x}^t\mathbf{B}^t\mathbf{B}^{-t}\mathbf{y} = \langle \mathbf{x}, \mathbf{y}\rangle.$$

□

Security. If $\mathcal{FE}$ is IND-IPFE-CPA secure, then $\mathcal{FE}'$ is partially function hiding. More precisely, we have the following theorem.

Theorem 3.4.11. *Let $\mathcal{FE}$ and $\mathcal{FE}'$ be defined as in Construction 3.4.10. If $\mathcal{FE}$ is* IND-IPFE-CPA *secure, then $\mathcal{FE}'$ is partially function hiding. More precisely, for any adversary $\mathscr{A}$, there exists and adversary $\mathscr{B}$ which runs in roughly the same time as $\mathscr{A}$ and such that:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}ipfe\text{-}cpa}}_{\mathcal{FE},\kappa}(\mathscr{B}) = \mathsf{Adv}^{\mathsf{pfh\text{-}ipfe}}_{\mathcal{FE}',\kappa}(\mathscr{A})$$

Proof. Unless the event bad is triggered by $\mathscr{A}$, there exists an invertible matrix $\mathbf{M} \in \mathcal{R}^{\ell\times\ell}$ such that $\mathbf{y}_0 = \mathbf{M}\mathbf{y}_1$ for all key queries $(\mathbf{y}_0, \mathbf{y}_1)$. Because the distribution of $\boldsymbol{B}$ is uniform, we can multiply it by any invertible matrix without changing the view of the adversary. In particular, $\mathscr{B}$ can take $\mathscr{A}$'s queries $\{(\mathbf{x}_0, \mathbf{x}_1)\}$ and $\{(\mathbf{y}_0, \mathbf{y}_1)\}$ and replace them by $\{\boldsymbol{B}\mathbf{x}_0, \boldsymbol{B}\mathbf{M}^{-t}\mathbf{x}_1\}$ and $\{\boldsymbol{B}^{-t}\mathbf{y}_0 = \boldsymbol{B}^{-t}\mathbf{M}\mathbf{y}_1\}$ and use those queries in $\mathsf{Exp}^{\mathsf{ind\text{-}ipfe\text{-}cpa\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{B})$. The equality between their advantages follow. □

3.4.3.3 Function Hiding against Bounded Collusion.

In this section, we will compare our weakening on the function hiding with another intuitive weakening: function hiding against bounded collusions. We show that any partially function hiding IPFE scheme can be used to construct an IPFE scheme function hiding against bounded collusion for any bound q_y on the user secret key queries made by the adversary. The idea is to concatenate a random vector at the end of each key vector before deriving the user secret key. We also append zeros to each plaintext vector so that the inner-product will cancel the uniform parts of the key. Those uniform vectors with correctly chosen length will be linearly independent with high probability, in which case the event bad won't be triggered, and we can use the advarsary breaking the function hiding against bounded collusions property in order to break the partial function hiding property.
We let $\mathcal{PH}$ denote a partial function hiding IPFE. We also let q_y be a parameter that will be used as a bound on the secret key queries.

Construction 3.4.12 (Inner-Product Functional Encryption Function Hiding against Bounded Collusions)**.** *Let us assume the existence of a modular secret key IPFE scheme $\mathcal{FE} = (\mathcal{FE}.\mathsf{Setup}, \mathcal{FE}.\mathsf{KeyDer}, \mathcal{FE}.\mathsf{Encrypt}, \mathcal{FE}.\mathsf{Decrypt})$. We define a new secret key IPFE scheme $\mathcal{FE}'$ as follows:*

- $\mathsf{Setup}(1^\kappa, 1^\ell, 1^{q_y}) \mapsto (pp, msk)$. *On input security parameter κ and functionality parameter ℓ, samples $(pp, msk) \xleftarrow{\$} \mathcal{FE}.\mathsf{Setup}(1^\kappa, 1^{\ell+q_y+1})$, and outputs public parameters pp and master secret key msk;*

- $\mathsf{KeyDer}(pp, msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input public parameter pp, master secret key msk, and key $\mathbf{y} \in \mathcal{R}^\ell$, samples a random vector $\mathbf{r} \xleftarrow{\$} \mathcal{R}^{q_y+1}$, and $sk_{\mathbf{y}} \xleftarrow{\$} \mathcal{FE}.\mathsf{KeyDer}(pp, msk, (\mathbf{y}|\mathbf{r}))$, and outputs user secret key $sk_{\mathbf{y}}$;*

- $\mathsf{Encrypt}(pp, msk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input public parameter pp, master secret key msk, and plaintext $\mathbf{x} \in \mathcal{R}^\ell$, samples $ct_{\mathbf{x}} \xleftarrow{\$} \mathcal{FE}.\mathsf{Encrypt}(pp, msk, (\mathbf{x}|\mathbf{0}))$ and outputs ciphertext $ct_{\mathbf{x}}$;*

- $\mathsf{Decrypt}(pp, sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input public parameters pp, user secret key $sk_{\mathbf{y}}$, and ciphertext $ct_{\mathbf{x}}$, outputs $\mathcal{FE}.\mathsf{Decrypt}(pp_1, sk_{\mathbf{y}}, ct_{\mathbf{x}})$.*

Correctness. The correctness of $\mathcal{FE}'$ follows the one of $\mathcal{FE}$ by construction: for any $\kappa \in \mathbb{Z}$, $\ell \in \mathbb{Z}$, $\mathbf{y} \in \mathcal{R}^\ell$, $\mathbf{x} \in \mathcal{R}^\ell$, let $(pp, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$, $sk_\mathbf{y} \xleftarrow{\$} \mathsf{KeyDer}(pp, msk, \mathbf{y})$, $ct_\mathbf{x} \xleftarrow{\$} \mathsf{Encrypt}(pp, msk, \mathbf{x})$. We have:

$$\mathsf{Decrypt}(pp, sk_\mathbf{y}, ct_\mathbf{x}) = \mathcal{FE}.\mathsf{Decrypt}(pp, sk_\mathbf{y}, ct_\mathbf{x}) = \langle (\mathbf{x}|\mathbf{0}), (\mathbf{y}|\mathbf{r}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle.$$

□

Security. We have the following security theorem:

Theorem 3.4.13. *Let $\mathcal{FE}$ and $\mathcal{FE}'$ be defined as in Construction 3.4.12. If $\mathcal{FE}$ is partially function hiding, then $\mathcal{FE}'$ is function hiding against bounded collusions. More precisely, for any adversary $\mathscr{A}$ that makes at most q_y queries to **KeyDer**, there exists an adversary $\mathscr{B}$ which runs in roughly the same time as $\mathscr{A}$ and such that:*

$$\mathsf{Adv}^{\mathsf{pfh\text{-}ipfe}}_{\mathcal{FE},\kappa}(\mathscr{B}) = (1 - \frac{1}{p}) \cdot \mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE}',\kappa}(\mathscr{A}).$$

We recall that p is the order of the message space.

Proof. Since the games are exactly the same, except the allowed queries, we only need to look at the probabilities of the two events bad in $\mathsf{Exp}^{\mathsf{pfh\text{-}ipfe\text{-}}b}_{\mathcal{FE},\kappa}(\mathscr{B})$ and $\mathsf{Exp}^{\mathsf{function\text{-}hiding\text{-}ipfe\text{-}}b}_{\mathcal{FE}',\kappa}(\mathscr{A})$ that we will denote $\mathsf{bad}_{\mathcal{FE}}$ and $\mathsf{bad}_{\mathcal{FE}'}$ It is easy to see that if $\mathscr{A}$ triggers bad, then $\mathscr{B}$ also does. That is, if a query is forbidden in the function hiding security game, then it is also forbidden in the partial function hiding security game. We now assume without loss of generality that $\mathscr{A}$ makes q_y queries. Using an union bound, we can get the following bound on the probability $P_{fail} = 1 - P_{ind}$ that the vectors $(\mathbf{r}_i)_{i \in [q_y]}$ are linearly dependent:

$$P_{fail} \leq \sum_{i=0}^{q_y} \frac{p^i}{p^{q_y+1}} \leq \frac{1}{p^{q_y+1}} \cdot \frac{1 - p^{q_y}}{1 - p} \leq \frac{1}{p}$$

Thus,

$$\Pr[\mathsf{bad}_{\mathcal{FE}'}] \leq (1 - \frac{1}{p}) \cdot \Pr[\mathsf{bad}_{\mathcal{FE}}],$$

and the adversary $\mathscr{B}$ has advantage $(1 - \frac{1}{p}) \cdot \mathsf{Adv}^{\mathsf{function\text{-}hiding\text{-}ipfe}}_{\mathcal{FE}',\kappa}(\mathscr{A})$ against the security of $\mathcal{FE}'$.

□

Remark 3.4.14. *Even though we only look at bounded collusions, if an adversary makes more queries, the IND-IPFE-CPA security will still protect the plaintexts, and the scheme will still be partially function hiding. That is because we are using directly the previous scheme that has these security properties against unbounded collusions that we inherit.*

Chapter 4

Generic Construction from Public Key Encryption

The goal of this chapter is to generically build an inner-product functional encryption scheme $\mathcal{FE}$ from a basic public key encryption scheme $\mathcal{E}$. It combines the generic construction we have done in [ABDP15b] and [ABCP16].
We start by explaining the ideas behind the construction, and the proof of security.
Afterwards, we will define additional properties that the public key encryption scheme must have in order to continue with our construction: structural properties, but also homomorphic properties and security properties.
Then, we construct s-IND-IPFE-CPA secure inner-product functional encryption, and prove its correctness and security using the properties of $\mathcal{E}$.
Finally, we construct directly IND-IPFE-CPA secure schemes without requiring the generic techniques presented in Section 3.2.1, thus having a tighter reduction to the underlying assumption, and we explain the downside of this construction: it requires bigger parameters, which makes it slower, and sometimes rely on a stronger assumption, depending on the underlying scheme.

Contents

4.1 Overview **42**
4.2 Additional Properties of Public Key Encryption **43**
4.2.1 Structural Properties 43
4.2.2 Homomorphic Properties 44
4.2.3 Security Properties 45
4.3 Generic Selectively Secure Inner-Product Functional Encryption . **46**
4.3.1 Construction and Correctness 46
4.3.2 A Simpler Case: Randomness Reuse 47
4.3.3 Security in the General Case 51
4.4 Generic Adaptively Secure Inner-Product Functional Encryption . **56**
4.4.1 Construction and Correctness 56
4.4.2 Adaptive Security 58

4.1 Overview

Before going into more details, we first give a high-level overview of the intuition behind the construction, and the ideas behind the proof of security for the simpler s-IND-IPFE-CPA security, as well as how to reach full IND-IPFE-CPA security.

Overview of the Construction. In order to build an inner-product functional encryption scheme, the first observation is the following: in order to compute an inner-product between two vectors $\mathbf{x}$ and $\mathbf{y}$ and knowing $\mathbf{y}$, it is enough to do a weighted sum of the coordinates of $\mathbf{x}$. So intuitively, if $\mathbf{x}$ is encrypted with a homomorphically additive scheme, we could derive an encryption of $\langle \mathbf{x}, \mathbf{y} \rangle$ using this property.
However, this is not enough for our purpose, because we need to give a key that decrypts this new ciphertext. Moreover, we don't want this key to allow decryption of any other ciphertext. So, we need this additive homomorphism to change the secret key at the same time as the plaintext. Usually, when dealing with additively homomorphic public key encryption schemes, adding the ciphertexts usually adds up the plaintext as well as the randomness used for encryption, which is good because usually, we want the key to stay the same, and the randomness doesn't really matter. Here we are going to use a dual approach: we will see the randomness used to encrypt as a secret key, and the secret key as the encryption randomness. So in the end, we encrypt each coordinate of the vector $\mathbf{x}$ under the same randomness, and a different key. This will ensure that when adding the different parts of the ciphertext to compute $\langle \mathbf{x}, \mathbf{y} \rangle$, the key will also be changed to $\langle sk, \mathbf{y} \rangle$.

Selective Security. The trick that makes this construction secure against collusion is that the leakage of information about the secret key is exactly the same as the leakage produced by the functionality, so combining user secret keys yields exactly the same result as combining results once decrypted. In fact, this means that our scheme supports key delegation as per Definition 3.2.8.
The fact that our scheme allows to delegate user secret keys is very useful in order to prove its selective security. Once the challenge vectors $\mathbf{x}_0^*$ and $\mathbf{x}_1^*$ are set, the adversary is only allowed key queries $\mathbf{y}$ in the set $(\mathbf{x}_1^* - \mathbf{x}_0^*)^\perp$. This is crucial in the proof, because then all key queries can be simulated only knowing a basis of this set, which is either a vector space in the case of $\mathcal{R} = \mathbb{Z}_q$, or a lattice in the case of $\mathcal{R} = \mathbb{Z}$. Then it is left to show that we can simulate all the adversary's view only using a challenge public key and ciphertext from the public key encryption, plus the user secret keys for $(\mathbf{x}_1^* - \mathbf{x}_0^*)^\perp$.

Adaptive Security. In order to reach full adaptive security for our schemes, we use the technique described in Section 3.2.1. However, this doesn't directly apply to non-modular IPFE schemes. In order to reach IND-IPFE-CPA security for all IPFE schemes, we use the same technique of sampling a random vector $\mathbf{t}$ to hide the message which will be stored in the master secret key. The downside is that we cannot use a uniform vector $\mathbf{t}$ in $\mathcal{R}^\ell$ to mask the plaintext $\mathbf{x}$, so we have to sample $\mathbf{t}$ in a set superpolynomially bigger than $\mathcal{M}_x$ in order to statistically hide $\mathbf{x}$.
The underlying idea behind this transformation is to use a specific challenge message in the proof: $(\mathbf{t}, 0)$ against $(0, 0)$. In order for the proof to follow, we just require $\mathbf{x} + \mathbf{t}$ to hide $\mathbf{x}$, Hence the choice of $\mathcal{M}_t$.

4.2 Additional Properties of Public Key Encryption

Even if our generic constructions can be instantiated using many known public key encryption schemes, it doesn't work with all of them. We require a few additional properties, structural and homomorphic properties for the correctness of the inner-product functional encryption as well as security properties in order to protect the vector against collusions.

4.2.1 Structural Properties

We now define the structure that we need on $\mathcal{E}$, its secret and public keys, its plaintexts and ciphertexts, and its Setup algorithm. These structural properties are mostly rewriting to explicit the randomness sampled by the algorithms in order to present the homomorphic properties we require more easily. The parameter τ that we introduce in the Setup algorithm is useful for schemes that allows multiple public keys to correspond to the same underlying secret key. For example, it will be useful for schemes based on the learning with error assumption that we will use in Chapter 8, where the public key is noisy: a small error is added to the correct computation to prevent adversaries to revert the secret key to public key computation. In this case, τ could be the parameters that sets this error distribution.

- The secret keys are elements of a group $(G, +, 0_G)$, for which we use an additive notation. We note that this group doesn't have to be public, as our proof only requires the operation $+$ to be publicly available and efficiently computable.
- The public keys are elements of a group $(H, \cdot, 1_H)$, for which we use a multiplicative notation.
- The message space is $\mathcal{R}$ or a subset of it. We require that the order of $\mathcal{R}$ is greater than $\|\mathbf{x}_1 - \mathbf{x}_0\|^2$ for all $(\mathbf{x}_0, \mathbf{x}_1) \in \mathcal{M}_x^2$.
- The ciphertexts can be splitted in two parts:
  1. The first part ct_0 corresponds to some *commitment* $\mathsf{C}(r)$ of the randomness r used for the encryption.
  2. The second part ct_1 is the *encryption* $\mathsf{E}(pk, x; r)$ of the message x under public key pk and randomness r. We require that this part is an element of a group $(I, \cdot, 1_I)$ for which we use the multiplicative notation.
- The Setup can also be splitted in two parts:
  1. $\mathsf{SKGen}(1^\kappa) \mapsto sk$. On input the security parameter κ samples and returns a secret key sk from the secret key space according to the same distribution induced by Setup.
  2. $\mathsf{PKGen}(sk, \tau) \mapsto pk$. On input a secret key sk and parameters τ, generates a public key pk corresponding to sk according to the distribution induced by τ. We will omit τ when it is clear from the context.

Now that we defined the structure of $\mathcal{E}$, we can define the homomorphic properties that we will use in order to prove the correctness of our inner-product functional encryption scheme.

4.2.2 Homomorphic Properties

In order for the decryption to be correct, we require two homomorphic properties. One on the public and secret keys of the scheme, as well as one on the second part of the ciphertexts E. The first one more or less states that PKGen is a group homomorphism from G to H. This would be exact if the parameter τ wasn't there. Instead, it states that a linear combination of secret keys is functional with regard to the same linear combination of corresponding public keys.

Definition 4.2.1 (Linear Key Homomorphism.)**.** *We say that a PKE has* linear key homomorphism *(LKH, for short) if the following is true: for any two secret keys $sk_1, sk_2 \in G$ and any $y_1, y_2 \in \mathbb{Z}$, the component-wise G-linear combination formed by $y_1 sk_1 + y_2 sk_2$ can be computed efficiently only using public parameters, the secret keys sk_1 and sk_2 and the coefficients y_1 and y_2.*
Moreover, this combination $y_1 sk_1 + y_2 sk_2$ also functions as a secret key for a public key that can be computed as $pk_1^{y_1} \cdot pk_2^{y_2}$, where pk_1 (resp. pk_2) is a public key corresponding to sk_1 (resp. sk_2): $\forall x \in \mathcal{M}_x$, $(sk_1, pk_1), (sk_2, pk_2) \xleftarrow{\$} \mathsf{Setup}(1^\ell)$, $y_1, y_2 \in \mathbb{Z}$,

$$\mathsf{Decrypt}(y_1 sk_1 + y_2 sk_2, \mathsf{Encrypt}(pk_1^{y_1} \cdot pk_2^{y_2}, x)) = x.$$

The second homomorphic property that we need for the correctness of our construction states that the second part E of the $\mathsf{Encrypt}$ algorithm is a group homomorphism from $G \times \mathcal{R}$ to I for a fixed randomness r. That is, when multiplying two encryptions with same randomness, both the plaintexts and the public key undergo the same linear combination.

Definition 4.2.2 (Linear Ciphertext Homomorphism under Shared Randomness.)**.** *We say that a PKE has* linear ciphertext homomorphism under shared randomness *(LCH, for short) if for any plaintexts $x_1, x_2 \in \mathcal{M}_x$, any public keys $pk_1, pk_2 \in H$, and any randomness r, it holds that*

$$\mathsf{E}(pk_1, x_1; r) \cdot \mathsf{E}(pk_2, x_2; r) = \mathsf{E}(pk_1 \cdot pk_2, x_1 + x_2; r).$$

Note the contrast with Definition 3.2.3: Here we fix the randomness, and both the public key and message change: the public keys get multiplied whereas the messages get added. In the previous case, the public key was fixed, while the messages were added. This is crucial for the security of our scheme, because if we were not changing the public key, a secret key allowing decryption would leak too much information about the plaintext. Our setting is close to the one of multi-key fully homomorphic encryption [CM15; MW16; BP16; CO17], which supports any computation on the ciphertexts, while the secret key used for decryption changes during computation, or to the one of fully key-homomorphic ABE [BGG+14].

Remark 4.2.3. *Combining both definitions, if a public key encryption scheme has both linear key homomorphism and linear ciphertext homomorphism under shared randomness, it is possible to encrypt two messages x_1 and x_2 using the same randomness and different public keys pk_1 and pk_2 corresponding to sk_1 and sk_2, multiply the encryptions $\mathsf{E}(pk_1, x_1; r)$ and $\mathsf{E}(pk_2, x_2; r)$ and decrypt it using $sk_1 + sk_2$ to recover $x_1 + x_2$. We note that the plaintexts and the secret keys follow exactly the same computation.*

We now move on to the definitions of the security properties we need beside $\mathsf{s\text{-}IND\text{-}CPA}$ security for $\mathcal{E}$. They closely follow the homomorphic properties defined previously, but are not directly implied in the general case.

4.2.3 Security Properties

In order to prove security of our inner-product functional encryption scheme, we need the following two additional security properties on our public key encryption scheme. The first one is closely related to the linear key homomorphism. Indeed, it states that it is hard to distinguish between a public key that has been freshly generated by PKGen, and a key that has been generated using the homomorphism described in Definition 4.2.1. Of course, for noisy schemes such as lattice-based ones, the distribution will not exactly be the same, so we allow the parameters τ used in the two different world to be different.

Definition 4.2.4 (ℓ-Public-Key-Reproducibility.)**.** *For a public-key encryption scheme $\mathcal{E}$ we define* ℓ-public-key-reproducibility *via the security game depicted on Figure 4.1: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}}_{\mathcal{E},\kappa}(\mathscr{A}) = \left|2 \cdot \Pr[\mathsf{Exp}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\kappa}(\mathscr{A})] - 1\right|$$

Then, we say that $\mathcal{E}$ has ℓ-public-key-reproducibility if there exists $\tau, \tau', (\tau_i)_{i\in[\ell]}$ such that $\mathsf{Adv}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}}_{\mathcal{E},\kappa}(\mathscr{A})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.

Game $\mathsf{Exp}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\kappa}(\mathscr{A})$

proc Initialize$(\kappa, \mathcal{M})$

$(sk, (\alpha_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{D}(1^\kappa)$
if $b = 0$
  then $(pk_i = \mathcal{E}.\mathsf{PKGen}(\alpha_i sk + sk_i, \tau))_{i\in[\ell]}$
  else $pk \leftarrow \mathcal{E}.\mathsf{PKGen}(sk, \tau')$
    $(pk_i \leftarrow pk^{\alpha_i} \cdot \mathcal{E}.\mathsf{PKGen}(sk_i, \tau_i))_{i\in[\ell]}$
Return $(pk_i, sk_i)_{i\in[\ell]}$

proc Finalize(b')

Return $(b' = b)$

Figure 4.1: Game $\mathsf{Exp}^{\ell\text{-}\mathsf{pk}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\kappa}(\mathscr{A})$ defines ℓ-public-key-reproducibility of $\mathcal{E}$. $\mathcal{D}$ samples tuples of the form $(sk, (\alpha_i, sk_i)_{i\in[\ell]})$ where sk and the sk_i's are sampled from SKGen, and the α_i's are in $\mathcal{M}$.

The second additional security property on $\mathcal{E}$ is related to the linear ciphertext homomorphism under shared randomness. Basically, we want it to be hard for the adversary to dinstinguish between a ciphertext generated by homomorphism, and a fresh ciphertext. But in order to use the LCH property of Definition 4.2.2, we need to be able to generate second parts of ciphertexts using only the first part $\mathsf{C}(r)$ of a ciphertext and secret keys sk.

Definition 4.2.5 (ℓ-Ciphertext-Reproducibility.)**.** *For a public-key encryption scheme $\mathcal{E}$ we define* ℓ-ciphertext-reproducibility *via the security game depicted on Figure 4.2: we define the advantage of an adversary $\mathscr{A}$ to be*

$$\mathsf{Adv}^{\ell\text{-}\mathsf{ct}\text{-}\mathsf{rep}}_{\mathcal{E},\kappa}(\mathscr{A}) = \left|2 \cdot \Pr[\mathsf{Exp}^{\ell\text{-}\mathsf{ct}\text{-}\mathsf{rep}\text{-}\mathsf{b}}_{\mathcal{E},\kappa}(\mathscr{A})] - 1\right|$$

Then, we say that $\mathcal{E}$ has ℓ-ciphertext-reproducibility if there exists τ', τ_i's and algorithm E' such that $\mathsf{Adv}^{\ell\text{-ct-rep}}_{\mathcal{E},\kappa}(\mathscr{A})$ is negligible for all probabilistic polynomial time $\mathscr{A}$.

Game $\mathsf{Exp}^{\ell\text{-ct-rep-b}}_{\mathcal{E},\kappa}(\mathscr{A})$

proc Initialize$(\kappa, \mathcal{M})$

$(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{D}(1^\kappa)$
$(sk, pk) \xleftarrow{\$} \mathcal{E}.\mathsf{Setup}(1^\kappa)$
$(pk_i \xleftarrow{\$} \mathcal{E}.\mathsf{PKGen}(sk_i, \tau_i))_{i\in[\ell]}$
$ct_0 \leftarrow \mathcal{E}.\mathsf{C}(r)$
$ct \leftarrow \mathcal{E}.\mathsf{E}(pk, a; r)$
if $b = 0$
  then $ct_i \leftarrow ct^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(pk_i, x_i; r)$
  else $ct_i \leftarrow ct^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}'(sk_i, x_i, ct_0, \tau_i)$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]})$

proc Finalize(b')

Return $(b' = b)$

Figure 4.2: Game $\mathsf{Exp}^{\ell\text{-ct-rep-b}}_{\mathcal{E},\kappa}(\mathscr{A})$ defines ℓ-ciphertext-reproducibility of $\mathcal{E}$. $\mathcal{D}$ samples tuples of the form $(a, (\alpha_i, x_i, , sk_i)_{i\in[\ell]})$, where sk_i's are sampled from SKGen, α_i's are in $\mathcal{M}$ and a and the x_i's are in $\mathcal{M}_x$. E' is an algorithm that takes as input a secret key in G, a message in $\mathcal{R}$, a first part ciphertext $\mathsf{C}(r)$ for some r in the randomness space, and the parameters needed to generate public keys, and output a second part ciphertext.

We are now ready to show our construction of inner-product functional encryption.

4.3 Generic Selectively Secure Inner-Product Functional Encryption

We will first define our IPFE scheme and show its correctness, then we start with a proof of security in a simpler case, where the underlying public key encryption scheme has a property that we call randomness reuse. And then we move on to the full proof of security.

4.3.1 Construction and Correctness

Our construction is very simple. The basic idea is to use ℓ independent key pairs to encrypt individually the ℓ elements of the vector $\mathbf{x}$, and use the homomorphic properties described in Section 4.2.2 in order to compute and decrypt the inner-product with a given vector $\mathbf{y}$.

Construction 4.3.1 (Selectively Secure Inner-Product Functional Encryption from Public Key Encryption)**.** *Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a PKE scheme with the properties defined previously, we define our inner-product functional encryption scheme $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows.*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (msk, mpk)$. *On input security parameter κ and functionality parameter ℓ, generates ℓ independent key pairs*

$$\forall i \in [\ell],\ (sk_i, pk_i) \xleftarrow{\$} \mathcal{E}.\mathsf{Setup}(1^\kappa)$$

sharing the same public parameters pp, and returns master secret key $msk = (sk_1, \ldots, sk_\ell)$ and master public key $mpk = (pk_1, \ldots, pk_\ell)$;

- $\mathsf{KeyDer}(msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input master secret key msk, and key $\mathbf{y} \in \mathcal{M}_y$, outputs user secret key*

$$sk_{\mathbf{y}} \leftarrow \sum_{i \in [\ell]} y_i sk_i;$$

- $\mathsf{Encrypt}(mpk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input master public key mpk, and plaintext $\mathbf{x} \in \mathcal{M}_x$, samples shared randomness r in the randomness space of $\mathcal{E}$, and computes*

$$ct_0 \leftarrow \mathcal{E}.\mathsf{C}(r),$$
$$ct_i \leftarrow \mathcal{E}.\mathsf{E}(pk_i, x_i; r),$$

and outputs ciphertext $ct_{\mathbf{x}} = (ct_0, (ct_i)_{i \in [\ell]})$;

- $\mathsf{Decrypt}(sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input user secret key $sk_{\mathbf{y}}$, and ciphertext $ct_{\mathbf{x}}$, returns the output of*

$$\mathcal{E}.\mathsf{Decrypt}(sk_{\mathbf{y}}, (ct_0, \prod_{i \in [\ell]} ct_i^{y_i})).$$

The correctness of our scheme directly follows from the homomorphic properties defined in Section 4.2.2 and the correctness of the public key encryption scheme $\mathcal{E}$.

Correctness. For all $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$, all $\mathbf{y} \in \mathcal{M}_y$ and $\mathbf{x} \in \mathcal{M}_x$, for $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ and $ct \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x})$, we have that

$$\begin{aligned}
\mathsf{Decrypt}(mpk, ct, sk_{\mathbf{y}}) &= \mathcal{E}.\mathsf{Decrypt}(sk_{\mathbf{y}}, (ct_0, \prod_{i \in [\ell]} ct_i^{y_i})) \\
&= \mathcal{E}.\mathsf{Decrypt}(sk_{\mathbf{y}}, (ct_0, \prod_{i \in [\ell]} \mathcal{E}.\mathsf{E}(pk_i, x_i; r)^{y_i})) \\
&= \mathcal{E}.\mathsf{Decrypt}(sk_{\mathbf{y}}, (ct_0, \mathcal{E}.\mathsf{Encrypt}(\prod_{i \in [\ell]} pk_i^{y_i}, \sum_{i \in [\ell]} y_i x_i; r))) \\
&= \sum_{i \in [\ell]} y_i x_i\ .
\end{aligned}$$

by the LCH property. Finally, note that the decryption is allowed because $(sk_{\mathbf{y}}, \prod_{i \in [\ell]} pk_i^{y_i})$ is a valid key pair, due to the LKH property.

4.3.2 A Simpler Case: Randomness Reuse

As we want our constrution to be generic enough to encompass some lattice-based public key encryption schemes as building block, the proof might seem a bit complicated because we have to deal with a lot of approximate properties. However, in the special case where we

have randomness reuse, which is basically the perfect version of ciphertext reproducibility, there is a very simple and elegant proof.
Randomness reuse is also called *reproducibility* in [BBS03] and guarantees that it is secure to reuse randomness when encrypting under several independent public keys. The idea was first considered by [Kur02]. It is illustrated on Figure 4.3 and defined as follows.

Definition 4.3.2 (Randomness Reuse.)**.** *We say that a PKE has* randomness reuse *(RR, for short) if* $\mathsf{E}(pk, x; r)$ *is* efficiently computed *given the triple* (x, pk, r)*, or the triple* $(x, sk, \mathsf{C}(r))$ *where* sk *is a secret key corresponding to* pk*.*

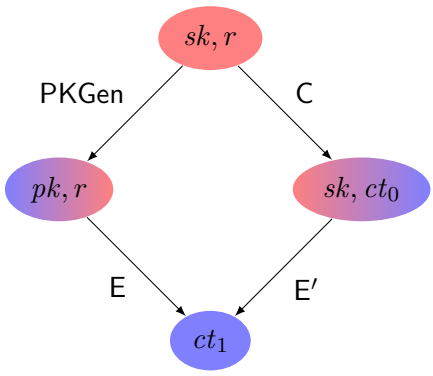

Figure 4.3: Illustration for randomness reuse. There are two paths to compute the second part of the ciphertext ct_1. The left path is used in the real scheme: the encryption randomness is used together with the public key. The right path is only used in the proof: the encryption randomness is not needed, it is sufficient to have the first part of the ciphertext $ct_0 = \mathsf{C}(r)$ and the secret key.

This property is enough to replace both security properties defined in Section 4.2.3, hence the following theorem.

Theorem 4.3.3. *Let* $\mathcal{E}$ *be a public-key encryption scheme with the structural properties defined in Section 4.2.1, and let* $\mathcal{FE}$ *be the functional encryption scheme for the inner-product functionality obtained by applying Construction 4.3.1 to* $\mathcal{E}$*. If* $\mathcal{E}$ *is* s-IND-CPA*, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, and remains secure under randomness-reuse then* $\mathcal{FE}$ *is* s-IND-IPFE-CPA*.*

Remark 4.3.4. *For our security proof to work, it must hold that* $\|\mathbf{x}_1 - \mathbf{x}_0\|^2 \neq 0$*, where* $\mathbf{x}_0, \mathbf{x}_1$ *are the challenge messages. The lower bound on the order of the message space ensures exactly this. We note that in some cases, like the instantiation based on DDH, a direct proof would avoid this issue.*

Proof. To prove the security of our scheme we will show that the s-IND-IPFE-CPA game is indistinguishable from a game where the challenge ciphertext encrypts a uniformly random linear combination of the challenge messages whose coefficients sum up to one. Thus, the challenge ciphertext decrypts to the expected values and information theoretically hides the challenge bit.

Given an adversary $\mathscr{A}$ that breaks the s-IND-IPFE-CPA security of our $\mathcal{FE}$ scheme with non-negligible probability ε, we construct an adversary $\mathscr{B}$ that breaks the s-IND-CPA security of the underlying PKE scheme $\mathcal{E}$ with comparable probability.

$\mathscr{B}$ starts by picking a random element a in the full message space of the underlying PKE $\mathcal{E}$, and sends challenge messages 0 and a to the challenger $\mathscr{C}$ of PKE security game. $\mathscr{C}$ answers by sending an encryption $ct = (ct_0, ct_1)$ of either 0 or a and public key pk.

$\mathscr{B}$ then invokes $\mathscr{A}$ on input the security parameter and gets two different challenge messages in output, namely $(\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,\ell}))_{i\in\{0,1\}}$ both in M.

Recall that, by the constraints of security game, the adversary can only issue secret key queries for vectors $\mathbf{y}$ such that $\langle \mathbf{x}_0, \mathbf{y}\rangle = \langle \mathbf{x}_1, \mathbf{y}\rangle$. Thus, we have that $\langle \mathbf{y}, \mathbf{x}_1 - \mathbf{x}_0\rangle = 0$ meaning that $\mathbf{y}$ is in the vector space defined by $(\mathbf{x}_1 - \mathbf{x}_0)^{\perp}$.

Then, $\mathscr{B}$ generates the view for $\mathscr{A}$ in the following way:

Public Key. To generate master public key mpk, $\mathscr{B}$ does the following. First, $\mathscr{B}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^{\perp}$. Then we can write the canonical vectors in the basis $((\mathbf{x}_1 - \mathbf{x}_0), \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathcal{R}$ and $\alpha_i \in \mathcal{R}$ such that:

$$\mathbf{e}_i = \alpha_i(\mathbf{x}_1 - \mathbf{x}_0) + \sum_{j\in[\ell-1]} \lambda_{i,j}\mathbf{z}_j. \tag{4.1}$$

Then, for $j \in [\ell-1]$, $\mathscr{B}$ sets $(pk_{\mathbf{z}_j}, sk_{\mathbf{z}_j}) = \mathcal{E}.\mathsf{Setup}(1^\kappa)$, and for $i \in [\ell]$,

$$\gamma_i = \prod_{j\in[\ell-1]} pk_{\mathbf{z}_j}^{\lambda_{i,j}} \quad \text{and} \quad pk_i = pk^{\alpha_i}\gamma_i.$$

Eventually, $\mathscr{B}$ invokes $\mathscr{A}$ on input $mpk = (pk_i)_{i\in[\ell]}$.

Notice that, $\mathscr{B}$ is implicitly setting $sk_i = \alpha_i sk + \sum_{j\in[\ell-1]} \lambda_{i,j} sk_{\mathbf{z}_j}$ because of the LKH property, where sk is the secret key corresponding to pk, which is unknown to $\mathscr{B}$.

Challenge Ciphertext. $\mathscr{B}$ computes the challenge ciphertext ct^* as follows. $\mathscr{B}$ randomly picks $b \xleftarrow{\$} \{0,1\}$, computes $\mathcal{E}.\mathsf{E}(\gamma_i, 0; r)$ from ct_0 and $\sum_{j\in[\ell-1]} \lambda_{i,j} sk_{\mathbf{z}_j}$ and $\mathcal{E}.\mathsf{E}(1_H, x_{b,i}; r)$ from secret key 0_G and ct_0, by randomness reuse. $\mathscr{B}$ then sets

$$ct_0^* = ct_0 \quad \text{and} \quad (ct_i^* = ct_1^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, 0; r) \cdot \mathcal{E}.\mathsf{E}(1_H, x_{b,i}; r))_{i\in[\ell]} \ ,$$

Then the algorithm returns the challenge ciphertext $ct^* = (ct_0^*, (ct_i^*)_{i\in[\ell]})$.

Secret Keys. To generate a secret key for vector $\mathbf{y}$, $\mathscr{B}$ computes $sk_{\mathbf{y}}$ as

$$sk_{\mathbf{y}} = \sum_{j\in[\ell-1]} \left(\sum_{i\in[\ell]} y_i \lambda_{i,j} \right) sk_{\mathbf{z}_j}$$

At the end of the simulation, if $\mathscr{A}$ correctly guesses b, then $\mathscr{B}$ returns 0 ($\mathscr{B}$ guesses that $\mathscr{C}$ encrypted 0), else $\mathscr{B}$ returns 1 ($\mathscr{B}$ guesses that $\mathscr{C}$ encrypted a). This concludes the description of adversary $\mathscr{B}$.

It remains to verify that $\mathscr{B}$ correctly simulates $\mathscr{A}$'s environment.

First see that the master public key is well distributed, because we are just applying a change of basis to a well distributed master public key. Now it holds that $\alpha_i = \frac{x_{1,i}-x_{0,i}}{\|\mathbf{x}_1-\mathbf{x}_0\|^2}$ because

$$\begin{aligned} x_{1,i} - x_{0,i} &= \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{e}_i \rangle \\ &= \alpha_i \|\mathbf{x}_1 - \mathbf{x}_0\|^2 + \sum_{j\in[\ell-1]} \lambda_{i,j} \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{z}_j \rangle \\ &= \alpha_i \|\mathbf{x}_1 - \mathbf{x}_0\|^2 \ , \end{aligned}$$

where the $\mathbf{e}_i$ are the vectors of the canonical basis.

To ensure that $\|\mathbf{x}_1 - \mathbf{x}_0\|^2$ is different from 0 modulo q, fixed the message space $M = \{0, \ldots, B-1\} \subseteq \mathcal{R}$, q needs to be set to be a prime larger then $\ell \cdot B^2$.

Now recall that a vector $\mathbf{y}$ satisfying the security game constraints is such that $\langle \mathbf{y}, \mathbf{x}_0 \rangle = \langle \mathbf{y}, \mathbf{x}_1 \rangle$, so

$$\sum_{i\in[\ell]} y_i \alpha_i = \sum_{i\in[\ell]} y_i \frac{x_{1,i} - x_{0,i}}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2} = 0$$

which in turn implies that a secret key $sk_{\mathbf{y}}$ for the vector $\mathbf{y}$ is distributed as

$$\begin{aligned} sk_{\mathbf{y}} = \sum_{i\in[\ell]} y_i sk_i &= \sum_{i\in[\ell]} y_i \alpha_i sk + \sum_{i\in[\ell]} \sum_{j\in[\ell-1]} y_i \lambda_{i,j} sk_{\mathbf{z}_j} \\ &= \sum_{j\in[\ell-1]} \left(\sum_{i\in[\ell]} y_i \lambda_{i,j} \right) sk_{\mathbf{z}_j} \end{aligned}$$

On the other hand, if $\mathscr{A}$ asks for a secret key for some vector $\mathbf{y} \notin (\mathbf{x}_1 - \mathbf{x}_0)^\perp$, $\mathscr{B}$ would need to know sk in order to generate a correct secret key for $\mathbf{y}$.

Now, we have to analyze the following two cases, depending on which message was encrypted by $\mathscr{C}$ in the challenge ciphertext:

1. $\mathscr{C}$ encrypted 0. Then, the challenge ciphertext ct^* for message $\mathbf{x}_b$ is distributed as

   $$ct_0^* = ct_0$$

   and

   $$\begin{aligned} ct_i^* &= \mathcal{E}.\mathsf{E}(pk, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, x_{b,i}; r) \\ &= \mathcal{E}.\mathsf{E}(pk_i, x_{b,i}; r) \ , \end{aligned}$$

   thanks to the LCH property, and then as in the real game.

   Thus, in this case, $\mathscr{B}$ generates a view identical to that $\mathscr{A}$ would see in the real game. Hence, the advantage of $\mathscr{B}$ in this game is ε, the same advantage as $\mathscr{A}$ against s-IND-IPFE-CPA of $\mathcal{FE}$ when 0 has been encrypted.

2. $\mathscr{C}$ encrypted a. First, in Equation 4.1, we have $\alpha_i = (x_{1,i} - x_{0,i})/\|\mathbf{x}_1 - \mathbf{x}_0\|^2$. Let us analyze the distribution of the challenge ciphertext in this case. We have $ct_0^* = ct_0$ and

   $$\begin{aligned} ct_i^* &= \mathcal{E}.\mathsf{E}(pk, a; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(\gamma_i, x_{b,i}; r) \\ &= \mathcal{E}.\mathsf{E}(pk_i, x_{b,i} + \alpha_i a; r) \\ &= \mathcal{E}.\mathsf{E}(pk_i, \hat{x}_i; r), \end{aligned}$$

thanks to the LCH property, where $\hat{x}_i$ is defined as follows:

$$\begin{aligned}\hat{x}_i = x_{b,i} + \alpha_i a &= \frac{a}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}(x_{1,i} - x_{0,i}) + x_{b,i} \\ &= \frac{a}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2}(x_{1,i} - x_{0,i}) + x_{0,i} + b(x_{1,i} - x_{0,i}).\end{aligned}$$

Let us set $u = a/\|\mathbf{x}_1 - \mathbf{x}_0\|^2 + b$, which is a random value in the full message space of $\mathcal{E}$, given that a is random in the same space, then $\hat{x}_i = ux_{1,i} + (1-u)x_{0,i}$. Then, the challenge ciphertext is a valid ciphertext for the message $\hat{\mathbf{x}} = u\mathbf{x}_1 + (1-u)\mathbf{x}_0$, which is a random linear combination of the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ whose coefficients sum up to one, as expected. Notice that b is information theoretically hidden because the distribution of u is independent from b. Hence, the advantage of $\mathscr{B}$ in this game is 0, when a random non-zero a has been encrypted.

Eventually, this shows that ε is bounded by the best advantage one can get against s-IND-IPFE-CPA of $\mathcal{E}$. Hence, taking the maximal values, the best advantage one can get against s-IND-IPFE-CPA of $\mathcal{FE}$ is bounded by the best advantage one can get against s-IND-IPFE-CPA of $\mathcal{E}$. □

4.3.3 Security in the General Case

If the underlying PKE scheme does not support randomness-reuse defined in the previous section then it is still possible to prove the security of Construction 4.3.1 using the security properties defined in Section 4.2.3. We have the following security theorem, similar to Theorem 4.3.3.

Theorem 4.3.5. *Let $\mathcal{E}$ be a public-key encryption scheme with the structural properties defined in Section 4.2.1, and let $\mathcal{FE}$ be the functional encryption scheme for the inner-product functionality obtained by applying Construction 4.3.1 to $\mathcal{E}$. If $\mathcal{E}$ is* s-IND-CPA*, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, ℓ-public-key-reproducible, and ℓ-ciphertext-reproducible then $\mathcal{FE}$ is* s-IND-IPFE-CPA*.*

Proof. The proof strategy is essentially the same as that of Theorem 4.3.3. For increased clarity and readability, we prove security via a sequence of hybrid experiments. We first describe a sequence of hybrid games and then show indistinguishability between adjacent hybrids, thus proving s-IND-IPFE-CPA security.

Hybrid H_1: This is the s-IND-IPFE-CPA game.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$	**proc Encrypt**()
$(msk, mpk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$	$ct^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
$V \leftarrow \emptyset$	Return ct^*
Return mpk	**proc Finalize**(b')
proc KeyDer$(\mathbf{y})$	**if** $\exists \mathbf{y} \in V$ such that
$V \leftarrow V \cup \{\mathbf{y}\}$	$\langle \mathbf{x}_0, \mathbf{y}\rangle \neq \langle \mathbf{x}_1, \mathbf{y}\rangle$
$1sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$	**then return** false
Return $sk_{\mathbf{y}}$	Return $(b' = b)$

Chapter 4

Hybrid H_2: This is like H_1 except that the master public key is generated by invoking the algorithm $H_2.\mathsf{Setup}$ defined as follows:

$H_2.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$: The algorithm finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell - 1]$, there exist $\lambda_{i,j} \in \mathcal{R}$ and $\alpha_i \in \mathcal{R}$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j .$$

Then, the algorithms samples $sk \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$ and, for $j \in [\ell]$, PKE secret key $sk_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$. Finally, the algorithm sets:

$$sk_i = \alpha_i \cdot sk + \sum \lambda_{i,j} sk_{\mathbf{z}_j}, \quad pk_i = \mathcal{E}.\mathsf{PKGen}(sk_i, \tau) \ ,$$

where τ is the same used in the Setup algorithm. The algorithm returns $msk = (sk_i)$ and $mpk = (pk_i)_{i \in [\ell]}$.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$
$(msk, mpk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$
$V \leftarrow \emptyset$
Return mpk

proc KeyDer$(\mathbf{y})$
$V \leftarrow V \cup \{\mathbf{y}\}$
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$
Return $sk_{\mathbf{y}}$

proc Encrypt()
$\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
Return Ct^*

proc Finalize(b')
if $\exists \mathbf{y} \in V$ such that
$\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$
then return false
Return $(b' = b)$

Hybrid H_3: This is like H_2 except that the master public key is generated by invoking the algorithm $H_3.\mathsf{Setup}$ and the secret keys are generated by invoking the algorithm $H_3.\mathsf{KeyDer}$ which are defined as follows.

$H_3.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$: The algorithm finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell-1]$, there exist $\lambda_{i,j} \in \mathcal{R}$ and $\alpha_i \in \mathcal{R}$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j.$$

Then, the algorithms first samples $sk \xleftarrow{\$} \mathcal{E}.\mathsf{SKGen}(1^\kappa)$ and $pk \xleftarrow{\$} \mathcal{E}.\mathsf{PKGen}(sk, \tau')$. Then, for $j \in [\ell-1]$, a PKE secret keys $sk_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$, sets $t_i = \sum \lambda_{i,j} sk_{\mathbf{z}_j}$, $pk_{t_i} \leftarrow \mathcal{E}.\mathsf{PKGen}(t_i, \tau_i)$ and computes

$$pk_i = pk^{\alpha_i} \cdot pk_{t_i}.$$

The algorithm returns $mpk = (pk_i)_{i \in [\ell]}$ and $msk' = (pk, sk_{\mathbf{z}_j})$.

$H_3.\mathsf{KeyDer}(msk', \mathbf{y})$: The algorithm computes secret key for vector $\mathbf{y}$ in the following way: $sk_{\mathbf{y}} = \sum_{i \in [\ell]} y_i \cdot t_i$.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $\boxed{(mpk, msk') \xleftarrow{\$} H_3.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)}$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $\boxed{sk_{\mathbf{y}} \xleftarrow{\$} H_3.\mathsf{KeyDer}(msk', k)}$ Return $sk_{\mathbf{y}}$	**proc Encrypt**() $\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Hybrid H_4: This is like H_3 except that the challenge ciphertext is generated by invoking the algorithm $H_4.\mathsf{Encrypt}$ defined as follows:

$H_4.\mathsf{Encrypt}(msk', \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$ct_0 = \mathcal{E}.\mathsf{C}(r) \quad \text{and} \quad \left(ct_i = \mathcal{E}.\mathsf{E}(pk, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(pk_{t_i}, x_i; r)\right)_{i \in [\ell]},$$

where r is some randomness in the random space of $\mathcal{E}$.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{\$} H_3.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} H_3.\mathsf{KeyDer}(msk', k)$ Return $sk_{\mathbf{y}}$	**proc Encrypt**() $\boxed{\mathsf{Ct}^* \xleftarrow{\$} H_4.\mathsf{Encrypt}(msk', \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Hybrid H_5: This is like H_4 except that the challenge ciphertext is generated by invoking the algorithm $H_5.\mathsf{Encrypt}$ defined as follows:

$H_5.\mathsf{Encrypt}(msk', ct, \mathbf{x})$: Let $ct = (ct_0, ct_1)$, then, the algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$ct'_0 = ct_0 \quad \text{and} \quad \left(ct'_i = ct_1^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}'(t_i, x_i, ct_0; \tilde{r})\right)_{i \in [\ell]} ,$$

where $\tilde{r}$ is some randomness shared among all the invocation of $\mathcal{E}.\mathsf{E}$.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{\$} H_3.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} H_3.\mathsf{KeyDer}(msk', k)$ Return $sk_{\mathbf{y}}$	**proc Encrypt**() $\boxed{ct = \mathcal{E}.\mathsf{Encrypt}(pk, 0)}$ $\boxed{\mathsf{Ct}^* \xleftarrow{\$} H_5.\mathsf{Encrypt}(msk', ct, \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Hybrid H_6: This is like H_5 except that ct encrypts a random value $a \in \mathbb{Z}_p$.

proc Initialize$(\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $(mpk, msk') \xleftarrow{\$} H_3.\mathsf{Setup}(1^\kappa, \mathbf{x}_0, \mathbf{x}_1)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} H_3.\mathsf{KeyDer}(msk', k)$ Return $sk_{\mathbf{y}}$	**proc Encrypt**() $\boxed{ct = \mathcal{E}.\mathsf{Encrypt}(pk, a),\ a \leftarrow \mathbb{Z}_p}$ $\mathsf{Ct}^* \xleftarrow{\$} H_5.\mathsf{Encrypt}(msk', ct, \mathbf{x}_b)$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

We now show that the relevant distinguishing probabilities between adjacent hybrids are negli- gible, which completes the proof.

Indistinguishability of H_1 and H_2: The distribution of the master public key in both the games is identically. This is because in H_2 a simple change of basis is applied to a well distributed master secret key and this change of basis can be computed due to the linear key-homomorphism of $\mathcal{E}$, that tells us that the sk_i's as computed in H_2 are valid secret keys of $\mathcal{E}$.

Moreover, notice that, by our change of basis it holds that $\alpha_i = x_{1,i} - x_{0,i}$ because

$$\begin{aligned} x_{1,i} - x_{0,i} &= \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{e}_i \rangle \\ &= \alpha_i + \sum_{j \in [\ell-1]} \lambda_{i,j} \langle \mathbf{x}_1 - \mathbf{x}_0, \mathbf{z}_j \rangle \\ &= \alpha_i \ . \end{aligned}$$

Then, the change of basis implies that for all the vectors $\mathbf{y} = (y_1, \ldots, y_\ell)$ satisfying the security game constraints, meaning that $\mathbf{y} \in (\mathbf{x}_1 - \mathbf{x}_0)^\perp$, it holds that $\sum_{i \in [\ell]} y_i \cdot \alpha_i = 0$. Thus, to generate a well-distributed secret key for a $\mathbf{y}$ satisfying the security game constraints, sk is not required.

Indistinguishability of H_2 and H_3: This holds under the ℓ-public-key-reproducibility of $\mathcal{E}$ for the distribution $\mathcal{M}$ induced by the challenge messages, defined as follows:

$\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}(1^\kappa)$: $\mathcal{M}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell - 1]$, there exist $\lambda_{i,j} \in \mathcal{R}$ and $\alpha_i \in \mathcal{R}$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j .$$

Then, $\mathcal{M}$ first samples $sk \xleftarrow{\$} \mathcal{E}.\mathsf{SKGen}(1^\kappa)$, then, for $j \in [\ell - 1]$, samples $sk_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$, sets $t_i = \sum \lambda_{i,j} sk_{\mathbf{z}_j}$ and gives in output $(sk, (\alpha_i, t_i)_{i \in [\ell]})$.

For the sake of completeness, suppose that there exists and adversary $\mathscr{A}$ that distinguishes with non-negligible advantage H_2 from H_3. Then, we can construct an adversary $\mathscr{B}$ that uses $\mathscr{A}$ as a subroutine and breaks the ℓ-public-key-reproducibility of $\mathcal{E}$.

$\mathscr{B}$ does the following. $\mathscr{B}$ invokes $\mathscr{A}$ to gets the challenge messages $\mathbf{x}_0, \mathbf{x}_1$. Then, $\mathscr{B}$ receives from the challenger of the ℓ-public-key-reproducibility experiment values $(pk_i, t_i)_{i \in [\ell]}$ on input the security parameter and distribution $\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}$. $\mathscr{B}$ sets master public key as $mpk = (pk_i)_{i \in [\ell]}$ and gives it to $\mathscr{A}$. Then $\mathscr{B}$ answers secret key queries by using the t_i's which are enough to generate secret keys for vectors satisfying the security game constraints. $\mathscr{B}$ generates the challenge ciphertext by using mpk and uses $\mathscr{A}$'s guess as its guess for the ℓ-public-key-reproducibility game.

Finally, notice that if $\mathscr{B}$ plays $\mathsf{Exp}^{\ell\text{-pk-rep-0}}_{\mathcal{E},\kappa}(\mathscr{B})$ (resp. $\mathsf{Exp}^{\ell\text{-pk-rep-1}}_{\mathcal{E},\kappa}(\mathscr{B})$), then $\mathscr{B}$ perfectly simulates H_2 (resp. H_3).

Indistinguishability of H_3 and H_4: By linear ciphertext-homomorphism of $\mathcal{E}$, $H_3 = H_4$. In fact, notice that in both the games $pk_i = pk^\alpha \cdot pk_{t_i}$, then it holds that

$$\mathcal{E}.\mathsf{E}(pk^{\alpha_i} \cdot pk_{t_i}, x_i; r) = \mathcal{E}.\mathsf{E}(pk, 0; r)^{\alpha_i} \cdot \mathcal{E}.\mathsf{E}(pk_{t_i}, x_i; r) \ .$$

Indistinguishability of H_4 and H_5: This holds under the ℓ-ciphertext-reproducibility of $\mathcal{E}$ for the distribution $\mathcal{M}$, induced by the challenge messages, defined as follows:

$\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}(1^\kappa)$: $\mathcal{M}$ finds a basis $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{\ell-1})$ of $(\mathbf{x}_1 - \mathbf{x}_0)^\perp$. Then, the canonical vectors can be rewritten in this basis as follows: for $i \in [\ell], j \in [\ell - 1]$, there exist $\lambda_{i,j} \in \mathcal{R}$ and $\alpha_i \in \mathcal{R}$ such that:

$$\mathbf{e}_i = \alpha_i \frac{(\mathbf{x}_1 - \mathbf{x}_0)}{||\mathbf{x}_1 - \mathbf{x}_0||^2} + \sum_{j \in [\ell-1]} \lambda_{i,j} \mathbf{z}_j .$$

Then, $\mathcal{M}$ samples, for $j \in [\ell - 1]$, secret keys $sk_{\mathbf{z}_j} \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$.

Finally, $\mathcal{M}$, for $i \in [\ell]$, sets $t_i = \sum \lambda_{i,j} sk_{\mathbf{z}_j}$ and returns $(0, (\alpha_i, x_{\beta,i}, sk_{t_i})_{i\in[\ell]})$, where β is random bit.

For the sake of completeness, suppose that there exists and adversary $\mathscr{A}$ that distinguishes with non-negligible advantage H_4 from H_5. Then, we can construct an adversary $\mathscr{B}$ that uses $\mathscr{A}$ as a subroutine and breaks the ℓ-ciphertext-reproducibility of $\mathcal{E}$.

$\mathscr{B}$ does the following. $\mathscr{B}$ invokes $\mathscr{A}$ to gets the challenge messages $\mathbf{x}_0, \mathbf{x}_1$. Then, $\mathscr{B}$ receives from the challenger of the ℓ-ciphertext-reproducibility values $(pk, (\alpha_i, pk_{t_i}, t_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]})$ on input the security parameter and distribution $\mathcal{M}^{\mathbf{x}_0,\mathbf{x}_1}$. $\mathscr{B}$ sets master public key as $mpk = (pk_i = pk^{\alpha_i} \cdot pk_{t_i})_{i\in[\ell]}$ and gives it to $\mathscr{A}$. Then $\mathscr{B}$ answers secret key queries by using the t_i's which are enough to generate secret keys for vectors satisfying the security game constraints. $\mathscr{B}$ uses as challenge ciphertext $(ct_0, (ct_i)_{i\in[\ell]})$ and uses $\mathscr{A}$'s guess as its guess for the ℓ-ciphertext-reproducibility game.

Finally, notice that if $\mathscr{B}$ plays $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\kappa}(\mathscr{B})$ (resp. $\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{E},\kappa}(\mathscr{B})$), then $\mathscr{B}$ perfectly simulates H_4 (resp. H_5).

Indistinguishability of H_5 and H_6: The only difference between H_5 and H_6 is in the message that ct encrypts. In H_5, ct is an encryption of 0 and H_5, it is an encryption of a for a random $a \in \mathcal{R}$. Moreover, notice that sk is never used. Therefore under the the IND-CPA security of $\mathcal{E}$, H_5 is computational indistinguishable from H_6.

Advantage of any adversary in H_6. In this game, the challenge ciphertext is for message $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_\ell)$ defined as follows:

$$\begin{aligned}\hat{x}_i = x_{b,i} + \alpha_i a &= a(x_{1,i} - x_{0,i}) + x_{b,i} \\ &= a(x_{1,i} - x_{0,i}) + x_{0,i} + b(x_{1,i} - x_{0,i}).\end{aligned}$$

Let us set $u = a + b$, which is a random value in $\mathbb{Z}_p$, then $\hat{x}_i = ux_{1,i} + (1-u)x_{0,i}$. Then, the challenge ciphertext is a valid ciphertext for the message $\hat{\mathbf{x}} = u\mathbf{x}_1 + (1-u)\mathbf{x}_0$, which is a random linear combination of the vectors $\mathbf{x}_0$ and $\mathbf{x}_1$ whose coefficients sum up to one. Notice that b is information theoretically hidden because the distribution of u is independent from b. Hence, the advantage of any adversary in this game is 0. □

4.4 Generic Adaptively Secure Inner-Product Functional Encryption

Now that we constructed a selectively secure inner-product functional encryption scheme, we want to improve it to reach full adaptive security against chosen-plaintext attacks. In the case of modular IPFE schemes, one can use the generic Construction 3.2.4. However, we want to stay as generic as possible and include the non-modular IPFE schemes to our framework.

4.4.1 Construction and Correctness

Our IND-IPFE-CPA secure scheme is very closely related to the two schemes of Construction 4.3.1 and Construction 3.2.4. It is essentially the latter construction applied to the former, but instead of sampling uniform vectors $\mathbf{t}$ to mask the vector in the proof, we sample it in some set $\mathcal{T} \subset \mathbb{Z}$. So the construction boils down to encrypting $(\mathbf{x}, 0)$ and decrypting with key $(\mathbf{y}, -\langle \mathbf{t}, \mathbf{y}\rangle)$ with the selectively secure scheme.

Construction 4.4.1 (Adaptively Secure Inner-Product Functional Encryption from Public Key Encryption)**.** *Let us consider a PKE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *with the properties defined previously, we define our inner-product functional encryption scheme* $\mathcal{FE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *as follow. We set* $\mathcal{T} = \{0, \ldots, T\}^\ell$ *in this case, where* T *will be set according to the security properties needed. (*T/M_x *superpolynomial is needed for security against polynomially bounded adversaries,* T/M_x *exponential provides security against sub-exponetially bounded adversaries, where* M_x *is the biggest possible coordinate of any vector in* $\mathcal{M}_x$*)*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (msk, mpk)$. *On input security parameter* κ *and functionality parameter* ℓ*, generates* ℓ *independent secret keys*

$$\forall i \in [\ell],\ s_i \xleftarrow{\$} \mathcal{E}.\mathsf{SKGen}(1^\kappa),$$

  and a key pair

$$(sk, pk) \xleftarrow{\$} \mathcal{E}\mathsf{Setup}(1^\kappa),$$

  sharing the same public parameters pp, and

$$\mathbf{t} \xleftarrow{\$} \mathcal{T},$$

  and samples ℓ *public keys corresponding to the secret keys* $sk_i = s_i + t_i sk$

$$\forall i \in [\ell],\ pk_i \xleftarrow{\$} \mathcal{E}.\mathsf{PKGen}(s_i + t_i sk),$$

  and returns master secret key $msk = (s_1, \ldots, s_\ell, \mathbf{t})$ *and master public key* $mpk = (pk_1, \ldots, pk_\ell, pk)$*;*

- $\mathsf{KeyDer}(msk, \mathbf{y}) \mapsto sk_{\mathbf{y}}$. *On input master secret key* msk*, and key* $\mathbf{y} \in \mathcal{M}_y$*, outputs user secret key*

$$sk_{\mathbf{y}} \leftarrow (\sum_{i \in [\ell]} y_i s_i, \sum_{i \in [\ell]} y_i t_i);$$

- $\mathsf{Encrypt}(mpk, \mathbf{x}) \mapsto ct_{\mathbf{x}}$. *On input master public key* mpk*, and plaintext* $\mathbf{x} \in \mathcal{M}_x$*, samples shared randomness* r *in the randomness space of* $\mathcal{E}$*, and computes*

$$ct_0 \leftarrow \mathcal{E}.\mathsf{C}(r), \qquad ct_1 \leftarrow \mathcal{E}.\mathsf{E}(pk, 0; r),$$

$$\forall i \in [\ell],\ ct_{2,i} \leftarrow \mathcal{E}.\mathsf{E}(pk_i, x_i; r),$$

  and outputs ciphertext $ct_{\mathbf{x}} = (ct_0, ct_1, (ct_{2,i})_{i \in [\ell]})$*;*

- $\mathsf{Decrypt}(sk_{\mathbf{y}}, ct_{\mathbf{x}}) \mapsto m$ *or* $\perp$. *On input user secret key* $sk_{\mathbf{y}}$*, and ciphertext* $ct_{\mathbf{x}}$*, returns the output of*

$$\mathcal{E}.\mathsf{Decrypt}(sk_{\mathbf{y},0}, (ct_0, \prod_{i \in [\ell]} ct_{2,i}^{y_i} \cdot ct_1^{-sk_{\mathbf{y},1}})).$$

Correctness. Once again, the correctness follows from the homomorphic properties of the scheme $\mathcal{E}$ as well as its correctness. It is directly implied by the correctness of Construction 4.3.1 and Construction 3.2.4. It is easy to see that the plaintext computed by the

homomorphic properties is the value of the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$, while the key required for decryption is $\sum_{i \in [\ell]} y_i(s_i + t_i sk) - \sum_{i \in [\ell]} t_i sk = sk_{\mathbf{y}}$.

The main downside of this construction, is that $\mathcal{T}$ can be huge. This is a very slight problem for efficiency, but it can become a problem in security too, because it impacts the parameters we have to handle reproducibility for, and leads stronger assumption in the case of lattice-based schemes for example.

4.4.2 Adaptive Security

The security of Construction 4.4.1 is the same as the security of Construction 4.3.1, except that here we know exactly in advance the challenge vector to protect: $\mathbf{0}$ against $(\mathbf{t}, 1)$, as shown in the proof of Theorem 3.2.5, and at the end, our choice of $\mathcal{T}$ ensures that $\mathbf{x}$ is statistically hidden from the adversary.

Theorem 4.4.2. *Let $\mathcal{E}$ be a public-key encryption scheme with the structural properties defined in Section 4.2.1, and let $\mathcal{FE}$ be the functional encryption scheme for the inner-product functionality obtained by applying Construction 4.4.1 to $\mathcal{E}$. If $\mathcal{E}$ is* s-IND-CPA*, linear-key homomorphic, linear-ciphertext homomorphic under shared randomness, ℓ-public-key-reproducible, and ℓ-ciphertext-reproducible with coefficients $\alpha_i \in \mathcal{T}$, then $\mathcal{FE}$ is* IND-IPFE-CPA.

Proof. We prove security via a sequence of hybrid experiments, and then we show they are indistinguishable. The techniques used to switch public keys and ciphertext values with reproducibility properties closely follows the proof of Theorem 4.3.5.

Hybrid H_1: This is the IND-IPFE-CPA game:

proc Initialize(κ)	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$
$(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$	$\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$
$V \leftarrow \emptyset$	Return Ct^*
Return mpk	**proc Finalize**(b')
proc KeyDer$(\mathbf{y})$	**if** $\exists \mathbf{y} \in V$ such that
$V \leftarrow V \cup \{\mathbf{y}\}$	$\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$
$sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$	**then return** false
Return $sk_{\mathbf{y}}$	Return $(b' = b)$

Hybrid H_2: This is like H_1 except that the master public key is generated by invoking the algorithm $H_2.\mathsf{Setup}$ defined as follows:

$H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$: The algorithm samples $sk \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$, for $i \in [\ell]$, PKE secret key $s_i \leftarrow \mathcal{E}.\mathsf{SKGen}(1^\kappa)$ and uniformly random scalar $t_i \xleftarrow{\$} \mathcal{T}$. Finally, the algorithm sets:

$$pk = \mathcal{E}.\mathsf{PKGen}(sk, \tau) \qquad sk_i = s_i + t_i \cdot sk$$
$$pk_{s_i} = \mathcal{E}.\mathsf{PKGen}(s_i, \tau_i) \qquad pk_i = pk^{t_i} \cdot pk_{s_i}$$

where τ is the same as used in the Setup algorithm, and τ_i is such that $pk_{s_i} \cdot pk^{t_i}$ is close to $\mathcal{E}.\mathsf{PKGen}(sk_i)$. The algorithm returns $mpk = (pk, (pk_i)_{i \in [\ell]})$ and $msk = (s_i, t_i)_{i \in [\ell]}$.

proc Initialize(κ) $\boxed{(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)}$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b)$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Under the ℓ-public-key-reproducibility of $\mathcal{E}$, H_1 and H_2 are indistinguishable.

Hybrid H_3: This is like H_2 except that the challenge ciphertext is generated by invoking the algorithm $H_3.\mathsf{Encrypt}$ defined as follows:

$H_3.\mathsf{Encrypt}(msk, pk, \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$ct_0 = \mathcal{E}.\mathsf{C}(r) \qquad ct_1 = \mathcal{E}.\mathsf{E}(pk, 0; r) \qquad ct_{2,i} = ct_1^{t_i} \cdot \mathcal{E}.\mathsf{E}(pk_{s_i}, x_i; r)$$

where r is some randomness in the random space of $\mathcal{E}$.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\boxed{\mathsf{Ct}^* \xleftarrow{\$} H_3.\mathsf{Encrypt}(msk, pk, \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

By linear ciphertext-homomorphism of $\mathcal{E}$, $H_2 = H_3$.

Hybrid H_4: This is like H_3 except that the challenge ciphertext is generated by invoking the algorithm $H_4.\mathsf{Encrypt}$ defined as follows:

$H_4.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x})$: Let $\mathsf{Ct} = (ct_0, ct_1)$. Then, the algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$ct'_0 = ct_0 \qquad ct'_1 = ct_1 \qquad ct'_{2,i} = ct_1^{t_i} \cdot \mathcal{E}.\mathsf{E}'(s_i, x_i, ct_0; \tilde{r})$$

where $\mathcal{E}.\mathsf{E}'$ is the alternative encryption algorithm defined in the ℓ-ciphertext-reproducibility game, $\tilde{r}$ is some randomness shared among all the invocation of $\mathcal{E}.\mathsf{E}'$.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{E}(pk, 0)}$ $\boxed{\mathsf{Ct}^* \xleftarrow{\$} H_4.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Under the ℓ-ciphertext-reproducibility of $\mathcal{E}$, H_3 and H_4 are indistinguishable.

Hybrid H_5: This is like H_4 except that Ct encrypts 1.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\boxed{\mathsf{Ct} = \mathcal{E}.\mathsf{E}(pk, 1)}$ $\mathsf{Ct}^* \xleftarrow{\$} H_4.\mathsf{Encrypt}(msk, \mathsf{Ct}, \mathbf{x}_b)$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Under the s-IND-CPA security of $\mathcal{E}$, H_4 and H_5 are indistinguishable.

Hybrid H_6: This is like H_5 except that the challenge ciphertext is generated by invoking the algorithm $H_6.\mathsf{Encrypt}$ defined as follows:

$H_6.\mathsf{Encrypt}(msk, pk, \mathbf{x})$: The algorithm computes the ciphertext for $\mathbf{x}$ in the following way:

$$ct_0 = \mathcal{E}.\mathsf{C}(r) \qquad ct_1 = \mathcal{E}.\mathsf{E}(pk, 1; r) \qquad ct_{2,i} = ct_1^{t_i} \cdot \mathcal{E}.\mathsf{E}(pk_{s_i}, x_i; r),$$

where r is some randomness in the random space of $\mathcal{E}$.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\boxed{\mathsf{Ct}^* \xleftarrow{\$} H_6.\mathsf{Encrypt}(msk, pk, \mathbf{x}_b)}$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Under the ℓ-ciphertext-reproducibility of $\mathcal{E}$, H_5 and H_6 are indistinguishable.

Hybrid H_7: This is like H_6 except that the challenge ciphertext is generated by invoking the algorithm Encrypt.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} H_2.\mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b + \mathbf{t})$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

By linear ciphertext-homomorphism of $\mathcal{E}$, $H_7 = H_6$.

Hybrid H_8: This is like H_7 except that the master public key is generated by invoking the algorithm Setup.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ $\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b + \mathbf{t})$ Return Ct^* **proc Finalize**(b') **if** $\exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

Under the ℓ-public-key-reproducibility of $\mathcal{E}$, H_7 and H_8 are indistinguishable.

Hybrid H_9: This is like H_8 except that the simulator guesses the value of $\mathbf{x}_1 - \mathbf{x}_0$ and aborts if the guess is wrong.

proc Initialize(κ) $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}(1^\kappa, 1^\ell)$ $V \leftarrow \emptyset$ $\mathbf{z} \xleftarrow{\$} \{\mathbf{x}_1 - \mathbf{x}_0 : \mathbf{x}_0, \mathbf{x}_1 \in \mathcal{M}_x\}$ Return mpk **proc KeyDer**$(\mathbf{y})$ $V \leftarrow V \cup \{\mathbf{y}\}$ $sk_{\mathbf{y}} \xleftarrow{\$} \mathsf{KeyDer}(msk, \mathbf{y})$ Return $sk_{\mathbf{y}}$	**proc Encrypt**$(\mathbf{x}_0, \mathbf{x}_1)$ **if** $\mathbf{x}_1 - \mathbf{x}_0 \neq \mathbf{z}$: **then** bad $\leftarrow$ true $\mathsf{Ct}^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathbf{x}_b + \mathbf{t})$ Return Ct^* **proc Finalize**(b') **if** bad $\vee \exists \mathbf{y} \in V$ such that $\langle \mathbf{x}_0, \mathbf{y} \rangle \neq \langle \mathbf{x}_1, \mathbf{y} \rangle$ **then return** false Return $(b' = b)$

If the adversary has advantage ϵ in H_9, then it has advantage $|\{\mathbf{x}_1 - \mathbf{x}_0 : \mathbf{x}_0, \mathbf{x}_1 \in \mathcal{M}_x\}|\epsilon$ in H_8.

Advantage of any Adversary in H_9. To analyze the advantage of an adversary in H_9, let us define Hybrids $H_{9,0}$ and $H_{9,2}$ be the games corresponding to H_9 when b is fixed to 0 and 1 respectively. Then we construct $H_{9,1}$ from $H_{9,0}$ by replacing $\mathbf{t}$ by $\mathbf{t}'$ and $\mathbf{s}$ by $\mathbf{s}'$ defined as follows:

$$\mathbf{t}' = \mathbf{t} + \mathbf{z}, \quad s'_i = s_i - \mathbf{z}_i sk$$

The statistical distance between $H_{9,1}$ and $H_{9,0}$ is upper bounded by the probability $\Pr[\mathbf{t}' \notin \mathcal{T} \vee \mathbf{s}' \notin \mathcal{S}]$. If the secret key sk is chosen uniformly in a group, taking $\mathcal{T}/(2\ell M_x^2)$ superpolynomial in κ is sufficient for the security to hold. If instead the secret key sk is chosen uniformly in an interval of $\mathbb{N}$, say $[S]$, we need to sample the s_i from $[S']$ instead, where $S'/(2\ell M_x S)$ is superpolynomial in κ. We now conclude by noticing that $H_{9,2}$ and $H_{9,1}$ are perfectly indistinguishable: if there is no abort and for any vector $\mathbf{y}$ satisfying the security game constraints, i.e. $\langle \mathbf{y}, \mathbf{x}_0 \rangle = \langle \mathbf{y}, \mathbf{x}_1 \rangle$, it holds that, since $\mathbf{x}_0 + \mathbf{t} = \mathbf{x}_1 + \mathbf{t}'$,

$$\sum_{i \in [\ell]} y_i t_i = \sum_{i \in [\ell]} y_i t'_i \quad \text{and} \quad \sum_{i \in [\ell]} y_i s_i = \sum_{i \in [\ell]} y_i s'_i.$$

We note that we had to guess the value of the challenge ciphertext, because otherwise, the values of $\mathbf{t}'$ and $\mathbf{s}'$ wouldn't be defined at the beginning of the experiment. This guess only hinders the security in the last hybrid, on a statistical argument, so it doesn't make the underlying assumption weaker.

□

Remark 4.4.3. *The only constraint we have on $\mathcal{T}$ is that it must be that for any two vectors $\mathbf{x}_0, \mathbf{x}_1$ in $\mathcal{M}_x$, for any vector $\mathbf{t}$ in $\mathcal{T}$, with overwhelming probability, $\mathbf{t} + \mathbf{x}_1 - \mathbf{x}_0$ or $\mathbf{t} - \mathbf{x}_1 + \mathbf{x}_0$ is in $\mathcal{T}$.*
We conjecture that for $\mathcal{M}_x = \{0, \ldots, M_x\}^\ell$ there exist a finite, non-empty set $\mathcal{T}$ for which this probability is 1; meaning that for any two vectors $\mathbf{x}_0, \mathbf{x}_1$ in $\mathcal{M}_x$, for any vector $\mathbf{t}$ in $\mathcal{T}$, either $\mathbf{t} + \mathbf{x}_1 - \mathbf{x}_0$ is in $\mathcal{T}$, or $\mathbf{t} - \mathbf{x}_1 + \mathbf{x}_0$ is in $\mathcal{T}$. This set would be a discretized hypersphere (integral points inside an hypersphere) in dimension ℓ whose radius would only depend on M_x.
If this conjecture turned out to be true, our scheme wouldn't require T to be superpolynomial in the security parameter, resulting in a better assumption for the schemes presented in Chapter 8.

Chapter 5

Generic Construction from Projective Hash Functions

In this chapter, we present the construction of IND-IPFE-CCA secure IPFE schemes from projective hash functions in our work [BBL17].
As a brief introduction, we first give an overview of the construction and explain the ideas behind the proofs.
Then we will present our construction in two steps: first we construct an IND-IPFE-CPA secure scheme, and then we upgrade it to an IND-IPFE-CCA secure scheme. For each step we will present the properties that we require on the PHFs, before going on to the construction and finally the proof of security.

Contents

5.1 Overview **64**
5.2 IPFE-CPA Friendly Projective Hash Functions **67**
5.2.1 Key Homomorphism 67
5.2.2 Projection Key Homomorphism 68
5.2.3 Strong Diversity 68
5.2.4 Translation Indistinguishability 69
5.2.5 IPFE-CPA Friendliness 69
5.3 Inner-Product Functional Encryption Secure Against Chosen-Plaintext Attacks **69**
5.3.1 Construction and Correctness 70
5.3.2 Security Against Chosen-Plaintext Attacks 72
5.4 IPFE-CCA Friendly Projective Hash Functions **74**
5.4.1 Tag-Based Projective Hash Function 75
5.4.2 2-Universality 76
5.4.3 Universal Translation Indistinguishability 76
5.4.4 IPFE-CCA Friendliness 77
5.5 Inner-Product Functional Encryption Secure Against Chosen-Ciphertext Attacks **77**
5.5.1 Construction and Correctness 78
5.5.2 Security Against Chosen-Ciphertext Attacks 80

5.1 Overview

In order to reach security against chosen-ciphertext attacks, we give up a bit on the generality of the framework and instead of using a public key encryption scheme as our building block, we are going to use the stronger projective hash functions.
Before presenting our construction, we give a quick informal introduction to projective hash functions and illustrate how to use them with a simple example. Then we will present the ideas behind the construction of a IND-IPFE-CPA secure inner-product functional encryption scheme, and how to improve it to reach IND-IPFE-CCA security.

Projective Hash Functions. Projective hash functions were defined by [CS02] in order to construct public key encryption schemes secure against chosen-plaintext attacks and even against chosen-ciphertext attacks. This concept has since found many other applications. Notably in password authenticated key exchange as in [GL03; CHK+05; KV09; KM14], but not only: [Kal05; ACP09; Wee12; BPV12; Wee16].

One of the most basic example use of projective hash functions is a zero knowledge proof of knowledge of a witness of a word in some NP language. In this example, Bob wants to prove to Alice that he knows a witness w that some word **b** is in a language. However, he does not want to reveal his witness to Alice. Here is how to solve this problem using projective hash functions as illustrated on Figure 5.1:

1. Alice calls the algorithm HashKg to create a hashing key hk.
2. Alice derives a projection key hp from her hashing key hk using ProjKg and sends it to Bob.
3. Bob computes a hash h of the word **b** using the algorithm ProjHash with inputs the witness w and the projection key hp. Then, Bob sends h to Alice
4. Alice computes the hash h' of the word **b** using the algorithm Hash with inputs the word **b** and the hashing key hk. She checks that $h' = h$ and is convinced that **b** is indeed in the language.

Why is Alice convinced? This is illustrated on Figure 5.2: if **b** is not in the language, Bob cannot know a witness w, and hence it is nearly impossible for him to guess correctly the value of the hash h. So alice knows that if Bob gives her the correct hash, he must know a witness, and thus **b** is in the language.
Why is this proof zero knowledge? Well, Bob only gave Alice a value she already knew, which is computable with the knowledge of hk and **b**, so she doesn't gain any additional knowledge from this proof.

Overview of the Constructions. We note that the PHF could be used to construct a public key encryption that could be generically transformed into an IPFE scheme using the results from Chapter 4, but presenting the first construction helps giving intuition about how we are using PHF to build IPFE schemes, and it also avoid more cumbersome notations.

The Figures 3.1 and 5.1 illustrate very well the parallel between inner-product functional encryption and projective hash function. From these, it is easy to see the strategy to construct the former using the latter as a building block, by mapping each node of the diagram to a node on the other diagram. The secret key will be hashing keys, while the public key will be projection keys. To encrypt, we will sample a word **b** together with a witness w, and use the

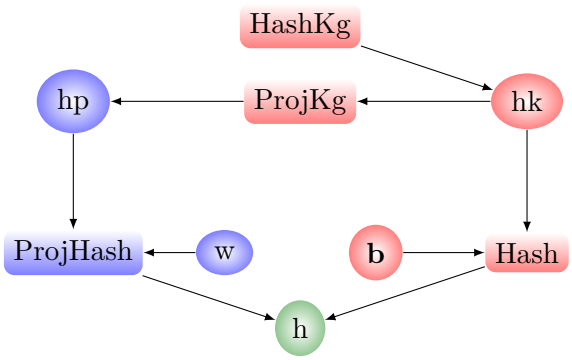

Figure 5.1: A zero knowledge proof using projective hash functions. w is a witness that **b** is in the language. In red is Alice's view, and in blue is Bob's view. Green means that Alice and Bob check that they have the same value.

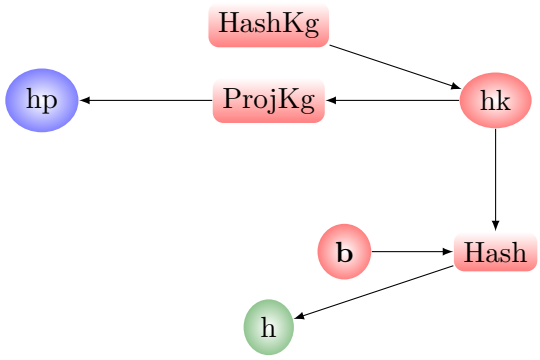

Figure 5.2: A zero knowledge proof using projective hash functions. **b** is not in the language. In red is Alice's view, and in blue is Bob's view. Here Bob cannot guess the value of h.

witness and the projection keys to derives hashes that will be used as a mask in order to hide the plaintext. Then to decrypt, we remove the mask using the hashing keys to hash the word **b**. As in Chapter 4, we will rely on additive homomorphism in order to ensure correctness, as well as security against collusions of our scheme. This technique gives an IND-IPFE-CPA secure IPFE scheme. This construction is simpler than the one from public key encryption. However, it cannot be applied to lattice-based schemes, for which we only know of approximate PHFs. We could maybe adapt the IND-IPFE-CPA secure scheme to include these schemes, but this would make the notation cumbersome and the PHFs wouldn't meet the requirements to build IND-IPFE-CCA secure schemes.

In order to reach IND-IPFE-CCA security, we need to ensure that the adversary cannot learn any information he shouldn't using the decryption oracle. To prevent leakage of information, our strategy is to ensure that ill-formed ciphertexts can be detected at decryption time. To do this, we use a second PHF with more statistical properties in order to ensure that the word is indeed inside the language, and that the adversary is not trying to get information

about the hashing keys by decrypting ciphertext for words outside the language. We also prevent malleability by using the technique already described in Section 3.3. In the end, our scheme is really close to ℓ parallel repetitions of the Cramer-Shoup encryption scheme [CS98], but instead of using a hash of the witness as a tag, we use the standard approach based on one-time signatures because the former wouldn't yield an IND-IPFE-CCA secure scheme in our case.

Security Against Chosen-Plaintext Attacks. The strategy to prove IND-IPFE-CPA security of our first scheme is not very far from the strategy used in Chapter 4 and the one used in [ALS16]. The proof proceeds in two steps:

1. First, we replace the word of the challenge ciphertext to a word outside the language. In order to generate the correct distribution for the ciphertext, we instead generate it using the hashing keys instead of the projection keys.

2. The second step is to use statistical properties of the projective hash functions to argue that the adversary cannot guess the challenge bit with non-negligible probability. The strategy here is to switch the hashing key for another hashing key that gives the same projection key, but a different hash for the word chosen in the challenge ciphertext.

In the second step, we will require to know the challenge messages, so we will guess them, as done in complexity leveraging. However, here we only guess the vectors in the second part of the proof, so the security loss doesn't apply to the computational argument which is used in the first step. In practice, this only impacts the size of the secret keys and doesn't weaken the hardness of the underlying assumption.

Security Against Chosen-Ciphertext Attacks. The core idea of our construction is similar to the one used in the Cramer-Shoup encryption scheme [CS98; CS02]: adding a hash value (from a 2-universal PHF) to ensure that the word $\mathfrak{b}$ is in the language $\mathcal{L}$, to our generic IND-IPFE-CPA construction in Section 5.3. Then, at least information-theoretically, the hash values $\mathsf{hash}(\mathsf{hk}_i, \mathfrak{b})$ used to decrypt a ciphertext could be computed using only the projection keys $\mathfrak{hp}_i$ and do not leak any information about hk_i. We can then conclude using the same ideas as in the IND-IPFE-CPA security proof of our generic construction.

However, this does not work directly, as checking a 2-universal hash value require to know the corresponding hashing key $\mathsf{hk}^\dagger$, and knowing this hashing key enables to fake these hash values. In other words, with the naive scheme described previously, an attacker knowing a secret key for any $\vec{y}$ could then generate a ciphertext with $\mathfrak{b} \notin \mathcal{L}$, but a valid 2-universal hash values. This completely removes the usefulness of the 2-universal hash value.

Our new idea is the following: instead of using only one hash value, we use ℓ such values. The secret key $sk_{\vec{y}}$ only enables to check that a linear combination (with coefficient $\vec{y}$) of these hash values is valid. This uses the key homomorphism property. Knowing $sk_{\vec{y}}$ enables to generate hash values that would be accepted by the decryption oracle with $\vec{y}$, and knowing $sk_{\vec{y}}$ for multiple vectors $\vec{y}$ enables to generate hash values for any vector in the span of these $\vec{y}$. But intuitively, this is not really an issue, as if the attacker already knows $sk_{\vec{y}}$, calling the decryption oracle for $\vec{y}$ is of no use to him, as he could decrypt the given ciphertext himself. The proof however is more subtle and requires a careful design of hybrid games to deal with adaptivity and the fact that we are working over a ring and not a field. In particular, we cannot directly rely on the notion of span of vectors.

5.2 IPFE-CPA Friendly Projective Hash Functions

In this section we present the properties required by our first projective hash function in order to build an IND-IPFE-CPA secure inner-product functional encryption scheme that will be later improved to IND-IPFE-CCA security. We first define 3 new properties on the projective hash functions, that we will regroup under one property for the sake of readability later.

5.2.1 Key Homomorphism

For correctness of the IPFE we will need the following property. Like in the generic construction from public key encryption, we use an additive homomorphism to compute the result of the inner-product, while changing the key used for decryption. We give an illustration of this property on Figure 5.3, and the formal definition below.

Definition 5.2.1 (Key Homomorphism [BJL16])**.** *A projective hash function* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *for a subset membership problem* $\mathbf{P}$ *is* key-homomorphic*, if it satisfies the following additional properties:*

1. *the set* $\mathcal{K}$ *of hashing keys and the set* Π *of hash values are additive Abelian groups, with polynomial time group operations;*

2. *for any instance* Λ*, and any word* $\mathfrak{b} \in \mathcal{X}$*, the function* $\mathsf{hk} \in \mathcal{K} \mapsto \mathsf{hash}(\mathsf{hk}, \mathfrak{b}) \in \Pi$ *is a group homomorphism, that is,* $\mathsf{hash}(\mathsf{hk}, \mathfrak{b}) + \mathsf{hash}(\mathsf{hk}', \mathfrak{b}) = \mathsf{hash}(\mathsf{hk} + \mathsf{hk}', \mathfrak{b})$*, for any* $\mathsf{hk}, \mathsf{hk}' \in \mathcal{K}$*.*

We do not require $\mathcal{K}$ to be finite. For example, in the DCR construction in Chapter 7, $\mathcal{K} = \mathbb{Z}$. However, we require that each group element of $\mathcal{K}$ and Π has a unique representation as a bit-string.

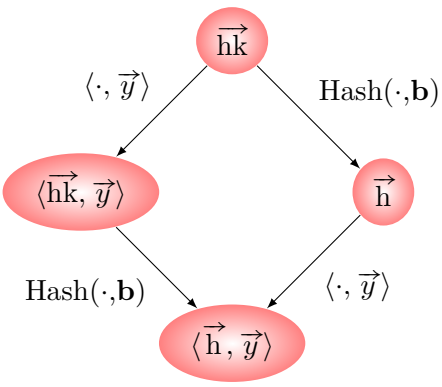

Figure 5.3: Illustration of key homomorphism. We see that a linear combination of the hashing keys gives the same linear combination of the hash values for any word. This property is similar to the linear ciphertext homomorphism under shared randomness of Section 4.2.2. Here the shared randomness is the word **b**.

5.2.2 Projection Key Homomorphism

For our IND-IPFE-CCA secure construction, we do not only need key homomorphism, in the security proof, we also need to simulate projection key, so we are going to require another homomorphic property, *projection key homomorphism*. It is only required in Section 5.4.3 (for the CCA security), but we will introduce it already here to discuss some possible simplifications of the next properties.

Definition 5.2.2 (Projection Key Homomorphism)**.** *A projective hash function* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *for a subset membership problem* $\mathbf{P}$ *is* projection-key-homomorphic *if it satisfies the following additional properties:*

1. *the set* $\mathcal{K}$ *of hashing keys and the set* $\mathcal{K}_{\mathsf{hp}}$ *of projection keys are additive Abelian groups, with polynomial time group operations;*
2. *for any instance* Λ*, the function* $\mathsf{hk} \in \mathcal{K} \mapsto \mathsf{projkg}(\mathsf{hk}) \in \mathcal{K}_{\mathsf{hp}}$ *is a group homomorphism, that is,* $\mathsf{projkg}(\mathsf{hk} + \mathsf{hk}') = \mathsf{projkg}(\mathsf{hk}) + \mathsf{projkg}(\mathsf{hk}')$*, for any* $\mathsf{hk}, \mathsf{hk}' \in \mathcal{K}$*.*

5.2.3 Strong Diversity

The second property we need for our PHFs is strong diversity. More precisely, we require that for each $\mathfrak{b}$ there exists a (not necessarily efficiently computable) hashing key $\mathsf{hk}_\perp(\mathfrak{b})$, such that hk and $\mathsf{hk} + \mathsf{hk}_\perp(\mathfrak{b})$ result in the same projection key, while the hash value of $\mathfrak{b}$ under the key $\mathsf{hk}_\perp(\mathfrak{b})$ is equal to $\mathfrak{g}_\perp$, where $\mathfrak{g}_\perp$ is a fixed efficiently computable group element.

Definition 5.2.3 (Strong diversity)**.** *A key-homomorphic projective hash function* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *for a subset membership problem* $\mathbf{P}$ *is* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp)$*-strongly diverse for a function* $\mathsf{hk}_\perp : \bar{\mathcal{L}} \to \Pi$*, an element* $\mathfrak{g}_\perp$ *of* Π*, and a positive integer* $M_\perp$*, if the following properties are satisfied:*

1. $\mathfrak{g}_\perp$ *and* $M_\perp$ *can be efficiently computed from* Λ*;*
2. *the group element* $\mathfrak{g}_\perp$ *has order* $M_\perp$*,*
3. *for any hashing key* $\mathsf{hk} \in \mathcal{K}$ *and any word* $\mathfrak{b} \in \bar{\mathcal{L}}$*:*

$$\mathsf{projkg}(\mathsf{hk} + \mathsf{hk}_\perp(\mathfrak{b})) = \mathsf{projkg}(\mathsf{hk})\,, \tag{5.1}$$

$$\mathsf{hash}(\mathsf{hk}_\perp(\mathfrak{b}),\, \mathfrak{b}) = \mathfrak{g}_\perp\ . \tag{5.2}$$

We do not require $\mathsf{hk}_\perp$ to be efficiently computable, as we are only using it to bound statistical distance.

In what follows, we will use the following straightforward lemma.

Lemma 5.2.4. *If a key-homomorphic PHF is also projection-key homomorphic, then Equation Equation (5.1) is true iff* $\mathsf{projkg}(\mathsf{hk}_\perp(\mathfrak{b})) = 0$*.*

Relation with Diverse Groups. Diverse groups were introduced in [CS02] as a way to construct PHFs. They can be seen as key-homomorphic projection-key-homomorphic strongly diverse PHFs with the two following differences: $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$ (instead of $\bar{\mathcal{L}} \subseteq \mathcal{X} \setminus \mathcal{L}$), and for any $\mathsf{hk} \in \mathcal{K}$ and any $\mathfrak{b} \in \bar{\mathcal{L}}$, it is only required that $\mathsf{hash}(\mathsf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) \neq 0$ instead of $\mathsf{hash}(\mathsf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) = \mathfrak{g}_\perp$. Nevertheless, all the diverse groups we currently know of are also strongly diverse for $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$.

5.2.4 Translation Indistinguishability

We also require one last statistical property, translation indistinguishability. Informally it says that translating the hashing key of the PHF by a small multiple of $\mathsf{hk}_\perp(\mathfrak{b})$ cannot be detected with non-negligible probability. In the proof, we use this as a statistical argument to conclude after using the computational assumption.

Definition 5.2.5 (Translation indistinguishability)**.** *A key-homomorphic projective hash function* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *is* $(\mathsf{hk}_\perp, M_x, \varepsilon_{\mathrm{ti}})$-translation-indistinguishable *for a function* $\mathsf{hk}_\perp : \bar{\mathcal{L}} \to \Pi$*, a positive integer* M_x*, and* $\varepsilon_{\mathrm{ti}} \in [0,1]$*, if for any integer* $x \in \{-M_x, \ldots, M_x\}$ *and for any* $\mathfrak{b} \in \bar{\mathcal{L}}$*,*

$$\Delta(\mathsf{hashkg}(\Lambda), \mathsf{hashkg}(\Lambda) + x \cdot \mathsf{hk}_\perp(\mathfrak{b})) \leq \varepsilon_{\mathrm{ti}} \ .$$

Important Particular Case: Key Uniformity. For many key-homomorphic PHFs, like the ones based on DDH and MDDH that we are going to use in Chapter 6, the output of hashkg is actually uniform over the group $\mathcal{K}$. In this case, the PHF is automatically $(\cdot, \cdot, 0)$-translation-indistinguishable. More formally, we have the following lemma.

Lemma 5.2.6. *Let* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a key-homomorphic PHF such that the distribution of* $\mathsf{hashkg}(\Lambda)$ *is uniform over* $\mathcal{K}$*. Let* $\bar{\mathcal{L}}$ *be a non-empty subset of* $\mathcal{X}$*,* $\mathsf{hk}_\perp$ *be a function from* $\bar{\mathcal{L}}$ *to* Π *and* M_x *be a positive integer. Then* PHF *is* $(\mathsf{hk}_\perp, M_x, 0)$*-translation-indistinguishable.*

Proof. Both $\mathsf{hashkg}(\Lambda)$ and $\mathsf{hashkg}(\Lambda) + x \cdot \mathsf{hk}_\perp(\mathfrak{b})$ are uniform group elements in $\mathcal{K}$. □

5.2.5 IPFE-CPA Friendliness

In the following, we regroup all 3 properties we have defined under the IPFE-CPA friendliness property.

Definition 5.2.7 (IPFE-CPA Friendliness)**.** *A projective hash function* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *is* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$-IPFE-CPA-friendly *for a function* $\mathsf{hk}_\perp$ *from* $\bar{\mathcal{L}}$ *to* Π*, an element* $\mathfrak{g}_\perp$ *of* Π*, and two positive integers* $M_\perp$ *and* M_x*, if it is key-homomorphic,* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp)$*-strongly diverse, and* $(\mathsf{hk}_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-translation-indistinguishable.*

We are now ready to show our construction of inner-product functional encryption from key homomorphic projective hash functions.

5.3 Inner-Product Functional Encryption Secure Against Chosen-Plaintext Attacks

We now define our generic construction for $\mathsf{IND\text{-}IPFE\text{-}CPA}$ secure IPFEs. Intuitively, we use ℓ PHFs in parallel, that are combined during decryption in order to only reveal a linear combination of the hashes, which implies that it only reveals this same linear combination of the messages. This restriction is enforced by the key generation algorithm, which only outputs linear combinations of the hashing keys.

5.3.1 Construction and Correctness

We first give an informal illustration of the construction and its correctness via an illustration on Figure 5.4. It is a very intuitive construction given the definitions of projective hash functions and the key homomorphism property. Then, we explicit more formally the construction and prove its correctness.

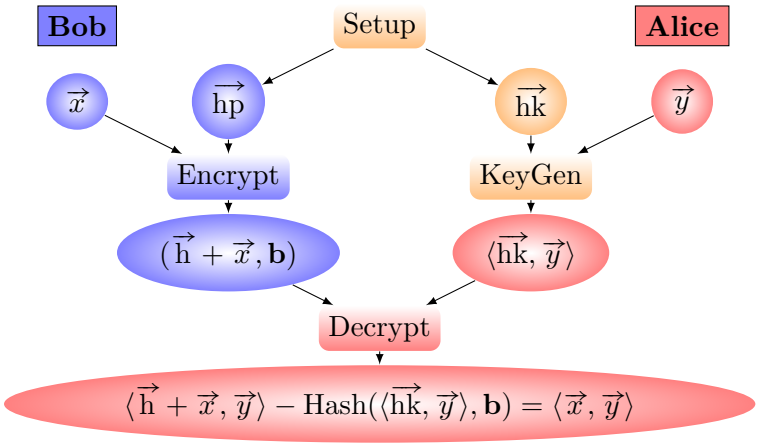

Figure 5.4: Construction of Inner-Product Functional Encryption from Key Homomorphic Projective Hash Function. This illustration is a copy of Figure 3.1, replacing the variables by their values in our construction. The correctness follows from the key homomorphism and the correctness of the projective hash function.

We suppose that we have a $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, x, \varepsilon_{\mathrm{ti}})$-IPFE-CPA-friendly projective hash function $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ for a subset membership problem $\mathbf{P}$. Let $\mathcal{R}$ be the ring $\mathbb{Z}$ or $\mathbb{Z}_{M_\perp}$, let ℓ be a positive integer parameter corresponding to the length of the message and key vectors, and let $\mathcal{M}_y$ and $\mathcal{M}_x$ two subsets of $\mathcal{R}^\ell$. We always suppose ℓ to be polynomial in the security parameter κ.

We suppose that the following condition is satisfied.

Condition 1. *Using the above notation:*

1. *if $\mathcal{R} = \mathbb{Z}_{M_\perp}$, the order of any hashing key $\mathsf{hk} \in \mathcal{K}$ divides $M_\perp$;*
2. *$\mathcal{M}_y$ and $\mathcal{M}_x$ are efficiently recognizable subsets of $\mathcal{R}^\ell$;*
3. *for any $\vec{x} \in \mathcal{M}_x$ and any i, $x_i \in \{-M_x, \ldots, M_x\}$;*
4. *there exists a polynomial time algorithm (in the security parameter κ) that given as input $ct_{\vec{y}} = \langle \vec{y}, \vec{x} \rangle \cdot \mathfrak{g}_\perp$ for $\vec{y} \in \mathcal{M}_y$ and $\vec{x} \in \mathcal{M}_x$, can compute $\log_{\mathfrak{g}_\perp} ct_{\vec{y}} = \langle \vec{y}, \vec{x} \rangle$;*
5. *for any $\vec{y} \in \mathcal{M}_y$ and $\vec{x} \in \mathcal{M}_x$, $\langle \vec{y}, \vec{x} \rangle$ is the same over $\mathcal{R}$ and over $\mathbb{Z}_{M_\perp}$ (this condition is trivial when $\mathcal{R} = \mathbb{Z}_{M_\perp}$).*

The first subcondition implies that $\mathcal{K}$ is a $\mathcal{R}$-module, which implies that, for any $t \in \mathcal{R}$, $t \cdot \mathsf{hk}$ is well defined. The second subcondition enables KeyDer and $\mathsf{Encrypt}$ to check in polynomial-time the validity of their arguments y and x respectively. The third subcondition

is used in the proof to apply the $(\mathsf{hk}_\perp, M_x, \varepsilon_{\mathrm{ti}})$-translation indistinguishability property. The fourth subcondition ensures that decryption can be performed in polynomial time. The last subcondition is similar as the condition in the "over $\mathbb{Z}$ constructions in [ALS16]. If $\mathcal{R} = \mathbb{Z}_{M_\perp}$, then—as in [ALS16]—a simple way to guarantee that subconditions 3 and 5 hold is to assume that $|y_i|, |x_i| < (M_\perp/\ell)^{1/2}$ for each $\overrightarrow{y} \in \mathcal{M}_y$, $\overrightarrow{x} \in \mathcal{M}_x$, and $i \leq \ell$. The fourth subcondition can potential restrict the values $|y_i|$ and $|x_i|$ even more. We are now ready to present our scheme and to prove its correctness.

Construction 5.3.1 (Inner-Product Functional Encryption secure against Chosen-Plaintext Attacks from Projective Hash Functions)**.** *Let* $\mathbf{P}$ *be a subset membership problem. Let* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CPA-friendly PHF. We assume that Condition 1 is satisfied. We define our inner-product functional encryption scheme* $\mathcal{FE}$ *as follows.*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (msk, mpk)$*. On input security parameter* κ *and functionality parameter* ℓ*, samples* $\Lambda \xleftarrow{\$} I_\kappa$*, sets* $pp = (\kappa, \ell, \Lambda)$*, computes for all* i *in* $[\ell]$

$$\mathsf{hk}_i \xleftarrow{\$} \mathsf{hashkg}(\Lambda)\,, \qquad\qquad \mathfrak{hp}_i \leftarrow \mathsf{projkg}(\mathsf{hk}_i)\,,$$

  and returns master secret key $msk = (pp, \overrightarrow{\mathsf{hk}})$ *and master public key* $mpk = (pp, \overrightarrow{\mathfrak{hp}})$*;*

- $\mathsf{KeyDer}(msk, \overrightarrow{y}) \mapsto sk_{\overrightarrow{y}}$*. On input master secret key* msk*, and key* $\overrightarrow{y} \in \mathcal{M}_y$*, computes*

$$\mathsf{hk}_{\overrightarrow{y}} \leftarrow \langle \overrightarrow{y}, \overrightarrow{\mathsf{hk}} \rangle\,,$$

  and outputs user secret key $sk_{\overrightarrow{y}} = \mathsf{hk}_{\overrightarrow{y}}$*;*

- $\mathsf{Encrypt}(mpk, \overrightarrow{x}) \mapsto ct_{\overrightarrow{x}}$*. On input master public key* mpk*, and plaintext* $\overrightarrow{x} \in \mathcal{M}_x$*, samples a random pair* $(\mathfrak{b}, w) \in \varrho$*, computes for all* i *in* $[\ell]$

$$ct_i \leftarrow \mathsf{projhash}(\mathfrak{hp}_i, \mathfrak{b}, w) + x_i \cdot \mathfrak{g}_\perp\,,$$

  and outputs ciphertext $ct_{\overrightarrow{x}} = (\mathfrak{b}, \overrightarrow{ct})$*;*

- $\mathsf{Decrypt}(sk_{\overrightarrow{y}}, ct_{\overrightarrow{x}}) \mapsto m$ *or* $\perp$*. On input user secret key* $sk_{\overrightarrow{y}}$*, and ciphertext* $ct_{\overrightarrow{x}}$*, computes*

$$ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle} \leftarrow \langle \overrightarrow{y}, \overrightarrow{ct} \rangle - \mathsf{hash}(\mathsf{hk}_{\overrightarrow{y}}, \mathfrak{b})$$

  and returns $\log_{\mathfrak{g}_\perp} ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle}$*.*

Correctness. The correctness of our construction follows from the correctness and key homomorphism of the PHF, the fact that $\mathfrak{g}_\perp$ has order $M_\perp$, and from Condition 1, as follows:

$$\begin{aligned}
\langle \overrightarrow{y}, \overrightarrow{ct} \rangle &= \sum_{i=1}^{\ell} y_i \cdot (\mathsf{projhash}(\mathfrak{hp}_i, \mathfrak{b}, w) + x_i \cdot \mathfrak{g}_\perp) \\
&= \sum_{i=1}^{\ell} y_i \cdot \mathsf{hash}(\mathsf{hk}_i, \mathfrak{b}) + \sum_{i=1} y_i \cdot x_i \cdot \mathfrak{g}_\perp \\
&= \mathsf{hash}(\sum_{i=1}^{\ell} y_i \cdot \mathsf{hk}_i, \mathfrak{b}) + \langle \overrightarrow{y}, \overrightarrow{x} \rangle \cdot \mathfrak{g}_\perp = \mathsf{hash}(\mathsf{hk}_{\overrightarrow{y}}, \mathfrak{b}) + \langle \overrightarrow{y}, \overrightarrow{x} \rangle \cdot \mathfrak{g}_\perp \ .
\end{aligned}$$

Now, because $\mathfrak{g}_\perp$ has order $M_\perp$, and thanks to Condition 1, one can extract $\langle \overrightarrow{y}, \overrightarrow{x} \rangle$ over $\mathcal{R}$. Since this equal to $\langle \overrightarrow{y}, \overrightarrow{x} \rangle$ over $\mathbb{Z}_{M_\perp}$, we get that the completeness holds.

5.3.2 Security Against Chosen-Plaintext Attacks

Before stating our theorem, we first define the following set:

$$\Delta\mathcal{M}_x := \{\vec{x}_1 - \vec{x}_0 | \vec{x}_0, \vec{x}_1 \in \mathcal{M}_x\} \ .$$

Its cardinality $|\Delta\mathcal{M}_x|$ is at most $(4M_x + 1)^\ell$, as the cardinality of $\mathcal{M}_x$ is at most $2M_x + 1$.

We have the following security theorem.

Theorem 5.3.2. *Let* $\mathbf{P}$ *be a subset membership problem. Let* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CPA-friendly projective hash function and let* $\mathcal{FE}$ *be the inner-product functional encryption scheme defined in Construction 5.3.1. We assume that Condition 1 is satisfied. Then* $\mathcal{FE}$ *is adaptively* IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A} = \mathscr{A}_{\mathcal{FE}}$ *that has advantage* $\varepsilon_{\mathscr{A}}$ *in breaking the* IND-IPFE-CPA *security of* $\mathcal{FE}$*, then there exists an attacker* $\mathscr{B}$ *that runs in approximately the same time and that has advantage* $\varepsilon_{\mathscr{B}}$ *in breaking the* $(\mathcal{L}, \bar{\mathcal{L}})$*-indistinguishability, such that*

$$\varepsilon_{\mathscr{A}} \leq 2 \cdot \varepsilon_{\mathscr{B}} + \ell \cdot |\Delta\mathcal{M}_x| \cdot \varepsilon_{\mathrm{ti}} \ .$$

Before going into the proof, we make the following remark on the precise advantage of the adversary in the reduction, and give a quick overview of the proof.

Remark 5.3.3. *When* $\varepsilon_{\mathrm{ti}} \neq 0$*, there is an exponential loss in the security proof in the term* $\ell|\Delta\mathcal{M}_x|\varepsilon_{\mathrm{ti}}$*. This term comes from the fact that at one point we guess the value of* $\vec{x}_1 - \vec{x}_0$*. This is not complexity leveraging, as the reduction loss is with regards to a statistical property. In particular, we do not need to rely on subexponential computational assumptions. Concretely, in our instantiations with DCR, we just need to take this security loss into account in the parameter* M *defining the bound on the size of the hashing key (see Chapter 7). This approximately multiplies by* $\log|\Delta\mathcal{M}_x|$ *the size of the secret keys which would be obtained if this security loss was not taken into account.*

We also remark that if we used a selective security notion, where the adversary announces $\vec{x}_0$ *and* $\vec{x}_1$ *before obtaining the public key, we would not lose the factor* $|\Delta\mathcal{M}_x|$*. We could then use classical complexity leveraging to go from this selective notion to the adaptive one we are considering. But then, we would need to use sub-exponential* $(\mathcal{L}, \bar{\mathcal{L}})$*-indistinguishability (if* ℓ *is polynomial in the security parameter), and the size of the ciphertexts, of the secret and public keys, and of the public parameters (and not just of the secret keys) would be multiplied by* $|\Delta\mathcal{M}_x|$*.*

For a quick overview of the proof, we illustrate the strategy on Figure 5.5.

Let us now move to the formal proof of the theorem.

Proof. We process through a sequence of games $Game_0, \ldots, Game_4$, at each game slightly modifying $\mathcal{FE}$. We denote by $\mathsf{Win}[Game_i]$ the probability that the adversary wins $Game_i$, and by $\mathsf{Adv}[Game_i] = |2 \cdot \mathsf{Win}[Game_i] - 1|$ its advantage. We then upper bound the advantage $\mathsf{Adv}[\Delta_i] = |\mathsf{Win}[Game_i] - \mathsf{Win}[Game_{i-1}]| = |\mathsf{Adv}[Game_i] - \mathsf{Adv}[Game_{i-1}]|/2$, that a probabilistic polynomial time adversary can have in distinguishing between the games $Game_i$ and $Game_{i-1}$, together with the advantage $\mathsf{Adv}[Game_4]$ that a probabilistic polynomial time adversary can have in the final game.

$Game_0$ corresponds to the actual IND-IPFE-CPA game. The challenge ciphertext is $(\mathfrak{b}, \vec{ct})$.

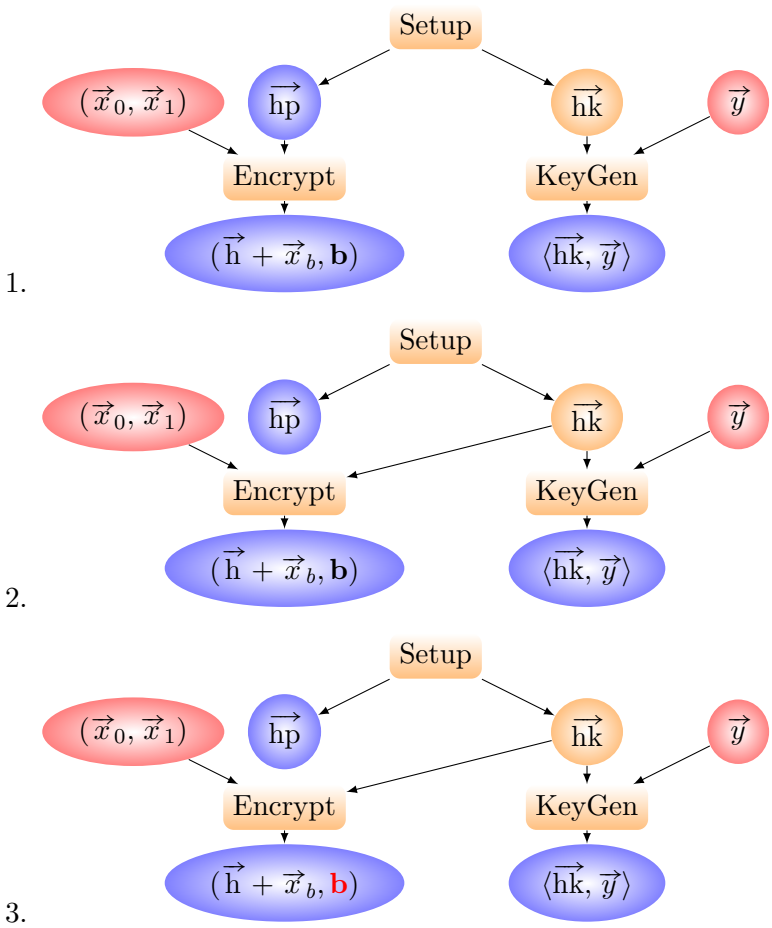

Figure 5.5: Strategy for the proof of Theorem 5.3.2. The first experiment is the IND-IPFE-CPA security game, then we present two other hybrids. In the first one, we change the way to compute the challenge ciphertext. In the second one, we change the word in the language. This last game is statistically unwinnable.

In $Game_1$, we change the way of computing the ciphertext. More precisely, we now compute ct_i as follows using hk_i instead of $\mathfrak{hp}_i$ and w:

$$ct_i \leftarrow \mathsf{hash}(\mathsf{hk}_i, \mathfrak{b}) + x_i \mathfrak{g}_\perp \ ,$$

which is perfectly indistinguishable thanks to the correctness of the PHF. Hence, $\mathsf{Adv}[\Delta_1] = 0$.

In $Game_2$, we sample $\mathfrak{b}$ from $\bar{\mathcal{L}}$ instead of from $\mathcal{L}$. Clearly, we can construct an attacker $\mathscr{B}$ that runs in approximately the same time as $\mathscr{A}$ such that $\mathsf{Adv}[\Delta_2] \leq \varepsilon_{\mathscr{B}}$, where $\mathscr{B}$ is the advantage of $\mathscr{B}$ against $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability.

In $Game_3$, we pick a uniform random value $\mathbf{x}^* \in \Delta\mathcal{M}_x$ at the beginning and abort if $\vec{x}_1 - \vec{x}_0 \neq \mathbf{x}^*$ and make the adversary win with probability $\frac{1}{2}$ in that case. As $\mathbf{x}^*$ is completely independent of everything else,

$$\mathsf{Adv}[Game_2] = |\Delta\mathcal{M}_x| \cdot \mathsf{Adv}[Game_3] \ ,$$

since:

$$\begin{aligned}\mathsf{Adv}[\mathit{Game}_3] &= |2 \cdot \mathsf{Win}[\mathit{Game}_3] - 1| \\ &= \left|2 \cdot \left(\frac{1}{|\Delta\mathcal{M}_x|} \cdot \mathsf{Win}[\mathit{Game}_2] + \frac{|\Delta\mathcal{M}_x| - 1}{|\Delta\mathcal{M}_x|} \cdot \frac{1}{2}\right) - 1\right| \\ &= \frac{1}{|\Delta\mathcal{M}_x|} \cdot |2 \cdot \mathsf{Win}[\mathit{Game}_2] - 1| = \frac{1}{|\Delta\mathcal{M}_x|} \cdot \mathsf{Adv}[\mathit{Game}_2] \ .\end{aligned}$$

In Game_4, we also abort if the adversary queries a vector $\vec{y} \in \mathcal{M}_y$ to the key oracle such that $\langle \vec{y}, \mathbf{x}^* \rangle \neq 0$. This game is perfectly indistinguishable from the previous one, as the adversary cannot ask for a key for $\vec{y} \in \mathcal{M}_y$ such that $\langle \vec{y}, \vec{x}_0 \rangle \neq \langle \vec{y}, \vec{x}_1 \rangle$. Hence, $\mathsf{Adv}[\Delta_4] = 0$.

Let us now bound $\mathsf{Adv}[\mathit{Game}_4]$. For that purpose, let $\mathit{Game}_{4,0}$ and $\mathit{Game}_{4,2}$ be the games corresponding to Game_4 when b is fixed to 0 and 1 respectively. We have:

$$\mathsf{Adv}[\mathit{Game}_4] = |\Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,2}] - \Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,0}]| \ .$$

We then construct the game $\mathit{Game}_{4,1}$ from $\mathit{Game}_{4,0}$ by replacing $\overrightarrow{\mathsf{hk}}$ by $\overrightarrow{\mathsf{hk}}'$ defined as follows:

$$\overrightarrow{\mathsf{hk}}' \leftarrow \overrightarrow{\mathsf{hk}} + \mathbf{x}^* \cdot \mathsf{hk}_\perp(\mathfrak{b}) \quad \text{where } \mathsf{hk}_i \xleftarrow{\$} \mathsf{hashkg}(\Lambda) \text{ for } i = 1, \ldots, \ell \ .$$

By translation indistinguishability, we have:

$$|\Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,1}] - \Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,0}]| \leq \ell\varepsilon_{\mathrm{ti}} \ .$$

We conclude by remarking that $\mathit{Game}_{4,1}$ and $\mathit{Game}_{4,2}$ are perfectly indistinguishable and that:

$$\Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,2}] = \Pr[b_{\mathscr{A}} = 1 \text{ in } \mathit{Game}_{4,1}] \ .$$

Indeed, in $\mathit{Game}_{4,1}$, assuming no abort, we have $\mathbf{x}^* = \vec{x}_1 - \vec{x}_0$ and for any allowed query $\vec{y}$ to KeyDer_{msk}:

$$\begin{aligned}\mathsf{projkg}(\mathsf{hk}'_i) &= \mathsf{projkg}(\mathsf{hk}_i) \\ ct_i &= \mathsf{hashkg}(\Lambda, \mathsf{hk}'_i, \mathfrak{b}) + x_{0,i}\mathfrak{g}_\perp = \mathsf{hashkg}(\Lambda, \mathsf{hk}_i, \mathfrak{b}) + x^*_i\mathfrak{g}_\perp + x_{0,i}\mathfrak{g}_\perp \\ &= \mathsf{hashkg}(\Lambda, \mathsf{hk}_i, \mathfrak{b}) + x_{1,i}\mathfrak{g}_\perp \\ \mathsf{hk}_{\vec{y}} &= \langle \vec{y}, \overrightarrow{\mathsf{hk}'} \rangle = \langle \vec{y}, \overrightarrow{\mathsf{hk}} \rangle + \langle \vec{y}, \mathbf{x}^* \rangle \mathsf{hk}_\perp(\mathfrak{b}) \\ &= \langle \vec{y}, \overrightarrow{\mathsf{hk}} \rangle \text{ as } \langle \vec{y}, \mathbf{x}^* \rangle = \langle \vec{y}, \vec{x}_1 \rangle - \langle \vec{y}, \vec{x}_0 \rangle = 0 \ .\end{aligned}$$

Finally, putting everything together, we get

$$\varepsilon_{\mathscr{A}} = \mathsf{Adv}[\mathit{Game}_0] \leq 2\mathsf{Adv}[\Delta_1] + 2\mathsf{Adv}[\Delta_2] + |\Delta\mathcal{M}_x| \cdot \mathsf{Adv}[\mathit{Game}_4] \leq 2\varepsilon_{\mathscr{B}} + \ell|\Delta\mathcal{M}_x|\varepsilon_{\mathrm{ti}} \ .$$

□

5.4 IPFE-CCA Friendly Projective Hash Functions

In order to achieve IND-IPFE-CCA security, we will require another kind of projective hash functions: *tag-based projective hash functions* [ABP15]. In this section, we first define this new tool, as well as the properties we need for our construction. As both a IPFE-CPA-friendly PHF and a IPFE-CCA-friendly PHF are used in our constructions of IND-IPFE-CCA inner-product functional encryption scheme in Section 5.5, we distinguish the two PHFs by adding a dagger to all the symbols defining the latter PHF. Both PHFs will be used on the same subset membership problem $\mathbf{P}$.

5.4.1 Tag-Based Projective Hash Function

A tag-based projective hash function [ABP15] is defined as a PHF, except that $\mathsf{hash}^\dagger$ and $\mathsf{projhash}^\dagger$ take an additional input (in some efficiently recognizable set $\mathcal{T}$) called a tag τ. We suppose that we can efficiently uniquely encode any 2κ-bit string as a tag τ, as a tag is usually the output of a collision-resistant hash-function. In our constructions, $\mathcal{T}$ is $\mathbb{Z}_M$ for some large integer M. As for basic projective hash functions, we provide some intuition on how to use tag-based PHFs using a diagram on Figure 5.6

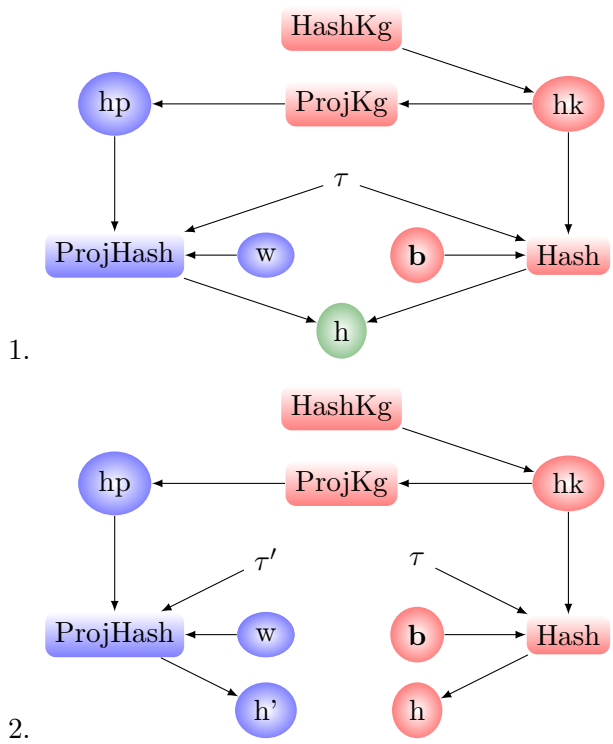

Figure 5.6: Tag-Based Projective Hash Functions. On the first illustration, we see the correctness property: if the same tag is used, then the same hash is derived. On the second illustration, we see the property that will be useful to prove the security against chosen-ciphertext attacks: if the tag is different, it is hard to guess the correct hash value.

Definition 5.4.1 (Tag-based Projective Hash Function [ABP15])**.** *Let* $\mathbf{P}$ *be a subset membership problem, specifying an ensemble* $(I_\ell)_{\ell\geq 0}$ *of instance distributions. A tag-based projective hash function for* $\mathbf{P}$ *is a tuple* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *of four probabilistic polynomial time algorithms:*

- $\mathsf{hashkg}^\dagger(\Lambda)$ *generates a hashing key* $\mathsf{hk}^\dagger$ *in a set* $\mathcal{K}^\dagger$ *for the instance* $\Lambda = \Lambda[\mathcal{X}, \mathcal{L}, \mathcal{W}, \varrho]$,
- $\mathsf{projkg}^\dagger(\mathsf{hk}^\dagger)$ *(deterministically) derives from the hashing key* $\mathsf{hk}^\dagger$ *a projection key* $\mathfrak{hp}^\dagger$ *from the set* $\mathcal{K}_{\mathsf{hp}}$ *of possible projection keys,*
- $\mathsf{hash}^\dagger(\mathsf{hk}^\dagger, \mathfrak{b}, \tau)$ *(deterministically) computes the hash value* $\mathfrak{H}^\dagger$ *(in some efficiently recognizable set* Π*), of* $\mathfrak{b} \in \mathcal{X}$ *under* $\mathsf{hk}^\dagger \in \mathcal{K}^\dagger$*, for the tag* $\tau \in \mathcal{T}$*,*

- $\mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger, \mathfrak{b}, w, \tau)$ *(deterministically) computes the projected hash value* $\mathfrak{p}\mathfrak{H}^\dagger$ *of* $\mathfrak{b} \in \mathcal{L}$ *using a witness* $w \in \mathcal{W}$*, for the tag* $\tau \in \mathcal{T}$*.*

It has to satisfy the following correctness property:

- *For any instance* Λ*, for any* $\mathfrak{b} \in \mathcal{X}$ *and* $w \in \mathcal{W}$*, s.t.* $(\mathfrak{b}, w) \in \varrho$*, for any hashing key* $\mathsf{hk}^\dagger \in \mathcal{K}^\dagger$*, for any tag* $\tau \in \mathcal{T}$*, if* $\mathfrak{hp}^\dagger \leftarrow \mathsf{projkg}^\dagger(\mathsf{hk}^\dagger)$*, then:*

$$\mathsf{hash}^\dagger(\mathsf{hk}^\dagger, \mathfrak{b}, \tau) = \mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger, \mathfrak{b}, w, \tau) \ .$$

The notion of key homomorphism can be adapted to tag-based PHFs in a straightforward way: it has to hold for any tag $\tau \in \mathcal{T}$.

In the sequel, we sometimes omit the term "tag-based" when it is clear from context.

5.4.2 2-Universality

We now recall the notion of *2-universality*, first introduced by Cramer and Shoup in [CS02], in order to ensure non-malleability. This will not be directly required by the tag-based PHF we use in the construction, but by a slight modification on it that will be used during the proof. It will ensure that decryption queries made by the adversary do not leak too much information.

Definition 5.4.2 (2-universality)**.** *A key-homomorphic tag-based projective hash function* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *for a subset membership problem* $\mathbf{P}$ *is* $\varepsilon^\dagger_{2\mathrm{u}}$-2-universal *if for any instance* Λ*, for any* $\mathfrak{b} \in \mathcal{X}$ *and* $\mathfrak{b}' \in \mathcal{X} \setminus \mathcal{L}$*, for any distinct tags* $\tau, \tau' \in \mathcal{T}$*, for any* $\mathfrak{hp}^\dagger \in \mathcal{K}_{\mathsf{hp}}$*, and for any* $\mathfrak{H}^\dagger \in \Pi$*,* $\tilde{\mathfrak{H}}^\dagger \in \Pi$*:*

$$\Pr_{\mathsf{hk}^\dagger}\left[\mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathsf{hk}^\dagger, \mathfrak{b}, \tau) \ \wedge \ \mathfrak{H}'^\dagger = \mathsf{hash}^\dagger(\mathsf{hk}^\dagger, \mathfrak{b}', \tau') \ \wedge \ \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathsf{hk}^\dagger)\right]$$
$$\leq \varepsilon^\dagger_{2\mathrm{u}} \cdot \Pr_{\mathsf{hk}^\dagger}\left[\mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathsf{hk}^\dagger, \mathfrak{b}, \tau) \ \wedge \ \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathsf{hk}^\dagger)\right] \ ,$$

where probabilities are taken over $\mathsf{hk}^\dagger \xleftarrow{\$} \mathsf{hashkg}^\dagger(\Lambda)$*. The PHF is* 2-universal *if it is* $\varepsilon^\dagger_{2\mathrm{u}}(\kappa)$*-2-universal for some negligible function* $\varepsilon^\dagger_{2\mathrm{u}}(\kappa)$*.*

In our generic construction, we will not require the PHF used in the construction to be 2-universal, but a variant of it where $\mathsf{hashkg}^\dagger$ is replaced by some other (not necessarily polynomial time) algorithm.

5.4.3 Universal Translation Indistinguishability

We also need one last statistical property to conclude the proof, as in the IND-IPFE-CPA case: *universal translation indistinguishability.* It is a strengthening of the previous translation indistinguishability in the sense that the algorithm defining the translation has to be the same for all words.

Definition 5.4.3 (Universal translation indistinguishability)**.** *A key-homomorphic tag-based projective hash function* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *is* $(\mathsf{hashkg}'^\dagger, M_x, \varepsilon^\dagger_{\mathrm{uti}})$-universally-translation-indistinguishable *for a (not necessarily polynomial time) algorithm* $\mathsf{hashkg}'^\dagger$ *taking as input* Λ *and outputting a hashing key* $\mathsf{hk}^\dagger$ *in some set* $\mathcal{K}'^{*\dagger} \subseteq \mathcal{K}$*, and for a positive integer* M_x*, if for any integer* x *such that* $|x| \leq M_x$*,*

$$\Delta(\mathsf{hashkg}^\dagger(\Lambda), \mathsf{hashkg}^\dagger(\Lambda) + x \cdot \mathsf{hashkg}'^\dagger(\Lambda)) \leq \varepsilon^\dagger_{\mathrm{uti}} \ .$$

Important Particular Case: Key Uniformity. For many key-homomorphic tag-based PHFs (i.e., the one based on the DDH assumption and its variants), the output of $\mathsf{hashkg}^\dagger$ is actually uniform over the group $\mathcal{K}^\dagger$. In this case, as for translation indistinguishability (Lemma 5.2.6), the PHF is automatically $(\mathsf{hashkg}'^\dagger, \cdot, 0)$-universally-translation-indistinguishable, for $\mathsf{hashkg}'^\dagger = \mathsf{hashkg}^\dagger$. More formally, we have the following lemma.

Lemma 5.4.4. *Let* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *be a key-homomorphic tag-based PHF such that the distribution of* $\mathsf{hashkg}^\dagger(\Lambda)$ *is uniform over* $\mathcal{K}^\dagger$*. Let* M_x *be a positive integer. Then* PHF *is* $(\mathsf{hashkg}^\dagger, M_x, 0)$*-universally-translation-indistinguishable.*

Proof. Both $\mathsf{hashkg}^\dagger(\Lambda)$ and $\mathsf{hashkg}^\dagger(\Lambda) + x \cdot \mathsf{hashkg}^\dagger(\Lambda)$ are uniform group elements in $\mathcal{K}^\dagger$. □

5.4.4 IPFE-CCA Friendliness

We now regroup the 3 properties we have just defined under the *IPFE-CCA friendliness* property. It is used as a shorthand for the sake of readability and regroups projection key homomorphism, universal translation indistinguishability, and 2-universality on a slight modification of the PHF.

Definition 5.4.5 (IPFE-CCA Friendliness)**.** *A tag-based projective hash function* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *is* $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{2\mathrm{u}}, M_x, \varepsilon^\dagger_{\mathrm{uti}})$-IPFE-CCA-friendly *for a (not necessarily polynomial time) algorithm* $\mathsf{hashkg}'^\dagger$ *taking as input* Λ *and outputting a hashing key* $\mathsf{hk}^\dagger$ *in some set* $\mathcal{K}'^{*\dagger} \subseteq \mathcal{K}$*, and for a positive integer* M_x*, for a subset* $\Sigma^\dagger$ *of* $\mathbb{Z}$*, and for a positive integer* M_x*, if* $\mathsf{PHF}^\dagger$ *is key-homomorphic, projection-key-homomorphic,* $(\mathsf{hashkg}'^\dagger, M_x, \varepsilon^\dagger_{\mathrm{uti}})$*-universally-translation-indistinguishable and if for any* $t \in \Sigma^\dagger$*, the PHF* $(t \cdot \mathsf{hashkg}'^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *is* $\varepsilon^\dagger_{2\mathrm{u}}$*-2-universal, where the algorithm* $t \cdot \mathsf{hashkg}'^\dagger$ *runs* $\mathsf{hashkg}'^\dagger$ *and multiplies the output by* t*.*

Important Particular Case: Key Uniformity. For many key-homomorphic PHFs, the output of $\mathsf{hashkg}^\dagger$ is actually uniform over the group $\mathcal{K}^\dagger$. In this case, we have the following lemma which proves IPFE-CCA friendliness from 2-universality.

Lemma 5.4.6. *Let* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *be a* $\varepsilon^\dagger_{2\mathrm{u}}$*-2-universal tag-based PHF such that the distribution of* $\mathsf{hashkg}^\dagger(\Lambda)$ *is uniform over* $\mathcal{K}^\dagger$*. Then for any* $t \in \mathbb{Z}$*,* $(t \cdot \mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *is* $\varepsilon^\dagger_{2\mathrm{u}}$*-2-universal.*

Proof. Since $\mathsf{hashkg}^\dagger(\Lambda)$ is uniformly distributed, $t \cdot \mathsf{hashkg}^\dagger(\Lambda)$ is as well, so both schemes are equal. □

2-universal tag-based PHFs can be constructed from diverse groups, as in [CS02]. All the constructions in [CS02] are key-homomorphic and projection-key-homomorphic. And for well-chosen parameters, they actually are IPFE-CCA-friendly.

5.5 Inner-Product Functional Encryption Secure Against Chosen-Ciphertext Attacks

In this section, we show a construction of a IND-TBIPFE-CCA secure tag-based inner-product functional encryption scheme. The concrete IND-IPFE-CCA secure inner-product functional

encryption scheme can then be obtained by Construction 3.3.5. This construction is an enhancement of the previous IND-IPFE-CPA secure scheme which uses also ℓ tag-based projective hash functions in order to ensure the non-malleability of the scheme.

5.5.1 Construction and Correctness

As our construction is closely related to the previous one, and the intuition is about the same, we first highlight the changes made to reach IND-IPFE-CCA security on Figure 5.7, before formally presenting the construction and the proof of its correctness. The idea is that the one-time signature prevents the use of the same tag for the challenge ciphertext, and a ciphertext generated by the adversary. Then, the security of the tag-based IPFE will convey to the tagless IPFE.

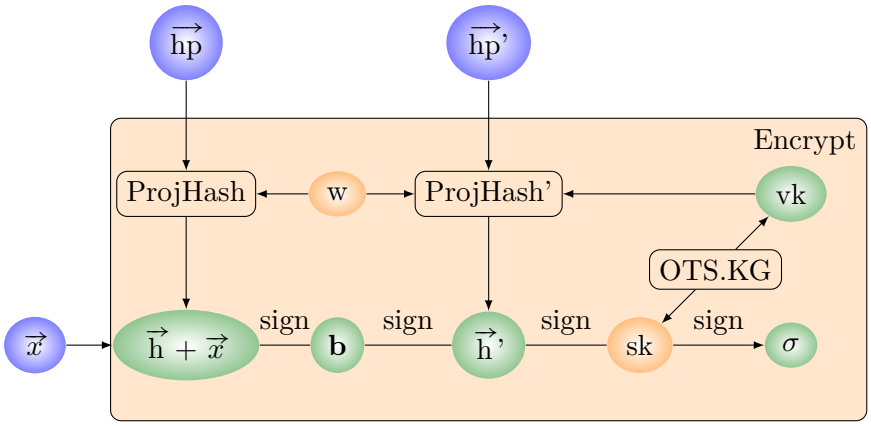

Figure 5.7: Encrypt algorithm for our inner-product encryption scheme secure against chosen-ciphertext attacks. We use a one-time signature and a tag-based projective hash function to prevent malleability and detect ill-formed ciphertext and prevent them from being decrypted.

Let us now define our construction of a IND-TBIPFE-CCA secure tag-based inner-product functional encryption scheme.

We suppose that we have a $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, x, \varepsilon_{\text{ti}})$-IPFE-CPA-friendly projective hash function $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ and a $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{\text{2u}}, M_x, \varepsilon^\dagger_{\text{uti}})$-IPFE-CCA-friendly projective hash function $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ for the subset membership problem $\mathbf{P}$. Let $\mathcal{R}$ be the ring $\mathbb{Z}$ or $\mathbb{Z}_{M_\perp}$, let ℓ be a positive integer parameter corresponding to the length of the message and key vectors, and let $\mathcal{M}_y$ and $\mathcal{M}_x$ be two subsets of $\mathcal{R}^\ell$. We always suppose ℓ to be polynomial in the security parameter κ.

We suppose that Condition 1 is satisfied, in addition to the following new condition.

Condition 2. *Using the above notation:*

1. *if $\mathcal{R} = \mathbb{Z}_{M_\perp}$, the order of any hashing key $\mathsf{hk} \in \mathcal{K}^\dagger$ divides $M_\perp$; and*
2. *for any $\vec{y} \in \mathcal{M}_y$ and $\vec{x} \in \mathcal{M}_x$, $\langle \vec{y}, \vec{x} \rangle \in \Sigma^\dagger \cup \{0\} \subseteq \mathcal{R}$.*

Construction 5.5.1 (Tag-Based Inner-Product Functional Encryption secure against Chosen–Ciphertext Attacks from Projective Hash Functions)**.** *Let $\mathbf{P}$ be a subset membership problem.*

Let $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CPA-friendly PHF, and* $\mathsf{PHF}^\dagger = (\mathsf{hashkg}^\dagger, \mathsf{projkg}^\dagger, \mathsf{hash}^\dagger, \mathsf{projhash}^\dagger)$ *be a* $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{\mathrm{2u}}, M_x, \varepsilon^\dagger_{\mathrm{uti}})$*-IPFE-CCA-friendly tag-based PHF. We assume that Conditions 1 and 2 are satisfied. We define our tag-based inner-product functional encryption scheme* $\mathcal{TBFE}$ *as follows.*

- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (msk, mpk)$. *On input security parameter* κ *and functionality parameter* ℓ*, samples* $\Lambda \xleftarrow{\$} I_\kappa$*, sets* $pp = (\kappa, \ell, \Lambda)$*, computes for all* i *in* $[\ell]$

$$\mathsf{hk}_i \xleftarrow{\$} \mathsf{hashkg}(\Lambda)\,, \qquad \mathfrak{hp}_i \leftarrow \mathsf{projkg}(\mathsf{hk}_i)\,,$$
$$\mathsf{hk}^\dagger_i \xleftarrow{\$} \mathsf{hashkg}^\dagger(\Lambda)\,, \qquad \mathfrak{hp}^\dagger_i \leftarrow \mathsf{projkg}^\dagger(\mathsf{hk}^\dagger_i)\,,$$

  and returns master secret key $msk = (pp, \overrightarrow{\mathsf{hk}}, \overrightarrow{\mathsf{hk}^\dagger})$ *and master public key* $mpk = (pp, \overrightarrow{\mathfrak{hp}}, \overrightarrow{\mathfrak{hp}^\dagger})$*;*

- $\mathsf{KeyDer}(msk, \vec{y}) \mapsto sk_{\vec{y}}$. *On input master secret key* msk*, and key* $\vec{y} \in \mathcal{M}_y$*, computes*

$$\mathsf{hk}_{\vec{y}} \leftarrow \langle \vec{y}, \overrightarrow{\mathsf{hk}} \rangle\,, \qquad \mathsf{hk}^\dagger_{\vec{y}} \leftarrow \langle \vec{y}, \overrightarrow{\mathsf{hk}^\dagger} \rangle\,,$$

  and outputs user secret key $sk_{\vec{y}} = (\mathsf{hk}_{\vec{y}}, \mathsf{hk}^\dagger_{\vec{y}})$*;*

- $\mathsf{Encrypt}(\tau, mpk, \vec{x}) \mapsto ct_{\vec{x}}$. *On input master public key* mpk*, and plaintext* $\vec{x} \in \mathcal{M}_x$*, samples a random pair* $(\mathfrak{b}, w) \in \varrho$*, computes for all* i *in* $[\ell]$

$$ct_i \leftarrow \mathsf{projhash}(\mathfrak{hp}_i, \mathfrak{b}, w) + x_i \cdot \mathfrak{g}_\perp\,, \qquad ct^\dagger_i \leftarrow \mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger_i, \mathfrak{b}, w, \tau)\,,$$

  and outputs ciphertext $ct_{\vec{x}} = (\mathfrak{b}, \vec{ct}, \vec{ct}^{\,\dagger})$*;*

- $\mathsf{Decrypt}(\tau, sk_{\vec{y}}, ct_{\vec{x}}) \mapsto m$ *or* $\perp$. *On input user secret key* $sk_{\vec{y}}$*, and ciphertext* $ct_{\vec{x}}$*, checks that*

$$\langle \vec{y}, \vec{ct}^{\,\dagger} \rangle = \mathsf{hash}^\dagger(\mathsf{hk}^\dagger_{\vec{y}}, \mathfrak{b}, \tau)$$

  and returns $\perp$ *if it fails, otherwise computes*

$$ct_{\langle \vec{x}, \vec{y} \rangle} \leftarrow \langle \vec{y}, \vec{ct} \rangle - \mathsf{hash}(\mathsf{hk}_{\vec{y}}, \mathfrak{b})$$

  and returns $\log_{\mathfrak{g}_\perp} ct_{\langle \vec{x}, \vec{y} \rangle}$.

Correctness. The correctness of our scheme follows from correctness and key homomorphism of both PHF and $\mathsf{PHF}^\dagger$ and from the fact that $\mathfrak{g}_\perp$ has order $M_\perp$. The proof is similar to the one for Theorem 5.3.2 with the additional remark:

$$\begin{aligned}\langle \vec{y}, \vec{ct}^{\,\dagger} \rangle &= \sum_{i=1}^{\ell} \vec{y}_i \cdot (\mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger_i, \mathfrak{b}, w, \tau)) = \sum_{i=1}^{\ell} \vec{y}_i \cdot \mathsf{hash}^\dagger(\mathsf{hk}^\dagger_i, \mathfrak{b}, \tau) \\ &= \mathsf{hash}^\dagger(\sum_{i=1}^{\ell} \vec{y}_i \cdot \mathsf{hk}^\dagger_i, \mathfrak{b}, \tau) = \mathsf{hash}^\dagger(\mathsf{hk}^\dagger_{\vec{y}}, \mathfrak{b}, \tau)\ .\end{aligned}$$

5.5.2 Security Against Chosen-Ciphertext Attacks

We have the following security theorem.

Theorem 5.5.2. *Let* $\mathbf{P}$ *be a subset membership problem. Let* $\mathsf{PHF} = (\mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$ *be a* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CPA-friendly projective hash function,* $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{\mathrm{2u}}, M_x, \varepsilon^\dagger_{\mathrm{uti}})$*-IPFE-CCA-friendly tag-based projective hash function, and let* $\mathcal{TBFE}$ *be the tag-based inner-product functional encryption scheme defined in Construction 5.5.1. We assume that Conditions 1 and 2 are satisfied. Then* $\mathcal{TBFE}$ *is* IND-TBIPFE-CCA *secure.*

More precisely, if there exists an adversary $\mathscr{A} = \mathscr{A}_{\mathcal{TBFE}}$ *that has advantage* $\varepsilon_{\mathscr{A}}$ *in breaking the* IND-TBIPFE-CCA *security of* $\mathcal{TBFE}$*, then there exists an attacker* $\mathscr{B}$ *that runs in approximately the same time and that has advantage* $\varepsilon_{\mathscr{B}}$ *in breaking the* $(\mathcal{L}, \bar{\mathcal{L}})$*-indistinguishability, such that*

$$\varepsilon_{\mathscr{A}} \leq 2 \cdot \varepsilon_{\mathscr{B}} + \ell \cdot |\Delta\mathcal{M}_x| \cdot (\varepsilon_{\mathrm{ti}} + 2 \cdot \varepsilon^\dagger_{\mathrm{uti}}) + 2 \cdot q_{dec} \cdot |\Delta\mathcal{M}_x| \cdot \varepsilon^\dagger_{\mathrm{2u}}\,,$$

where q_{dec} *is the number of queries to the decryption oracle.*

Before proving the theorem, we one again give a remark on the precise advantage of the adversary in the reduction.

Remark 5.5.3. *In addition to the exponential loss* $\ell \cdot |\Delta\mathcal{M}_x| \cdot (\varepsilon_{\mathrm{ti}} + 2 \cdot \varepsilon^\dagger_{\mathrm{uti}})$ *similar to the one for the generic* IND-IPFE-CPA *construction (Theorem 5.3.2), there is an addition exponential loss in the security proof in the term* $2q_{dec}|\Delta\mathcal{M}_x|\varepsilon^\dagger_{\mathrm{2u}}$*. We point out however that the resulting requirement that* $|\Delta\mathcal{M}_x|\varepsilon^\dagger_{\mathrm{2u}}$ *is negligible in the security parameter can easily be achieved: given a* $\varepsilon^\dagger_{\mathrm{2u}}$*-2-universal PHF, we can get a* $(\varepsilon^\dagger_{\mathrm{2u}})^\nu$*-2-universal PHF, by repeating it* ν*-times in parallel. This transformation preserves IPFE-CCA friendliness. We emphasize that the resulting key and ciphertext sizes remain polynomial in the security parameter* κ*, and that we do not rely on complexity leveraging nor subexponential assumptions (see Remark 5.3.3).*

Furthermore, as for the IND-IPFE-CPA *construction from translation-indistinguishable key-homomorphic PHF in Section 5.3, if we only consider a selective version of* IND-TBIPFE-CCA *security where the adversary announces* $\vec{x}_0$ *and* $\vec{x}_1$ *before receiving the public key, then we would not have this factor* $|\Delta\mathcal{M}_x|$*.*

We now resume with the proof of the theorem.

Proof. We first give some intuition about the proof: let us analyze the decryption queries made by the adversary $\mathscr{A}$, to give some intuition. Let $(\vec{y}\,'_j, \tau'_j, (\mathfrak{b}'_j, \vec{ct}\,'_j, \vec{ct}\,'^\dagger_j))$ be the jth decryption query and let $(\vec{y}_1, \ldots, \vec{y}_{i_j})$ be the key queries made before this decryption query. We now split the analysis in two cases. **Case** $\vec{y}\,'_j \in \mathrm{Span}(\vec{y}_1, \ldots, \vec{y}_{i_j})$**.** ($\mathrm{Span}(\vec{y}_1, \ldots, \vec{y}_{i_j})$ denotes the linear span of the vectors $(\vec{y}_1, \ldots, \vec{y}_{i_j})$) In this case, $\mathscr{A}$ can derive the key for $\vec{y}_j$ and do the decryption anyway, so there is nothing to hide. **Case** $\vec{y}\,'_j \notin \mathbf{span}(\vec{y}_1, \ldots, \vec{y}_{i_j})$**.** In this case, we argue that if $\mathfrak{b}'_j \notin \mathcal{L}$, then the decryption oracle will output $\perp$ with probability $1 - \varepsilon^\dagger_{\mathrm{univ}}$: by definition of the 2-universality of $\mathsf{PHF}^\dagger$, we have that conditioned on the value of $\vec{\mathfrak{hp}}^\dagger$ and on the value of the challenge ciphertext $(\mathfrak{b}, \vec{ct}, \vec{ct}^{\,\dagger})$ if it was already generated:

$$\Pr[\mathsf{hash}^\dagger(\mathsf{hk}^\dagger_{\vec{y}'_j}, \mathfrak{b}', \tau') = \langle \vec{y}'_j, \vec{ct}\,'^\dagger \rangle] \leq \varepsilon^\dagger_{\mathrm{univ}}.$$

Unfortunately, the fact that the adversary can adaptively choose the vectors $\vec{y}_i$ and $\vec{y}'_j$ in particular makes the above intuition hard to directly translate into a formal valid proof. To deal with adaptivity, we guess $\mathbf{x}^* = \vec{x}_1 - \vec{x}_0$ before using 2-universality. That is why we loose a factor $\Delta\mathcal{M}_x$ in the security reduction.

Let us now do a formal proof by games. We use the same notation as in the proof of Theorem5.3.2. The first four games are basically the same.

$Game_0$ corresponds to the actual IND-TBIPFE-CCA scheme. The challenge ciphertext is $(\mathfrak{b}, \vec{ct}, \vec{ct}^{\dagger})$.

In $Game_1$, we change the way of computing the ciphertext. More precisely, we now compute ct_i as follows using hk_i instead of $\mathfrak{hp}_i$ and w:

$$ct_i \leftarrow \mathsf{hash}(\mathsf{hk}_i, \mathfrak{b}) + x_i \cdot \mathfrak{g}_\perp$$
$$ct_i^\dagger \leftarrow \mathsf{hash}^\dagger(\mathsf{hk}_i^\dagger, \mathfrak{b}, \tau) \ ,$$

which is perfectly indistinguishable thanks to the correctness of the PHF. Hence, $\mathsf{Adv}[\Delta_1] = 0$.

In $Game_2$, we sample $\mathfrak{b}$ from $\bar{\mathcal{L}}$ instead of from $\mathcal{L}$. We can construct an attacker $\mathscr{B}$ that runs in approximately the same time as $\mathscr{A}$ such that $\mathsf{Adv}[\Delta_2] \leq \varepsilon_{\mathscr{B}}$, where $\mathscr{B}$ is the advantage of $\mathscr{B}$ against $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability.

In $Game_3$, we pick a uniform random value $\mathbf{x}^* \in \Delta\mathcal{M}_x$ at the beginning and abort if $\vec{x}_1 - \vec{x}_0 \neq \mathbf{x}^*$ and make the adversary win with probability $\frac{1}{2}$ in that case. As in the proof of Theorem 5.3.2, we have

$$\mathsf{Adv}[Game_2] = |\Delta\mathcal{M}_x| \cdot \mathsf{Adv}[Game_3] \ .$$

In $Game_4$, we also abort if the adversary queries a vector $\vec{y} \in \mathcal{M}_y$ to the key oracle such that $\langle \vec{y}, \mathbf{x}^* \rangle \neq 0$. This game is perfectly indistinguishable from the previous one, as the adversary cannot ask for a key for $\vec{y} \in \mathcal{M}_y$ such that $\langle \vec{y}, \vec{x}_0 \rangle \neq \langle \vec{y}, \vec{x}_1 \rangle$. Hence, $\mathsf{Adv}[\Delta_4] = 0$.

In the proof of Theorem 5.3.2, we then proved that $\mathsf{Adv}[Game_4] = 0$ using strong diversity and translation indistinguishability. But here, we cannot apply strong diversity directly, as the hashing keys $\vec{\mathsf{hk}}$ are still used to answer the queries to the decryption oracle and too much information might leak from there. To conclude the proof, we need more games.

In $Game_5$, we sample the hashing keys $\mathsf{hk}_i^\dagger$ as follows:

$$\vec{\mathsf{hk}}^\dagger \leftarrow \vec{\mathsf{hk}}''^\dagger + \mathbf{x}^* \cdot \mathsf{hk}'^\dagger \text{ with } \begin{cases} \mathsf{hk}_i''^\dagger \xleftarrow{\$} \mathsf{hashkg}^\dagger(\Lambda) & \text{for } i = 1, \ldots, \ell; \\ \mathsf{hk}'^\dagger \xleftarrow{\$} \mathsf{hashkg}'^\dagger(\Lambda) \ . \end{cases}$$

In the previous game, $\vec{\mathsf{hk}}^\dagger$ could be seen as being generated as $\vec{\mathsf{hk}}^\dagger \leftarrow \vec{\mathsf{hk}}''^\dagger$ Thanks to universal translation indistinguishability of $\mathsf{PHF}^\dagger$, using an hybrid argument, we have:

$$\mathsf{Adv}[\Delta_5] \leq \ell \cdot \varepsilon_{\text{uti}}^\dagger \ .$$

We remark that in $Game_5$, by projection key homomorphism:

$$\vec{\mathfrak{hp}}_i^\dagger = \mathsf{projkg}^\dagger(\mathsf{hk}_i^\dagger) = \mathsf{projkg}^\dagger(\vec{\mathsf{hk}}_i''^\dagger) + x_i^* \cdot \mathsf{projkg}^\dagger(\mathsf{hk}'^\dagger) \ .$$

Furthermore, for any vector $\vec{y} \in \mathcal{M}_y$ such that $\langle \vec{y}, \mathbf{x}^* \rangle = 0$, the corresponding secret key $msk_{\vec{y}} = (\mathsf{pp}, \mathsf{hk}_{\vec{y}}, \mathsf{hk}_{\vec{y}}^\dagger)$ is such that:

$$\mathsf{hk}_{\vec{y}}^\dagger = \langle \vec{y}, \vec{\mathsf{hk}}^\dagger \rangle = \langle \vec{y}, \vec{\mathsf{hk}}''^\dagger \rangle + \langle \vec{y}, \mathbf{x}^* \rangle \cdot \mathsf{hk}'^\dagger = \langle \vec{y}, \vec{\mathsf{hk}}''^\dagger \rangle \ .$$

These keys therefore do not reveal any information about $\mathsf{hk}'^{\dagger}$. We recall that the only places we compute a key $msk_{\vec{y}}$ are in response to a query to a key oracle or to the decryption oracle. In the first case, if there is no abort, we always have $\langle \vec{y}, \mathbf{x}^* \rangle = 0$.

Furthermore, we remark that the vector $\vec{ct}^{\dagger}$ of the challenge ciphertext is such that:

$$ct_i^{\dagger} = \mathsf{hash}^{\dagger}(\mathsf{hk}_i^{\dagger}, \mathfrak{b}, \tau) = \mathsf{hash}^{\dagger}(\mathsf{hk}_i''^{\dagger}, \mathfrak{b}, \tau) + x_i^* \cdot \mathsf{hash}^{\dagger}(\mathsf{hk}'^{\dagger}, \mathfrak{b}, \tau)\,,$$

by key homomorphism. This means that the only information that the attacker sees about $\mathsf{hk}'^{\dagger}$ comes from:

1. its projection key $\mathfrak{hp}'^{\dagger} = \mathsf{projkg}^{\dagger}(\mathsf{hk}'^{\dagger})$,
2. the keys $msk_{\vec{y}}$ used by the decryption oracle on a query associated to a vector $\vec{y} \in \mathcal{M}_y$ such that $\langle \vec{y}, \mathbf{x}^* \rangle \neq 0$,
3. the hash value $\mathsf{hash}^{\dagger}(\mathsf{hk}'^{\dagger}, \mathfrak{b}, \tau)$ used to compute the challenge ciphertext.

In $Game_6$, we change the decryption oracle as follows: on input query $(\vec{y}'_j, \tau'_j, (\mathfrak{b}'_j, \vec{ct}'_j, \vec{ct}'^{\dagger}_j))$:

- if $\langle \vec{y}'_j, \mathbf{x}^* \rangle = 0$, it behaves as usual using $\overrightarrow{\mathsf{hk}}_{\vec{y}'_j} = \langle \vec{y}'_j, \overrightarrow{\mathsf{hk}} \rangle$ and $\overrightarrow{\mathsf{hk}}^{\dagger}_{\vec{y}'_j} = \langle \vec{y}'_j, \overrightarrow{\mathsf{hk}}^{\dagger} \rangle$ to decrypt the ciphertext $(\mathfrak{b}'_j, \vec{ct}'_j, \vec{ct}'^{\dagger}_j)$ with the tag τ'_j; we recall that this does not reveal any information about $\mathsf{hk}'^{\dagger}$;
- otherwise:
  - if $\mathfrak{b}'_j \in \mathcal{L}$, it finds a witness w'_j,[1] and checks that

    $$\langle \vec{y}, \vec{ct}'^{\dagger}_j \rangle = \mathsf{projhash}^{\dagger}(\langle \vec{y}'_j, \overrightarrow{\mathfrak{hp}}^{\dagger} \rangle, \mathfrak{b}'_j, w'_j, \tau)$$

    instead of checking that:

    $$\langle \vec{y}, \vec{ct}'^{\dagger}_j \rangle = \mathsf{hash}^{\dagger}(\mathsf{hk}^{\dagger}_{\vec{y}'_j}, \mathfrak{b}'_j, \tau)\ .$$

    This is perfectly indistinguishable by correctness of $\mathsf{PHF}^{\dagger}$ and shows that this does not reveal any information about $\mathsf{hk}'^{\dagger}$ beyond $\mathfrak{hp}'^{\dagger}$;
  - if $\mathfrak{b}'_j \in \mathcal{X} \setminus \mathcal{L}$, then it rejects the ciphertext. This is statistically indistinguishable thanks to the 2-universality.

This game is statistically indistinguishable from the previous one using an hybrid argument changing the way the decryption oracle behaves query by query. We have: $\mathsf{Adv}[\Delta_6] \leq 2 \cdot q_{\mathrm{dec}} \cdot \varepsilon^{\dagger}_{2\mathrm{u}}$.

In $Game_7$, we change again the decryption oracle as follows, on input query $(\vec{y}'_j, \tau'_j, (\mathfrak{b}'_j, \vec{ct}'_j, \vec{ct}'^{\dagger}_j))$:

- if $\langle \vec{y}'_j, \mathbf{x}^* \rangle = 0$, it behaves as usual;
- otherwise:

[1] We recall that the reduction is statistical here.

– if $\mathfrak{b}'_j \in \mathcal{L}$, it finds a witness w'_j, it checks $\langle \vec{y}, \vec{ct}'^{\dagger}_j \rangle$ as before, but it now computes $ct_{\vec{x}}$ as follows:

$$ct_{\vec{x}} \leftarrow \langle \vec{y}'_j, \vec{ct}'_j \rangle - \mathsf{projhash}(\langle \vec{y}'_j, \overrightarrow{\mathfrak{hp}} \rangle, \mathfrak{b}'_j, w'_j)$$

instead of

$$ct_{\vec{x}} \leftarrow \langle \vec{y}'_j, \vec{ct}'_j \rangle - \mathsf{hash}(\mathsf{hk}, \mathfrak{b}'_j) \ ,$$

which is perfectly indistinguishable by correctness of PHF.

– if $\mathfrak{b}'_j \in \mathcal{X} \setminus \mathcal{L}$, then it rejects the ciphertext as in the previous game.

We now remark that if we forget about the part $\vec{ct}^{\dagger}$ of all the ciphertexts, we are exactly as in $Game_4$ of the proof of Theorem 5.3.2. The only information the adversary learns about $\overrightarrow{\mathsf{hk}}$ is $\langle \vec{y}, \overrightarrow{\mathsf{hk}} \rangle$, for vectors $\vec{y}$ such that $\langle \vec{y}, \mathbf{x}^* \rangle = 0$ (if $\mathbf{x}^*$ was guessed correctly). We conclude as in Theorem 5.3.2 that $\mathsf{Adv}[Game_7] \leq \ell \cdot \varepsilon_{\mathrm{ti}}$.

The bound of the theorem follows.

□

Chapter 6

Instantiations based on the Decisional Diffie-Hellman Assumption

In this chapter, we instantiate the generic constructions presented from public key encryption and from projective hash function with concrete constructions based on the decisional Diffie-Hellman assumption.
We give 3 concrete instantiations of inner-product functional encryption that respectively reach s-IND-IPFE-CPA security, IND-IPFE-CPA security and IND-IPFE-CCA security, and compare their trade-offs between efficiency and security.
The main downside of this construction is that the decryption is expensive: it requires the computation of a discrete logarithm in a group where it is in general hard. So the message space has to be restricted. However, the schemes are conceptually simple and are modular. The first construction is the original one that we proposed in [ABDP15b], while the two others are taken from [BBL17].

Contents

6.1 The Decisional Diffie Hellman Assumption and Subset Membership Problem . . . 86
6.1.1 The Decisional Diffie Hellman Assumption . . . 86
6.1.2 The Matrix Decisional Diffie-Hellman Assumption . . . 86
6.2 ElGamal Public Key Encryption Scheme . . . 87
6.2.1 Additively Homomorphic ElGamal . . . 87
6.2.2 Additional Properties of the Additive ElGamal Public Key Encryption Scheme . . . 88
6.3 Projective Hash Functions Based on the Decisional Diffie-Hellman Assumptions . . . 88
6.3.1 IPFE-CPA-Friendly Projective Hash Functions . . . 88
6.3.2 IPFE-CCA-Friendly Projective Hash Functions . . . 90
6.4 Inner-Product Functional Encryption Schemes Based on the Decisional Diffie-Hellman Assumption . . . 92
6.4.1 Scheme Secure Against Selective Chosen-Plaintext Attacks . . . 92
6.4.2 Scheme Secure Against Adaptive Chosen-Plaintext Attacks . . . 94
6.4.3 Scheme Secure Against Chosen-Ciphertext Attacks . . . 95

6.1 The Decisional Diffie Hellman Assumption and Subset Membership Problem

In this section, we recall the *decisional Diffie-Hellman* (DDH) assumption and its generalization to matrix decisional Diffie-Hellman assumption. Then we present the subset membership problem whose indistinguishability is based on the DDH assumption which will be used to construct key homomorphic projective hash functions and thus IND-IPFE-CCA secure schemes.

6.1.1 The Decisional Diffie Hellman Assumption

The *decisional Diffie-Hellman* (DDH) assumption has been widely used in public key cryptography since the original two party key exchange that introduced it [DH76] and the very simple public key encryption scheme by ElGamal [ElG85] that we will use to construct an inner-product functional encryption scheme. It is an assumption that can only hold in groups where discrete logarithm are hard, and there are plenty of groups where it is believed to hold. For example, it is used in elliptic curves-based cryptography that is deployed for the security of the Internet.

The Decisional Diffie-Hellman Assumption. Let GroupGen be a probabilistic polynomial-time algorithm that takes as input a security parameter 1^κ, and outputs a triplet $(\mathbb{G}, q, \mathfrak{g})$ where $\mathbb{G}$ is a group of order q that is generated by $\mathfrak{g} \in \mathbb{G}$, and q is a κ-bit prime number. Then, the *Decisional Diffie-Hellman (DDH) assumption* states that the tuples $(\mathfrak{g}, \mathfrak{g}^a, \mathfrak{g}^b, \mathfrak{g}^{ab})$ and $(\mathfrak{g}, \mathfrak{g}^a, \mathfrak{g}^b, \mathfrak{g}^c)$ are computationally indistinguishable, where $(\mathbb{G}, q, \mathfrak{g}) \leftarrow \mathsf{GroupGen}(1^\lambda)$, and $a, b, c \in \mathbb{Z}_q$ are chosen independently and uniformly at random.

We also define the following subset membership problem that we will use for our projective hash functions based on DDH.

DDH-Based Subset Membership Problem. Let $\mathbb{G}$ be an additive cyclic group of prime order q, let $\mathcal{X} = \mathbb{G}^2$, let $\mathcal{L}$ be the subgroup of $\mathcal{X}$ generated by $\mathfrak{g} = (\mathfrak{g}_1, \mathfrak{g}_2)^\intercal \in \mathbb{G}^2$, where $\mathfrak{g}_i$ are random generators of $\mathbb{G}$, and let $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$. A witness $w \in \mathcal{W} = \mathbb{Z}_q$ for $\mathfrak{b} \in \mathcal{L}$ is such that $\mathfrak{b} = w\mathfrak{g}$. In other words, we have $\mathcal{W} = \mathbb{Z}_q$ and $\varrho = \{(w \cdot \mathfrak{g},\, w) \,:\, w \in \mathbb{Z}_q\}$. We set $\Lambda = (\mathbb{G}, \mathfrak{g})$.

This defines a subset membership problem, whose $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability property is equivalent to the DDH assumption.

6.1.2 The Matrix Decisional Diffie-Hellman Assumption

For some interesting cryptographic cyclic groups, such as groups with a symmetric pairing, the DDH assumption does not hold. That is why weaker assumptions, such as the decisional linear assumption (DLIN, [BBS04]), have been considered. More recently, Escala et al. introduced the *Matrix Diffie-Hellman* (*MDDH*) assumption family [EHK+13] that generalizes DDH and its weaker variants like DLIN. Let us recall the MDDH assumption families in the context of subset membership problems.

MDDH-Based Subset Membership Problem.

Let $\mathbb{G}$ be a cyclic group of prime order q. Let $\mathcal{D}$ be a distribution of matrices in $\mathbb{G}^{t \times d}$ with $d < t$ being two positive integers. Let $\mathfrak{g} \xleftarrow{\$} \mathcal{D}$. Let $\mathcal{X} = \mathbb{G}^t$. Let $\mathcal{L}$ be the subgroup of $\mathcal{X}$ generated by the columns of $\mathfrak{g}$ and let $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$. A witness $\boldsymbol{w} \in \mathcal{W} = \mathbb{Z}_q^d$ for $\mathfrak{b} \in \mathcal{L}$ is such

that $\mathfrak{b} = \mathfrak{g} \cdot \boldsymbol{w}$. In other words, we have $\mathcal{W} = \mathbb{Z}_q^d$ and $\varrho = \{(\mathfrak{g} \cdot \boldsymbol{w}, \boldsymbol{w}) : \boldsymbol{w} \in \mathbb{Z}_q^d\}$. We set $\Lambda = (\mathbb{G}, \mathfrak{g})$.

This defines a subset membership problem, whose $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability property corresponds to the $\mathcal{D}$-MDDH assumption.

When $d = 1$, $t = 2$, and $\mathcal{D}$ is the uniform distribution over vectors of two generators of $\mathbb{G}$, then we get back the DDH-based subset membership problem.

6.2 ElGamal Public Key Encryption Scheme

We now present the public key encryption scheme whose security is based on the DDH assumption, and then we show that it satisfies the properties defined in Section 4.2.1.

6.2.1 Additively Homomorphic ElGamal

The standard ElGamal [ElG85] public key cryptosystem doesn't directly satisfy the properties that we need. However, as it is multiplicatively homomorphic, it is easy to transform it in a way that satisfy our requirements: instead of directly encoding the message as a group element, we chose a generator $\mathfrak{g}$ and encode a message m as $m \cdot \mathfrak{g}$. This transformed the multiplicative homomorphism into an additive homomorphism, which is exactly what we wanted for our construction. The downside is that now, the decryption requires the computation of a discrete logarithm in base $\mathfrak{g}$. As the basis of the discrete logarithm is always the same, it is possible to speed up the decryption process by having precomputed discrete logarithm tables, but this still only allows decryption in a fixed polynomial-sized set.

We now recall the scheme with additive notation:

Construction 6.2.1 (Additive ElGamal Public Key Encryption)**.** *The additive ElGamal public key encryption scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is defined as follows.*

- $\mathsf{Setup}(1^\kappa) \mapsto (pp, sk, pk)$. *On input security parameter* κ *samples* $(\mathbb{G}, q, \mathfrak{g}) \xleftarrow{\$} \mathsf{GroupGen}(1^\kappa)$ *and* $s \xleftarrow{\$} \mathbb{Z}_q$*, and outputs public parameters* $pp = (\mathbb{G}, \mathfrak{g})$*, secret key* $sk = s$*, and public key* $pk \leftarrow s \cdot \mathfrak{g}$*;*

- $\mathsf{Encrypt}(pp, pk, m) \mapsto ct$. *On input public parameter* pp*, public key* pk*, and plaintext* m*, samples* $r \xleftarrow{\$} \mathbb{Z}_q$*, computes* $ct_0 \leftarrow r \cdot \mathfrak{g}$ *and* $ct_1 \leftarrow r \cdot pk + m \cdot \mathfrak{g}$ *outputs ciphertext* $ct = (ct_0, ct_1)$*;*

- $\mathsf{Decrypt}(pp, sk, ct) \mapsto m$ *or* $\perp$. *On input public parameters* pp*, secret key* sk*, and ciphertext* ct*, computes and returns* $m = \log_{\mathfrak{g}}(ct_1 - sk \cdot ct_0)$ *outputs a message* m *or an error symbol* $\perp$*.*

Correctness. If the message belongs in a fixed polynomial range that allows the efficient computation of the discrete logarithm $\log_{\mathfrak{g}}(m \cdot \mathfrak{g})$, the correctness comes directly from the fact that

$$s \cdot r \cdot \mathfrak{g} = r \cdot s \cdot \mathfrak{g}.$$

We recall that there exist generic algorithms to compute the discrete logarithm of an element $t \cdot \mathfrak{g}_\perp$ in $O(\sqrt{|T|})$ group operations, when t is in an interval T; and in $O(T)$ group operations, when t is in an arbitrary subset of $T \subseteq \mathbb{Z}_q$.

6.2.2 Additional Properties of the Additive ElGamal Public Key Encryption Scheme

We now go through the additional properties defined in Section 4.2, and show that the additive ElGamal encryption scheme defined in Construction 6.2.1 satisfies this properties.

Structure. Let q be a prime and $\mathbb{G}$ a group of order q where the DDH assumption is supposed to be hard, and $\mathfrak{g}$ a generator of $\mathbb{G}$. Then, ElGamal's secret key space is the group $(\mathbb{Z}_q, +, 0)$, public key space is the group $(\mathbb{G}, +, 0)$, and the message space is $\mathbb{Z}_q$. An ElGamal ciphertext is of the form $ct = (ct_0 = r \cdot \mathfrak{g}, ct_1 = r \cdot pk + m \cdot \mathfrak{g})$ as required. Thus, $\mathsf{C}(r) = r \cdot \mathfrak{g}$ and $\mathsf{E}(pk, x; r) = r \cdot pk + m \cdot \mathfrak{g}$.

Linear Key Homomorphism. It is easy to see that for any two secret keys $sk_1, sk_2 \in \mathbb{Z}_q$ and any $y_1, y_2 \in \mathbb{Z}_q$, the component-wise linear combination formed by $y_1 sk_1 + y_2 sk_2$ can be computed efficiently only using public parameters, the secret keys sk_1 and sk_2 and the coefficients y_1 and y_2. And this combination $y_1 sk_1 + y_2 sk_2$ also functions as a secret key to a public key that can be computed as $y_1 \cdot pk_1 + y_2 \cdot pk_2 = (y_1 sk_1 + y_2 sk_2) \cdot \mathfrak{g}$, where pk_1 (resp. pk_2) is a public key corresponding to sk_1 (resp. sk_2).

Linear Ciphertext Homomorphism under Shared Randomness. It holds that

$$\begin{aligned}\mathsf{E}(pk_1, x_1; r) \cdot \mathsf{E}(pk_2, x_2; r) &= r \cdot sk_1 \cdot \mathfrak{g} + x_1 \cdot \mathfrak{g} + r \cdot sk_2 \cdot \mathfrak{g} + x_2 \cdot \mathfrak{g} \\ &= r \cdot (sk_1 + sk_2) \cdot \mathfrak{g} + (x_1 + x_2) \cdot \mathfrak{g} \\ &= \mathsf{E}(pk_1 pk_2, x_1 + x_2; r) \ .\end{aligned}$$

Security. The ElGamal encryption scheme remains secure under randomness reuse. In fact, it holds that

$$\mathsf{E}(pk, x; r) = r \cdot (sk \cdot \mathfrak{g}) + m \cdot \mathfrak{g} = sk \cdot (r \cdot \mathfrak{g}) + m \cdot \mathfrak{g}$$

.

Thus, the simplified proof of security applies, hence the construction of two inner-product functional encryption schemes, one being s-IND-IPFE-CPA secure, and the other one reaching adaptive IND-IPFE-CPA security. We will present those schemes in Section 6.4.

6.3 Projective Hash Functions Based on the Decisional Diffie-Hellman Assumptions

We are now ready to present the projective hash function that we will use to construct inner-product functional encryption scheme. We start with the IPFE-CPA-friendly PHF, and then present the IPFE-CCA-friendly one.

6.3.1 IPFE-CPA-Friendly Projective Hash Functions

In this section, we describe IPFE-CPA-friendly PHFs for the subset membership problems described in Section 6.1.

6.3.1.1 Based on Decisional Diffie-Hellman.

Let $\mathbb{G}$ be an additive cyclic group of prime order q, let $\mathcal{X} = \mathbb{G}^2$, let $\mathcal{L}$ be the subgroup of $\mathcal{X}$ generated by $\mathfrak{g} = (\mathfrak{g}_1, \mathfrak{g}_2)^\intercal \in \mathbb{G}^2$, where $\mathfrak{g}_i$ are random generators of $\mathbb{G}$. A witness $w \in \mathcal{W} = \mathbb{Z}_q$ for $\mathfrak{b} \in \mathcal{L}$ is such that $\mathfrak{b} = w \cdot \mathfrak{g}$. We set $\Lambda = (\mathbb{G}, \mathfrak{g})$.

We recall the PHF of Cramer and Shoup [CS01, Section 8.1.1] defined as follows:

- $\mathsf{hashkg}(\Lambda)$ outputs $\mathbf{hk} \xleftarrow{\$} \mathbb{Z}_q^2 = \mathcal{K}$;
- $\mathsf{projkg}(\mathbf{hk})$ outputs $\mathfrak{hp} \leftarrow \mathbf{hk}^\intercal \cdot \mathfrak{g} \in \mathbb{G}$;
- $\mathsf{hash}(\mathbf{hk}, \mathfrak{b})$ outputs $\mathfrak{H} \leftarrow \mathbf{hk}^\intercal \cdot \mathfrak{b} \in \mathbb{G} = \Pi$;
- $\mathsf{projhash}(\mathfrak{hp}, \mathfrak{b}, w)$ outputs $\mathfrak{p}\mathfrak{H} \leftarrow \mathfrak{hp} \cdot w \in \mathbb{G} = \Pi$.

Lemma 6.3.1. *Using above notation, let $\mathfrak{g}_\perp$ an arbitrary generator of $\mathbb{G}$, $M_\perp = q$, M_x be a positive integer, and $\varepsilon_{\mathrm{ti}} = 0$. For any $\mathfrak{b} \in \mathcal{X} \setminus \mathcal{L}$, let $\mathsf{hk}_\perp(\mathfrak{b})$ be defined as follows:*

$$\mathbf{hk}_\perp(\mathfrak{b}) = \frac{\log_{\mathfrak{g}_1} \mathfrak{g}_\perp}{\log_{\mathfrak{g}_1} \mathfrak{b}_1 \cdot \log_{\mathfrak{g}_1} \mathfrak{g}_2 - \log_{\mathfrak{g}_1} \mathfrak{b}_2} \cdot \begin{pmatrix} \log_{\mathfrak{g}_1} \mathfrak{g}_2 \\ -1 \end{pmatrix} \quad \textit{with } \mathfrak{b} = \begin{pmatrix} \mathfrak{b}_1 \\ \mathfrak{b}_2 \end{pmatrix} \in \mathbb{G}^2 \ .$$

Then, the PHF described above is $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$-IPFE-CPA-friendly.

Proof. We first remark that $\mathbf{hk}_\perp(\mathfrak{b})$ is well defined, as $\log_{\mathfrak{g}_1} \mathfrak{b}_1 \cdot \log_{\mathfrak{g}_1} \mathfrak{g}_2 \neq \log_{\mathfrak{g}_1} \mathfrak{b}_2$ since $\mathfrak{b} \notin \mathcal{L}$.

Key Homomorphism is straightforward.

Strong Diversity. Since the space of projection keys is also a group and projkg is a group homomorphism, we can use Lemma 5.2.4. Hence, we just need to prove that $\mathsf{projkg}(\mathbf{hk}_\perp(\mathfrak{b})) = 0$ and $\mathsf{hash}(\mathbf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) = \mathfrak{g}_\perp$. This follows from the following two facts:

$$\mathsf{projkg}(\mathbf{hk}_\perp(\mathfrak{b})) = \frac{\log_{\mathfrak{g}_1} \mathfrak{g}_\perp}{\log_{\mathfrak{g}_1} \mathfrak{b}_1 \cdot \log_{\mathfrak{g}_1} \mathfrak{g}_2 - \log_{\mathfrak{g}_1} \mathfrak{b}_2} \cdot \begin{pmatrix} \log_{\mathfrak{g}_1} \mathfrak{g}_2 & -1 \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{g}_1 \\ \mathfrak{g}_2 \end{pmatrix} \ ,$$

$$\mathsf{hash}(\mathbf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) = \frac{\log_{\mathfrak{g}_1} \mathfrak{g}_\perp}{\log_{\mathfrak{g}_1} \mathfrak{b}_1 \cdot \log_{\mathfrak{g}_1} \mathfrak{g}_2 - \log_{\mathfrak{g}_1} \mathfrak{b}_2} \cdot \begin{pmatrix} \log_{\mathfrak{g}_1} \mathfrak{g}_2 & -1 \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{b}_1 \\ \mathfrak{b}_2 \end{pmatrix} \ .$$

Translation Indistinguishability follows from Lemma 5.2.6. □

6.3.1.2 Based on Matrix Decisional Diffie-Hellman.

Let $\Lambda = (\mathbb{G}, \mathfrak{g})$ be defined as in the MDDH subset membership of Section 6.1. We recall that $\mathfrak{g} \in \mathbb{G}^{t \times d}$, $\mathcal{X} = \mathbb{G}^t$, $\mathcal{L}$ is the subgroup generated by the columns of $\mathfrak{g}$, and $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$. A witness $\boldsymbol{w} \in \mathcal{W} = \mathbb{Z}_q^d$ for $\mathfrak{b} \in \mathcal{L}$ is such that $\mathfrak{b} = \mathfrak{g} \cdot \boldsymbol{w}$.

We recall the PHF defined by Escala et al. in [EHK+13]:

- $\mathsf{hashkg}(\Lambda)$ outputs $\mathbf{hk} \xleftarrow{\$} \mathbb{Z}_q^t = \mathcal{K}$;
- $\mathsf{projkg}(\mathbf{hk})$ outputs $\mathfrak{hp} \leftarrow \mathfrak{g}^\intercal \cdot \mathbf{hk} \in \mathbb{G}^d$;
- $\mathsf{hash}(\mathbf{hk}, \mathfrak{b})$ outputs $\mathfrak{H} \leftarrow \mathbf{hk}^\intercal \cdot \mathfrak{b} \in \mathbb{G} = \Pi$;
- $\mathsf{projhash}(\mathfrak{hp}, \mathfrak{b}, w)$ outputs $\mathfrak{p}\mathfrak{H} \leftarrow \mathfrak{hp}^\intercal \cdot \boldsymbol{w} \in \mathbb{G} = \Pi$.

We can prove the following lemma similarly to Lemma 6.3.1:

Lemma 6.3.2. *Using above notation, let* $\mathfrak{g}_\perp$ *an arbitrary generator of* $\mathbb{G}$*,* $M_\perp = q$*,* M_x *be a positive integer, and* $\varepsilon_{\mathrm{ti}} = 0$*. Let* $\mathsf{hk}_\perp(\mathfrak{b})$ *be an arbitrary vector satisfying* $\mathsf{hk}_\perp(\mathfrak{b})^\intercal \cdot \mathfrak{g} = 0$ *and* $\mathsf{hk}_\perp(\mathfrak{b})^\intercal \cdot \vec{\mathfrak{b}} = \mathfrak{g}_\perp$*, which exists as* $\vec{\mathfrak{b}}$ *is not in the span of the columns of* $\mathfrak{g}$*. Then, the PHF described above is* $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CPA-friendly.*

We skip the proof because is follows the previous one in a very straigthforward manner.

The two PHFs we have just described can thus easily be used to construct IND-IPFE-CPA secure inner-product functional encryption schemes. We present the one based on DDH in Section 6.4.

6.3.2 IPFE-CCA-Friendly Projective Hash Functions

In this section, we describe IPFE-CCA-friendly PHFs for the subset membership problems described in Section 6.1.

DDH Based IPFE-CCA-Friendly PHF. Let $\mathbb{G}$ be a cyclic group of prime order q, let $\mathcal{X} = \mathbb{G}^2$, let $\mathcal{L}$ be the subgroup of $\mathcal{X}$ generated by $\mathfrak{g} = (\mathfrak{g}_1, \mathfrak{g}_2)^\intercal \in \mathbb{G}^2$, where $\mathfrak{g}_i$ are random generators of $\mathbb{G}^*$. A witness $w \in \mathcal{W} = \mathbb{Z}_q$ for $\mathfrak{b} \in \mathcal{L}$ is such that $\mathfrak{b} = w \cdot \mathfrak{g}$. We set $\Lambda = (\mathbb{G}, \mathfrak{g})$.

We first recall the following 2-universal hash from [ABP15]:

- The tag set is $\mathcal{T} = \mathbb{Z}_q$;
- $\mathsf{hashkg}^\dagger(\Lambda)$ outputs $\mathbf{hk}^\dagger \xleftarrow{\$} \mathbb{Z}_q^4 =: \mathcal{K}$;
- $\mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)$ outputs $\mathfrak{hp}^\dagger \leftarrow \begin{pmatrix} \mathfrak{g} & 0 \\ 0 & \mathfrak{g} \end{pmatrix}^\intercal \cdot \mathbf{hk}^\dagger \in \mathbb{G}^2 =: \mathcal{K}_{\mathsf{hp}}$;
- $\mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau)$ outputs $\mathfrak{H}^\dagger \leftarrow \mathbf{hk}^{\dagger\intercal} \cdot \begin{pmatrix} \mathfrak{b} \\ \tau \cdot \mathfrak{b} \end{pmatrix} \in \mathbb{G} =: \Pi$;
- $\mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger, \mathfrak{b}, w, \tau)$ outputs $\mathfrak{pH}^\dagger \leftarrow \mathfrak{hp}^{\dagger\intercal} \cdot \begin{pmatrix} w \\ \tau \cdot w \end{pmatrix} \in \mathbb{G} = \Pi$.

We have the following property.

Lemma 6.3.3. *Using above notation, let* $\mathsf{hashkg}'^\dagger = \mathsf{hashkg}^\dagger$*,* $\Sigma^\dagger = \mathbb{Z}_q$*,* $\varepsilon^\dagger_{2\mathrm{u}} = 1/q$*,* M_x *be a positive integer, and* $\varepsilon^\dagger_{\mathrm{uti}} = 0$*. Then, the PHF described above is a* $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{2\mathrm{u}}, M_x, \varepsilon^\dagger_{\mathrm{uti}})$*-IPFE-CCA-friendly.*

Proof. Key Homomorphism is straightforward.

Projection Key Homomorphism is straightforward.

Universal Translation Indistinguishability follows from Lemma 5.4.4.

2-Universality. Assume that $\mathfrak{b} = \begin{pmatrix} w_1\mathfrak{g}_1 \\ w_2\mathfrak{g}_2 \end{pmatrix}$ and $\mathfrak{b}' = \begin{pmatrix} w'_1\mathfrak{g}_1 \\ w'_2\mathfrak{g}_2 \end{pmatrix}$ for $w'_1 \neq w'_2$,

$$
\begin{aligned}
p_1 := \Pr_{\mathbf{hk}^\dagger}[\mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{H}'^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}', \tau') \wedge \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)] \\
= \Pr_{\mathbf{hk}^\dagger}\left[\mathfrak{H}^\dagger = \mathbf{hk}^{\dagger\intercal} \cdot \begin{pmatrix} \mathfrak{b} \\ \tau\mathfrak{b} \end{pmatrix} \wedge \mathfrak{H}'^\dagger = \mathbf{hk}^{\dagger\intercal} \cdot \begin{pmatrix} \mathfrak{b}' \\ \tau'\mathfrak{b}' \end{pmatrix} \wedge \mathfrak{hp}^\dagger = \mathbf{hk}^{\dagger\intercal} \cdot \begin{pmatrix} \mathfrak{g} & 0 \\ 0 & \mathfrak{g} \end{pmatrix}\right] \\
= \Pr_{\mathbf{hk}^\dagger}\left[\begin{pmatrix} \mathfrak{H}^\dagger & \mathfrak{H}'^\dagger & \mathfrak{hp}^\dagger_1 & \mathfrak{hp}^\dagger_2 \end{pmatrix} = \mathbf{hk}^{\dagger\intercal} \cdot \begin{pmatrix} w_1\mathfrak{g}_1 & w'_1\mathfrak{g}_1 & \mathfrak{g}_1 & 0 \\ w_2\mathfrak{g}_2 & w'_2\mathfrak{g}_2 & \mathfrak{g}_2 & 0 \\ \tau w_1\mathfrak{g}_1 & \tau' w'_1\mathfrak{g}_1 & 0 & \mathfrak{g}_1 \\ \tau w_2\mathfrak{g}_2 & \tau' w'_2\mathfrak{g}_2 & 0 & \mathfrak{g}_2 \end{pmatrix}\right] \\
= \Pr_{\mathbf{hk}^\dagger}\left[\begin{pmatrix} \log_{\mathfrak{g}_1}\mathfrak{H}^\dagger & \log_{\mathfrak{g}_1}\mathfrak{H}'^\dagger & \log_{\mathfrak{g}_1}\mathfrak{hp}^\dagger_1 & \log_{\mathfrak{g}_1}\mathfrak{hp}^\dagger_2 \end{pmatrix} = \mathbf{hk}^{\dagger\intercal} \cdot M_1\right] \quad ,
\end{aligned}
$$

where

$$M_1 = \begin{pmatrix} w_1 & w_1' & 1 & 0 \\ w_2 \log_{\mathfrak{g}_1} \mathfrak{g}_2 & w_2' \log_{\mathfrak{g}_1} \mathfrak{g}_2 & \log_{\mathfrak{g}_1} \mathfrak{g}_2 & 0 \\ \tau w_1 & \tau' w_1' & 0 & 1 \\ \tau w_2 \log_{\mathfrak{g}_1} \mathfrak{g}_2 & \tau' w_2' \log_{\mathfrak{g}_1} \mathfrak{g}_2 & 0 & \log_{\mathfrak{g}_1} \mathfrak{g}_2 \end{pmatrix} .$$

On the other hand,

$$\begin{aligned} p_2 &:= \Pr_{\mathbf{hk}^\dagger}[\mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)] \\ &= \Pr_{\mathbf{hk}^\dagger}\left[\mathfrak{H}^\dagger = \mathbf{hk}^{\dagger\mathsf{T}} \cdot \left(\begin{smallmatrix} \mathfrak{b} \\ \tau\mathfrak{b} \end{smallmatrix}\right) \wedge \mathfrak{hp}^\dagger = \mathbf{hk}^{\dagger\mathsf{T}} \cdot \left(\begin{smallmatrix} \mathfrak{g} & \mathbf{0} \\ \mathbf{0} & \mathfrak{g} \end{smallmatrix}\right)\right] \\ &= \Pr_{\mathbf{hk}^\dagger}\left[\left(\mathfrak{H}^\dagger \quad \mathfrak{hp}_1^\dagger \quad \mathfrak{hp}_2^\dagger\right) = \mathbf{hk}^{\dagger\mathsf{T}} \cdot \begin{pmatrix} w_1\mathfrak{g}_1 & \mathfrak{g}_1 & 0 \\ w_2\mathfrak{g}_2 & \mathfrak{g}_2 & 0 \\ \tau w_1\mathfrak{g}_1 & 0 & \mathfrak{g}_1 \\ \tau w_2\mathfrak{g}_2 & 0 & \mathfrak{g}_2 \end{pmatrix}\right] \\ &= \Pr_{\mathbf{hk}^\dagger}\left[\left(\log_{\mathfrak{g}_1} \mathfrak{H}^\dagger \quad \log_{\mathfrak{g}_1} \mathfrak{hp}_1^\dagger \quad \log_{\mathfrak{g}_1} \mathfrak{hp}_2^\dagger\right) = \log_{\mathfrak{g}_1} \mathbf{hk}^{\dagger\mathsf{T}} \cdot M_2\right] \end{aligned}$$

where

$$M_2 = \begin{pmatrix} w_1 & 1 & 0 \\ w_2 \log_{\mathfrak{g}_1} \mathfrak{g}_2 & \log_{\mathfrak{g}_1} \mathfrak{g}_2 & 0 \\ \tau w_1 & 0 & 1 \\ \tau w_2 \log_{\mathfrak{g}_1} \mathfrak{g}_2 & 0 & \log_{\mathfrak{g}_1} \mathfrak{g}_2 \end{pmatrix} .$$

The column span of M_2 is included in the column span of M_1, because M_1 contains M_2 as a submatrix. Moreover, the second column of M_1 is not in the column span of M_2, so $\mathrm{rank}(M_1) = \mathrm{rank}(M_2) + 1$. Hence, $p_1 \leq p_2/q$. The theorem follows from Lemma 5.4.6. □

We use a slight extension of this PHF because we need an exponentially small security parameter $\varepsilon_{2\mathrm{u}}^\dagger$, due our security reduction. The following PHF can be seen as repeating ν times the PHF of Lemma 6.3.3:

- The tag set is $\mathcal{T} = \mathbb{Z}_q$;
- $\mathsf{hashkg}^\dagger(\Lambda)$ outputs $\mathbf{hk}^\dagger \xleftarrow{\$} \mathbb{Z}_q^{4\times\nu} =: \mathcal{K}$;
- $\mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)$ outputs $\mathfrak{hp}^\dagger \leftarrow \left(\begin{smallmatrix} \mathfrak{g} & \mathbf{0} \\ \mathbf{0} & \mathfrak{g} \end{smallmatrix}\right) \cdot \mathbf{hk}^\dagger \in \mathbb{G}^{2\times\nu} =: \mathcal{K}_{\mathsf{hp}}$;
- $\mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau)$ outputs $\mathfrak{H}^\dagger \leftarrow \mathbf{hk}^{\dagger\mathsf{T}} \cdot \left(\begin{smallmatrix} \mathfrak{b} \\ \tau\cdot\mathfrak{b} \end{smallmatrix}\right) \in \mathbb{G}^\nu =: \Pi$;
- $\mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger, \mathfrak{b}, w, \tau)$ outputs $\mathfrak{pH}^\dagger \leftarrow \left(\begin{smallmatrix} w \\ \tau\cdot w \end{smallmatrix}\right)^\mathsf{T} \cdot \mathfrak{hp}^\dagger \in \mathbb{G}^\nu = \Pi$.

This can be use to construct IND-IPFE-CCA inner-product functional encryption schemes as showed by the following lemma.

Lemma 6.3.4. *Using above notation, let* $\mathsf{hashkg}'^\dagger = \mathsf{hashkg}^\dagger$, $\Sigma^\dagger = \mathbb{Z}_q$, $\varepsilon_{2\mathrm{u}}^\dagger = 1/q^\nu$, M_x *be a positive integer, and* $\varepsilon_{\mathrm{uti}}^\dagger = 0$. *Then, the PHF described above is a* $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon_{2\mathrm{u}}^\dagger, M_x, \varepsilon_{\mathrm{ti}})$*-IPFE-CCA-friendly.*

Proof. Key Homomorphism is straightforward.

Projection Key Homomorphism is straightforward.

Universal Translation Indistinguishability follows from Lemma 5.4.4.

2-Universality. As each 4-tuple of coordinates of $\mathbf{hk}^\dagger$ are generated independently,

$$\begin{aligned}
&\Pr\left[\ \mathfrak{H}'^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}', \tau') \wedge \mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)\ \right] \\
&= \prod_{i=1}^{\nu} \Pr[\mathfrak{H}'^\dagger = \overline{\mathsf{hash}^\dagger}(\mathbf{hk}_i^\dagger, \mathfrak{b}', \tau') \wedge \mathfrak{H}_i^\dagger = \overline{\mathsf{hash}^\dagger}(\mathbf{hk}_i^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}_i^\dagger = \overline{\mathsf{projkg}^\dagger}(\mathbf{hk}^\dagger)] \\
&\leq \prod_{i=1}^{\nu} \frac{1}{q} \Pr\left[\ \mathfrak{H}_i^\dagger = \overline{\mathsf{hash}^\dagger}(\mathbf{hk}_i^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}_i^\dagger = \overline{\mathsf{projkg}^\dagger}(\mathbf{hk}^\dagger)\ \right] \\
&= \frac{1}{q^\nu} \prod_{i=1}^{\nu} \Pr\left[\ \mathfrak{H}_i^\dagger = \overline{\mathsf{hash}^\dagger}(\mathbf{hk}_i^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}_i^\dagger = \overline{\mathsf{projkg}^\dagger}(\mathbf{hk}^\dagger)\ \right] \\
&= \Pr\left[\ \mathfrak{H}^\dagger = \mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau) \wedge \mathfrak{hp}^\dagger = \mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)\ \right]\ ,
\end{aligned}$$

where $(\overline{\mathsf{hashkg}^\dagger}, \overline{\mathsf{projkg}^\dagger}, \overline{\mathsf{hash}^\dagger}, \overline{\mathsf{projhash}^\dagger})$ is the $1/q$-2-universal PHF from Lemma 6.3.3.

The theorem follows from Lemma 5.4.6. □

The resulting IND-TBIPFE-CCA secure scheme will be explicited in Section 6.4.

MDDH Based IPFE-CCA-Friendly PHF. The previous construction can be extended in a straightforward way to any MDDH-based subset membership problem in a straightforward way, similar to what is done for our IPFE-CPA-friendly construction in the previous subsection.

6.4 Inner-Product Functional Encryption Schemes Based on the Decisional Diffie-Hellman Assumption

In this section, we present 3 inner-product functional encryption schemes based on the decisional Diffie-Hellman assumption. The first one is the most simple, but it only reaches s-IND-IPFE-CPA security. The second one is almost as efficient, adding 1 coordinate to the vectors only, and reaches adaptive IND-IPFE-CPA security. The last one reaches IND-IPFE-CCA security, but is more expensive: the public keys, secret keys, and ciphertexts are bigger than in the two previous schemes. We illustrate the differences between the schemes in Table 6.1 and present the tradeoffs between security and efficiency. We see that the adaptive security is almost for free, thus the second scheme should be prefered to the first one, which is still a good textbook example of inner-product functional encryption. On the other hand, security against active adversary is very expensive, and it should be considered with care if it is required or not.

6.4.1 Scheme Secure Against Selective Chosen-Plaintext Attacks

This scheme can be obtained by plugging the ElGamal public key encryption scheme in Construction 4.3.1. It is depicted on Figure 6.1 We recall that due to the constraint on the decryption, we have to set $\mathcal{M}_x$ and $\mathcal{M}_y$ such that the result of the inner-product is not too big in order to ensure correctness. Remember that we also need $\mathcal{M}_x$ to be small enough for the proof of security.

By using Theorem 4.3.3, we immediately get the following security theorem for our construction.

Security	s-IND-IPFE-CPA	IND-IPFE-CPA	IND-TBIPFE-CCA
mpk	ℓ	ℓ	$2\ell^2 + 3\ell$
msk	ℓ	2ℓ	$4\ell^2 + 6\ell$
$sk_{\vec{y}}$	1	2	$4\ell + 6$
$ct_{\vec{x}}$	$\ell + 1$	$\ell + 2$	$\ell^2 + 2\ell + 2$

Table 6.1: Comparison of the three different DDH-based inner-product functional encryption schemes. The sizes are represented as a number of group elements for the master public keys and ciphertexts, while it is represented as the number of scalars in $\mathbb{Z}_q$ for master and user secret keys. We recall that in order to achieve IND-IPFE-CCA security from the IND-TBIPFE-CCA secure scheme, we just need to add a one-time signature to the ciphertext, and does not make the size of the keys grow.

- Let $\mathbb{G}$ be a cyclic group of prime order q, $\mathfrak{g}_\perp$ a generator of $\mathbb{G}$.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Set $pp = (\kappa, \ell, \mathbb{G}, q, \mathfrak{g}_\perp)$. For $i = 1, \ldots, \ell$, set

$$s_i \xleftarrow{\$} \mathbb{Z}_q\,, \qquad\qquad h_i \leftarrow s_i \cdot \mathfrak{g}_\perp \in \mathbb{G}\,.$$

  Return $msk = \vec{s} \in \mathbb{Z}_q^\ell$ and $mpk = \vec{h} \in \mathbb{G}^\ell$;
- $\mathsf{KeyDer}(msk, \vec{y} \in \mathcal{M}_y) \mapsto sk_{\vec{y}}$: Return

$$sk_{\vec{y}} \leftarrow \langle \vec{y}, \vec{s} \rangle \in \mathbb{Z}_q\,;$$

- $\mathsf{Encrypt}(mpk, \vec{x} \in \mathcal{M}_x) \mapsto ct_{\vec{x}}$: Pick $r \xleftarrow{\$} \mathbb{Z}_q$ and set

$$ct_0 \leftarrow r \cdot \mathfrak{g}_\perp \in \mathbb{G}.$$

  For $i = 1, \ldots, \ell$, set

$$ct_i \leftarrow x_i \cdot \mathfrak{g}_\perp + r \cdot \mathfrak{hp}_i \in \mathbb{G}\,.$$

  Return $ct_{\vec{x}} = (ct_0, \vec{ct} \in \mathbb{G}^\ell)$;
- $\mathsf{Decrypt}(sk_{\vec{y}}, ct_{\vec{x}})$: Set

$$ct_{\langle \vec{x}, \vec{y} \rangle} \leftarrow \langle \vec{y}, \vec{ct} \rangle - \mathbf{hk}_{\vec{y}}^{\mathsf{T}} \cdot \mathfrak{b} \in \mathbb{G}.$$

  Return $\log_{\mathfrak{g}_\perp} ct_{\langle \vec{x}, \vec{y} \rangle}$.

Figure 6.1: DDH-based inner-product functional encryption secure against selective chosen-plaintext attacks

Theorem 6.4.1. *Under the DDH assumption in $\mathbb{G}$, the scheme $\mathcal{FE}$ depicted on Figure 6.1 is* s-IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the s-IND-IPFE-CPA *security of $\mathcal{FE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the DDH assumption, such that $\varepsilon_{\mathscr{A}} \leq \varepsilon_{\mathscr{B}}$.*

6.4.2 Scheme Secure Against Adaptive Chosen-Plaintext Attacks

The following scheme can be obtained in two different ways: plugging the ElGamal public key encryption scheme into Construction 4.4.1, or plugging our key homomorphic projective hash function based on DDH into Construction 5.3.1. It is depicted on Figure 6.2. To satisfy Condition 1, we need to choose the efficiently recognizable subsets $\mathcal{M}_y$ and $\mathcal{M}_x$ of $\mathcal{R}^\ell$ so that the discrete logarithm of $\langle \vec{y}, \vec{x} \rangle \cdot \mathfrak{g}_\perp \in \mathbb{G}$ is efficient to compute, for any $\vec{y} \in \mathcal{M}_y$ and $\vec{x} \in \mathcal{M}_x$. This scheme can easily be extended to rely on any MDDH assumption by using instead projective hash functions defined in Section 6.3.1.2, however we stick to the instantiation based on DDH for simplicity.

- Let $\mathbb{G}$ be a cyclic group of prime order q, $\mathfrak{g}_\perp$ a generator of $\mathbb{G}$.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Choose $\mathfrak{g} \xleftarrow{\$} \mathbb{G}^2$. Set $pp = (\kappa, \ell, \mathfrak{g})$. For $i = 1, \ldots, \ell$, set

$$\mathbf{hk}_i \xleftarrow{\$} \mathbb{Z}_q^2 , \qquad \mathfrak{hp}_i \leftarrow \mathbf{hk}_i^\intercal \cdot \mathfrak{g} \in \mathbb{G} .$$

  Return $msk = \overrightarrow{\mathbf{hk}} \in (\mathbb{Z}_q^2)^\ell$ and $mpk = \overrightarrow{\mathfrak{hp}} \in \mathbb{G}^\ell$.
- $\mathsf{KeyDer}(msk, \vec{y} \in \mathbb{Z}_q^\ell) \mapsto sk_{\vec{y}}$: Set

$$\mathbf{hk}_{\vec{y}} \leftarrow \langle \vec{y}, \overrightarrow{\mathbf{hk}} \rangle \in \mathbb{Z}_q^2 .$$

  Return $sk_{\vec{y}} = \mathbf{hk}_{\vec{y}} \in \mathbb{Z}_q^2$.
- $\mathsf{Encrypt}(mpk, \vec{x} \in \mathbb{Z}_q^\ell) \mapsto ct_{\vec{x}}$: Pick $r \xleftarrow{\$} \mathbb{Z}_q$ and set $\mathfrak{b} \leftarrow r \cdot \mathfrak{g} \in \mathbb{G}^2$.
  For $i = 1, \ldots, \ell$, set

$$ct_i \leftarrow x_i \cdot \mathfrak{g}_\perp + r \cdot \mathfrak{hp}_i \in \mathbb{G} .$$

  Return $ct_{\vec{x}} = (\mathfrak{b}, \overrightarrow{ct} \in \mathbb{G}^\ell)$.
- $\mathsf{Decrypt}(sk_{\vec{y}}, ct_{\vec{x}})$: Set

$$ct_{\langle \vec{x}, \vec{y} \rangle} \leftarrow \langle \vec{y}, \overrightarrow{ct} \rangle - \mathbf{hk}_{\vec{y}}^\intercal \cdot \mathfrak{b} \in \mathbb{G}.$$

  Return $\log_{\mathfrak{g}_\perp} ct_{\langle \vec{x}, \vec{y} \rangle}$.

Figure 6.2: DDH-based inner-product functional encryption secure against chosen-plaintext attacks

This scheme corresponds to the DDH-based schemes of [ALS16] and [ABDP16]. Applying Theorem 5.3.2 or Theorem 4.4.2, we immediately get the following security theorem.

Theorem 6.4.2. *Under the DDH assumption in $\mathbb{G}$, the scheme $\mathcal{FE}$ depicted on Figure 6.2 is* IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the IND-IPFE-CPA *security of $\mathcal{FE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the DDH assumption, such that $\varepsilon_{\mathscr{A}} \leq 2 \cdot \varepsilon_{\mathscr{B}}$.*

It is worth noting that the term $\ell \cdot |\Delta\mathcal{M}_x| \cdot \varepsilon_{\text{ti}}$ from Theorem 5.3.2 has disappeared because of the key-uniformity.

6.4.3 Scheme Secure Against Chosen-Ciphertext Attacks

Let us now instantiate the framework for IND-TBIPFE-CCA secure tag-based inner-product functional encryption schemes with the DDH-based IPFE-CPA-friendly PHF defined in Section 6.3.1.1, and the DDH-based IPFE-CCA-friendly tag-based PHF defined in Section 6.3.2. We set $\mathcal{R} = \mathbb{Z}_q$ and $M_x = q$ (or any large enough integer). As for the IND-IPFE-CPA secure scheme in the previous section, we need to choose the efficiently recognizable subsets $\mathcal{M}_y$ and $\mathcal{M}_x$ of $\mathcal{R}^\ell$ so that the discrete logarithm of $\langle \vec{y}, \vec{x} \rangle \cdot \mathfrak{g}_\perp \in \mathbb{G}$ is efficient to compute, for any $\vec{y} \in \mathcal{M}_y$ and $\vec{x} \in \mathcal{M}_x$ in order to satisfy Condition 2. The resulting construction is depicted on Figure 6.3 and can be easily extended to use any MDDH-based PHF defined in Section 6.3.2.

Applying Theorem 5.5.2, we immediately get the following security theorem.

Theorem 6.4.3. *Under the DDH assumption in $\mathbb{G}$, the scheme $\mathcal{TBFE}$ depicted on Figure 6.2 is* IND-TBIPFE-CCA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the IND-TBIPFE-CCA *security of $\mathcal{TBFE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the DDH assumption, such that $\varepsilon_{\mathscr{A}} \leq 2 \cdot \varepsilon_{\mathscr{B}} + 2 \cdot q_{dec} \cdot q^{\ell-\nu}$.*

In particular, setting $\nu = \ell + 1$, we have the following bound: $\varepsilon_{\mathscr{A}} \leq 2 \cdot \varepsilon_{\mathscr{B}} + 2 \cdot \frac{q_{\text{dec}}}{q}$. We consider this setting of parameter for the comparison of the schemes.

- Let $\mathbb{G}$ be a cyclic group of prime order q, $\mathfrak{g}_\perp$ a generator of $\mathbb{G}$.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Choose $\mathfrak{g} \xleftarrow{\$} \mathbb{G}^2$. Set $pp = (\kappa, \ell, \mathfrak{g})$. For $i = 1, \ldots, \ell$, set

$$\mathbf{hk}_i \xleftarrow{\$} \mathbb{Z}_q^2\,, \qquad \mathfrak{hp}_i \leftarrow \mathbf{hk}_i^\intercal \cdot \mathfrak{g} \in \mathbb{G}\,.$$
$$\mathbf{hk}_i^\dagger \xleftarrow{\$} \mathbb{Z}_q^{4\times\nu}\,, \qquad \mathfrak{hp}_i^\dagger \leftarrow \begin{pmatrix} \mathfrak{g} & \mathbf{0} \\ \mathbf{0} & \mathfrak{g} \end{pmatrix}^\intercal \cdot \mathbf{hk}_i^\dagger \in \mathbb{G}^{2\times\nu}\,.$$

  Return $msk = (\overrightarrow{\mathbf{hk}} \in (\mathbb{Z}_q^2)^\ell, \overrightarrow{\mathbf{hk}}^\dagger \in (\mathbb{Z}_q^{4\times\nu})^\ell)$ and $mpk = (\overrightarrow{\mathfrak{hp}} \in \mathbb{G}^\ell, \overrightarrow{\mathfrak{hp}}^\dagger \in (\mathbb{G}^{2\times\nu})^\ell)$.
- $\mathsf{KeyDer}(msk, \overrightarrow{y} \in \mathbb{Z}_q^\ell) \mapsto sk_{\overrightarrow{y}}$: Set

$$\mathbf{hk}_{\overrightarrow{y}} \leftarrow \langle \overrightarrow{y}, \overrightarrow{\mathbf{hk}} \rangle \in \mathbb{Z}_q^2\,, \qquad \mathbf{hk}_{\overrightarrow{y}}^\dagger \leftarrow \langle \overrightarrow{y}, \overrightarrow{\mathbf{hk}}^\dagger \rangle \in \mathbb{Z}_q^{4\times\nu}\,.$$

  Return $sk_{\overrightarrow{y}} = (\mathbf{hk}_{\overrightarrow{y}} \in \mathbb{Z}_q^2, \mathsf{hk}_{\overrightarrow{y}}^\dagger \in \mathbb{Z}_q^{4\times\nu})$.
- $\mathsf{Encrypt}(\tau, mpk, \overrightarrow{x} \in \mathbb{Z}_q^\ell) \mapsto ct_{\overrightarrow{x}}$: Pick $r \xleftarrow{\$} \mathbb{Z}_q$ and set $\mathfrak{b} \leftarrow r \cdot \mathfrak{g} \in \mathbb{G}^2$.
  For $i = 1, \ldots, \ell$, set

$$ct_i \leftarrow x_i \cdot \mathfrak{g}_\perp + r \cdot \mathfrak{hp}_i \in \mathbb{G}\,, \qquad \boldsymbol{ct}_i^\dagger \leftarrow \mathfrak{hp}_i^\dagger \cdot \begin{pmatrix} r \\ \tau \cdot r \end{pmatrix} \in \mathbb{G}^\nu\,.$$

  Return $ct_{\overrightarrow{x}} = (\mathfrak{b}, \overrightarrow{ct} \in \mathbb{G}^\ell, \overrightarrow{\boldsymbol{ct}}^\dagger \in (\mathbb{G}^\nu)^\ell)$.
- $\mathsf{Decrypt}(\tau, sk_{\overrightarrow{y}}, ct_{\overrightarrow{x}})$: Check that $\langle \overrightarrow{y}, \overrightarrow{\boldsymbol{ct}}^\dagger \rangle = \mathbf{hk}_{\overrightarrow{y}}^{\dagger\,\intercal} \cdot \begin{pmatrix} \mathfrak{b} \\ \tau \cdot \mathfrak{b} \end{pmatrix}$; return $\perp$ if it fails.
  Set

$$ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle} \leftarrow \langle \overrightarrow{y}, \overrightarrow{ct} \rangle - \mathbf{hk}_{\overrightarrow{y}}^\intercal \cdot \mathfrak{b} \in \mathbb{G}.$$

  Return $\log_{\mathfrak{g}_\perp} ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle}$.

Figure 6.3: DDH-based tag-based inner-product functional encryption secure against chosen-ciphertext attacks

Chapter 7

Instantiations based on the Decisional Composite Residuosity Assumption

In this chapter, we instantiate the generic constructions of inner-product functional encryption from key-homomorphic projective hash functions with concrete constructions based on the decisional composite residuosity assumption.
We present two non-modular IPFE schemes reaching respectively IND-IPFE-CPA security (adaptive) and IND-TBIPFE-CCA security and compare their pros and cons in term of efficiency and security.
These schemes solve the problem from which the schemes of Chapter 6 suffer: the decryption is very efficient. However, the downside is that the scheme has bigger keys and ciphertexts, because it cannot rely on elliptic curve cryptography, and requires huge integers to guarantee security. The fact that some computations are done over the integers also makes it hard to analyze the efficiency of the scheme. We also note that the schemes presented in this chapter are non-modular, which is also another downside.
The constructions, and proofs that the projective hash functions verify the correct properties are taken from our contribution [BBL17].

Contents

7.1 The Decisional Composite Residuosity Assumption and Subset Membership Problem . . . 98
7.1.1 The Decisional Composite Residuosity Assumption . . . 98
7.1.2 Subset Membership Problem Based on the Decisional Composite Residuosity Assumption . . . 98
7.2 Projective Hash Functions Based on the Decisional Composite Residuosity Assumptions . . . 100
7.2.1 IPFE-CPA-Friendly Projective Hash Function . . . 100
7.2.2 IPFE-CCA-Friendly Projective Hash Function . . . 101
7.3 Inner-Product Functional Encryption Schemes Based on the Decisional Composite Residuosity Assumption . . . 103
7.3.1 Scheme Secure Against Chosen-Plaintext Attacks . . . 103
7.3.2 Scheme Secure Against Chosen-Ciphertext Attacks . . . 105

7.1 The Decisional Composite Residuosity Assumption and Subset Membership Problem

In this section, we recall the *decisional composite residuosity* (DCR) assumption and present a subset membership problem whose indistinguishability can be reduced to the DCR assumption, which will be used to construct key homomorphic projective hash functions and inner-product functional encryption schemes.

7.1.1 The Decisional Composite Residuosity Assumption

The DCR assumption was introduced to build public key encryption schemes [Pai99]. It is part of one of the two most important famillies of assumption for public key cryptography: assumptions based on the hardness of factorization, that started with the RSA cryposystem [RSA78]. Indeed, if it is easy to factor big numbers, then the DCR assumption does not hold. However, we do not know any better attack on this assumption than factoring the modulus and recover the two primes that compose it. As with all other assumptions based on factoring, we will work in the ring $\mathbb{Z}_N$, where N is an RSA number, i.e., a product of two big prime numbers. We need to be careful here because $\mathbb{Z}_N$ is not a field as was $\mathbb{Z}_q$ in Chapter 6. However, it is very hard to find a number that do not have an inverse, in fact, it is as hard as finding the two prime factors of N.

The Decisional Composite Residuosity Assumption. Let $N = pq$ be a product of two λ-bit random safe primes $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes and where λ is a function of the security parameter κ. The *decisional composite residuosity* assumption states that z and y^N are computationally indistinguishable, where z and y are chosen uniformly at random from $\mathbb{Z}_{N^2}$ and the exponentiation is done modulu N^2.

We won't directly use this assumption but an assumption which is equivalent for our subset membership problem.

7.1.2 Subset Membership Problem Based on the Decisional Composite Residuosity Assumption

In order to improve the efficiency of our scheme, we will work in the ring $\mathbb{Z}_{N^{s+1}}$ for some integer s instead of the ring $\mathbb{Z}_{N^2}$. This change was introduced by Damgård and Jurik [DJ01] to improve the asymptotic efficiency of the Pailler encryption scheme. Everything behaves the same way, but instead of having a message space of size N, we have message space N^s, while still having an overhead of $\log(N)$ bits for security reasons.

The second change we make in order to have a better inner-product functional encryption scheme at the end is using the group of signed quadratic residues [FS97; HK09; HKS13]. Basically, we change the group law in order to ensure that we are only dealing with quadratic numbers, and that it is efficient to check if a number is a square. This isn't the case when taking numbers in $\mathbb{Z}_N$, in fact it is even a cryptographic assumption: the quadratic residuosity (QR) assumption that is used for example to build public key encryption schemes [GM82; GM84].
Using only squares ensures that the smallest subgroup of our field has a big order, because we removed both subgroup of order 2. Hence, when an element is uniform in a subgroup

chosen by the adversary, it is still statistically impossible to guess its value beforehand.

Signed Quadratic Residues. Let $N = pq$ be a product of two λ-bit random safe primes $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes and where λ is a function of the security parameter κ and let $N' = p'q'$. The order of the group $\mathbb{Z}_N^*$ of invertible elements in $\mathbb{Z}_N$ is $\phi(N) = 4N'$ (here, ϕ is Euler's totient function). We denote by $\mathbb{J}_N$ the subgroup of $\mathbb{Z}_N^*$ order $2N'$ of elements from $\mathbb{Z}_N^*$ with Jacobi symbol 1. We also denote by $\mathbb{QR}_N = \{x : \exists y \in \mathbb{Z}_N, x = y^2[N]\}$ the subgroup of $\mathbb{J}_N$ of order N' of quadratic residues modulo N. As N is a Blum integer (since p and q are both safe primes), $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$, because it is not a square modulo p nor modulo q.
We now write the elements of $\mathbb{Z}_N$ as $\{-\frac{N-1}{2}, \ldots, \frac{N-1}{2}\}$, and we define the group of *signed quadratic residues* modulo N

$$\mathbb{QR}_N^+ = \{|x| : x \in \mathbb{QR}_N\},$$

where the group operation $\circ$ is defined through $|x| \circ |y| = |xy|$. As for any $x \in \mathbb{QR}_N$, $-x \in \mathbb{J}_N \setminus \mathbb{QR}_N$, $|\mathbb{QR}_N^+| = |\mathbb{QR}_N| = \frac{1}{2}|\mathbb{J}_N|$, so $\mathbb{QR}_N^+ = \mathbb{J}_N^+$, where $\mathbb{J}_N^+ = \{|x| : x \in \mathbb{J}_N\} = \mathbb{J}_N/\pm 1$. In particular, since the jacobi symbol of an element is efficiently computable, it is easy to verify if an element is in $\mathbb{QR}_N^+$ or not. We are now ready to define our subset membership problem.

DCR-Based Subset Membership Problem. Let $N = pq$ be a product of two λ-bit random safe primes $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes and where λ is a function of the security parameter κ, and let $N' = p'q'$. Let $s \geq 1$ be an integer parameter. Let us write $\mathbb{Z}_{N^{s+1}}^* \cong G_{N^s} \oplus G_{N'} \oplus G_2 \oplus T$, where $\cong$ denotes group isomorphism, $\oplus$ is the direct sum or Cartesian product, G_i are cyclic groups of order i, and T is the order-2 cyclic group generated by $-1 \mod N^{s+1}$. As previously, let us represent the elements of $\mathbb{Z}_{N^{s+1}}$ as $\{-(N^{s+1} - 1)/2, \ldots, (N^{s+1} - 1)/2\}$. We define $\mathbb{G} = \mathcal{X} = \{|x| : \exists y \in \mathbb{Z}_{N^{s+1}}^* \text{ s.t. } x = y^2\} = \mathbb{J}_{N^{s+1}} \cap \{1, \ldots, (N^{s+1} - 1)/2\} \cong G_{N^s} \oplus G_{N'}$. We will use additive notation for $\mathbb{G}$ in order to have notation closer to the ones used for the generic constructions: $\mathfrak{g}_1 + \mathfrak{g}_2 = |\mathfrak{g}_1\mathfrak{g}_2|$. Let $\mathfrak{g}$ be a random generator of $\mathcal{L} \cong G_{N'}$, that is a subgroup of $\mathcal{X}$; $\mathfrak{g}$ can be thought of as a random $2N^s$-th residue. A witness $w \in \mathcal{W} = \mathbb{Z}$ for $\mathfrak{b} \in \mathcal{L}$ is such that $\mathfrak{b} = w \cdot \mathfrak{g}$. Finally, let $\mathfrak{g}_\perp$ be an arbitrary generator of the cyclic group G_{N^s} (for example $\mathfrak{g}_\perp = 1 + N \in \mathbb{Z}_{N^{s+1}}$, where $+$ here is the additive law of $\mathbb{Z}_{N^{s+1}}$) and let $\bar{\mathcal{L}} = \mathcal{L} + \mathfrak{g}_\perp$. We set $\Lambda = (N, s, \mathfrak{g}, \mathfrak{g}_\perp)$.

One cannot sample uniform witnesses as $\mathcal{W} = \mathbb{Z}$ is infinite. We cannot set $\mathcal{W} = \mathbb{Z}_{N'}$, as computing N' from $\Lambda = (N, s, \mathfrak{g})$ requires to factor N. Instead, we sample witnesses uniformly from $S_N := \{0, \ldots, \lfloor N/4 \rfloor - 1\}$. Clearly, the statistical distance $\Delta(U(\mathbb{Z}_{N'}), U(S_N)) = 1 - p'q'/(pq/4) = (2p' + 2q' + 1)/(pq) < 2(p + q)/(pq) < 4/\operatorname{spf}(N)$. From this distribution over $\mathcal{W}$, we can derive distributions over ϱ, $\mathcal{L}$, and $\bar{\mathcal{L}} = \mathcal{L} + \mathfrak{g}_\perp$. The two latter distributions are statistically close to uniform.

This setting defines a subset membership problem, whose $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability property can be proven under the decisional composite residuosity assumption. More precisely, we consider the DCR assumption for moduli that are product of safe primes; the DCR assumption then basically states that in the case $s = 1$, no probabilistic polynomial time adversary can distinguish between uniform elements of $\mathcal{L}$ and $\mathcal{X}$. We note that the original assumption actually does not restrict the the elements to be of Jacobi symbol 1 nor square, but doing this restriction yields an equivalent assumption, since we can multiply element

of Jacobi symbol -1 by an arbitrary N^s-residue of Jacobi symbol -1, and as we are using the group of signed quadratic residues, we can go from a non square element to a square by multiplying it by -1. This is a classical variant of DCR, which is equivalent to the original DCR assumption [Pai99], assuming that safe primes are sufficiently dense (see, e.g., [CS02]). We have the following lemma, following [DJ01]:

Lemma 7.1.1. *If the DCR assumption holds, the above subset membership problem is $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishable. More precisely, if there exists an adversary $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking $(\mathcal{L}, \bar{\mathcal{L}})$-indistinguishability, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking DCR, such that $\varepsilon_{\mathscr{A}} \leq 2s \cdot \varepsilon_{\mathscr{B}} + 8/\operatorname{spf}(N)$.*

Proof. We use an hybrid proof as in [DJ01] using s hybrids. The term $8/\operatorname{spf}(N)$ comes from the statistical distance between our distribution over $\mathcal{X}$ and $\mathcal{L}$ and the uniform distribution. □

7.2 Projective Hash Functions Based on the Decisional Composite Residuosity Assumptions

We are now ready to present the projective hash function that we will use to construct inner-product functional encryption scheme. We start with the IPFE-CPA-friendly PHF, and then present the IPFE-CCA-friendly one.

7.2.1 IPFE-CPA-Friendly Projective Hash Function

We now define our IPFE-CPA-friendly projective hash function based on the DCR subset membership problem. It is a generalization of the PHF of Cramer and Shoup based on DCR in [CS02] Let $\Lambda = (N, s, \mathfrak{g}, \mathfrak{g}_\perp)$ be defined as in Section 7.1.2. We have: $\mathbb{G} = \mathcal{X} = \mathbb{J}_{N^{s+1}} \cap \{1, \ldots, (N^{s+1}-1)/2\} \cong G_{N^s} \oplus G_{N'}$, $\mathcal{L} = G_{N'}$, and $\bar{\mathcal{L}} = \mathcal{L} + \mathfrak{g}_\perp$. The element $\mathfrak{g}$ is a generator of $\mathcal{L}$, while $\mathfrak{g}_\perp$ is a generator of G_{N^s}. We recall that we use additive notation for the group $\mathbb{G}$: $\mathfrak{g}_1 + \mathfrak{g}_2 = |\mathfrak{g}_1\mathfrak{g}_2|$, where elements of $\mathbb{Z}_{N^{s+1}}$ are represented as $\{-(N^{s+1}-1)/2, \ldots, (N^{s+1}-1)/2\}$.

We define the DCR-based PHF as follows:

- $\mathsf{hashkg}(\Lambda)$ outputs $\mathsf{hk} \xleftarrow{\$} \{0, \ldots, \lfloor MN^{s+1}/4 \rfloor\} =: \mathcal{K}^* \subseteq \mathbb{Z} =: \mathcal{K}$, where M is a positive integer and is a parameter of the scheme;
- $\mathsf{projkg}(\mathsf{hk})$ outputs $\mathfrak{hp} \leftarrow \mathsf{hk} \cdot \mathfrak{g} \in \mathbb{G}$;
- $\mathsf{hash}(\mathsf{hk}, \mathfrak{b})$ outputs $\mathfrak{H} \leftarrow \mathsf{hk} \cdot \mathfrak{b} \in \mathbb{G} =: \Pi$;
- $\mathsf{projhash}(\mathfrak{hp}, \mathfrak{b}, w)$ outputs $\mathfrak{pH} \leftarrow \mathfrak{hp} \cdot w \in \mathbb{G} = \Pi$.

When $M = 2$, this PHF corresponds to the one of Cramer and Shoup in [CS02].

We insist on the fact that the set of hashing keys is $\mathcal{K} = \mathbb{Z}$ so that it is a group. However, hashkg only samples a hashing key from a finite subset $\mathcal{K}^*$ of $\mathcal{K}$.

Lemma 7.2.1. *Using above notation, let $M_\perp = N^s$, M_x be a positive integer, and $\varepsilon_{\mathrm{ti}} = M_x/M$. Let $\mathsf{hk}_\perp$ be defined as follows:*

$$\mathsf{hk}_\perp(\mathfrak{b}) = N' \cdot (N'^{-1} \bmod N^s) \quad (< N'N^s < N^{s+1}/4) \ .$$

Then, the PHF described above is $(\mathsf{hk}_\perp, \mathfrak{g}_\perp, M_\perp, M_x, \varepsilon_{\mathrm{ti}})$-IPFE-CPA-friendly.

Proof. Key homomorphism and strong diversity are proven similarly as in the DDH case, while translation indistinguishability follows from Lemma 2.1.2.

Key Homomorphism is straightforward.

Strong Diversity. Assume $\mathfrak{b} \in \bar{\mathcal{L}} = \mathcal{L} + \mathfrak{g}_\perp$, write $\mathfrak{b} = w\mathfrak{g} + \mathfrak{g}_\perp$ for $w \in \mathbb{Z}_{N'}$. Since the space of projection keys is also a group and projkg is a group homomorphism, we can use Lemma 5.2.4. Hence, we just need to prove that $\mathsf{projkg}(\mathsf{hk}_\perp(\mathfrak{b})) = 0$ and $\mathsf{hash}(\mathsf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) = \mathfrak{g}_\perp$. The first equality follows from the facts that

$$\mathsf{projkg}(\mathsf{hk}_\perp(\mathfrak{b})) = \mathsf{hk}_\perp(\mathfrak{b}) \cdot \mathfrak{g} = N' \cdot (N'^{-1} \bmod N^s) \cdot \mathfrak{g}$$

and that $\mathfrak{g}$ has order N'. For the second equality, we remark that

$$\mathsf{hash}(\mathsf{hk}_\perp(\mathfrak{b}), \mathfrak{b}) = \mathsf{hk}_\perp(\mathfrak{b}) \cdot \mathfrak{b} = N' \cdot (N'^{-1} \bmod N^s) \cdot w \cdot \mathfrak{g} + N' \cdot (N'^{-1} \bmod N^s) \cdot \mathfrak{g}_\perp \ .$$

Since $\mathfrak{g}$ has order N', as before $N' \cdot (N'^{-1} \bmod N^s) \cdot w \cdot \mathfrak{g} = 0$. Furthermore, since $\mathfrak{g}_\perp$ has order N^s,

$$N' \cdot (N'^{-1} \bmod N^s) \cdot \mathfrak{g}_\perp = (N' \cdot N'^{-1} \bmod N^s) \cdot \mathfrak{g}_\perp = \mathfrak{g}_\perp \ .$$

Translation Indistinguishability. For any $x \in \{-M_x, \ldots, M_x\}$, we have

$$\begin{aligned}\Delta(\mathsf{hashkg}(\Lambda), \mathsf{hashkg}(\Lambda) + x \cdot \mathsf{hk}_\perp(\mathfrak{b})) &= \frac{|x \cdot \mathsf{hk}_\perp(\mathfrak{b})|}{|\mathcal{K}^*|} \\ &\leq \frac{M_x \cdot N'N^s}{MN^{s+1}/4} < \frac{M_x N/4}{MN/4} = \frac{M_x}{M} = \varepsilon_{\mathrm{ti}} \ .\end{aligned}$$

This concludes the proof. □

Interestingly, because of our choice of $\bar{\mathcal{L}}$, $\mathsf{hk}_\perp(\mathfrak{b})$ does not depend on $\mathfrak{b}$. Note also that for $M < M_x/\varepsilon_{\mathrm{ti}}$, this PHF is still key-homomorphic and strongly diverse, but might lack the translation indistinguishability property that is necessary for our application.

7.2.2 IPFE-CCA-Friendly Projective Hash Function

We now define our IPFE-CCA-friendly projective hash function based on the DCR subset membership problem. As for the DDH-based IPFE-CCA-friendly projective hash function of Section 6.3.2, we use a PHF that is basically ν copies in parallel of a 2-universal PHF based on DCR.

Once again, let $\Lambda = (N, s, \mathfrak{g}, \mathfrak{g}_\perp)$ be defined as in Section 7.1.2. We have: $\mathbb{G} = \mathcal{X} = J_{N^{s+1}} \cap \{1, \ldots, (N^{s+1}-1)/2\} \cong G_{N^s} \oplus G_{N'}$, $\mathcal{L} = G_{N'}$, and $\bar{\mathcal{L}} = \mathcal{L} + \mathfrak{g}_\perp$. The element $\mathfrak{g}$ is a generator of $\mathcal{L}$, while $\mathfrak{g}_\perp$ is a generator of G_{N^s}. We recall that we use additive notation for the group $\mathbb{G}$: $\mathfrak{g}_1 + \mathfrak{g}_2 = |\mathfrak{g}_1\mathfrak{g}_2|$, where elements of $\mathbb{Z}_{N^{s+1}}$ are represented as $\{-(N^{s+1}-1)/2, \ldots, (N^{s+1}-1)/2\}$.

We define a PHF as follows:

- The tag set is $\mathcal{T} = \{0, \ldots, \lfloor N/2 \rfloor\} \subseteq \mathbb{Z}_{N'}$
- $\mathsf{hashkg}^\dagger(\Lambda)$ outputs $\mathbf{hk}^\dagger \xleftarrow{\$} \{0, \ldots, \lfloor \nu M^\dagger N^{s+1}/2 \rfloor\}^{2\times\nu} =: \mathcal{K}^* \subseteq \mathbb{Z}^{2\times\nu} =: \mathcal{K}$, where $M^\dagger$ is a positive integer and is a parameter of the scheme;
- $\mathsf{projkg}^\dagger(\mathsf{hk}^\dagger)$ outputs $\mathfrak{hp}^\dagger \leftarrow \left(\begin{smallmatrix} \mathfrak{g} & 0 \\ 0 & \mathfrak{g} \end{smallmatrix}\right)^\mathsf{T} \cdot \mathbf{hk}^\dagger \in \mathbb{G}^{2\times\nu} =: \mathcal{K}_{\mathsf{hp}}$;
- $\mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau)$ outputs $\mathfrak{H}^\dagger \leftarrow \mathbf{hk}^{\dagger\mathsf{T}} \cdot \left(\begin{smallmatrix} \mathfrak{b} \\ \tau\cdot\mathfrak{b} \end{smallmatrix}\right) \in \mathbb{G}^\nu =: \Pi$;
- $\mathsf{projhash}^\dagger(\mathfrak{hp}^\dagger, \mathfrak{b}, w, \tau)$ outputs $\mathfrak{p}\mathfrak{H}^\dagger \leftarrow \mathfrak{hp}^{\dagger\mathsf{T}} \cdot \left(\begin{smallmatrix} w \\ \tau\cdot w \end{smallmatrix}\right) \in \mathbb{G}^\nu = \Pi$.

We have the following property.

Lemma 7.2.2. *Using above notation,* $\Sigma^\dagger = \{-N^s+1, \ldots, N^s-1\} \setminus \{0\}$, $\varepsilon^\dagger_{\mathrm{2u}} = 1/\operatorname{spf}(N)^\nu$, M_x *be a positive integer, and* $\varepsilon^\dagger_{\mathrm{uti}} = M_x/M^\dagger$. *Define in addition the following algorithm:*

- $\mathsf{hashkg}'^\dagger(\Lambda)$ *output* $\mathbf{hk}^\dagger \xleftarrow{\$} \mathbb{Z}^{2\times\nu}_{N'N^s} = \mathcal{K}^{*\dagger}$.

Then, the PHF described above is a $(\mathsf{hashkg}'^\dagger, \Sigma^\dagger, \varepsilon^\dagger_{\mathrm{2u}}, M_x, \varepsilon^\dagger_{\mathrm{uti}})$*-IPFE-CCA-friendly.*

Proof. Key Homomorphism is straightforward.

Projection Key Homomorphism is straightforward.

Universal Translation Indistinguishability. We recall the following classical lemma.

Lemma 7.2.3. *Let* $(A_1, \ldots, A_k)$ *and* $(B_1, \ldots, B_k)$ *be two tuples of* k *independent distributions. Let* A *(resp.* B*) be the product distribution of* $(A_1, \ldots, A_k)$ *(resp.* $(B_1, \ldots, B_k)$*), i.e., the distributions of tuples* $(x_1, \ldots, x_k)$ *where* x_i *is distributed according to* A_i *(resp.* B_i*), for* $i = 1, \ldots, k$. *Then:*

$$\Delta(A, B) \le \sum_{i=1}^{k} \Delta(A_i, B_i) \ .$$

Using this fact, for any $x \in \{-M_x, \ldots, M_x\}$ and any $\mathbf{hk}^\dagger \xleftarrow{\$} \mathsf{hashkg}'^\dagger(\Lambda)$, we have

$$\begin{aligned}
&\Delta(\mathsf{hashkg}(\Lambda), \mathsf{hashkg}(\Lambda) + x \cdot \mathsf{hk}) \\
&\le \sum_{i=1}^{2}\sum_{j=1}^{\nu} \Delta(\{0, \ldots, \lfloor \nu M^\dagger N^{s+1}/2 \rfloor\}, x \cdot \mathsf{hk}_{i,j} + \{0, \ldots, \lfloor \nu M^\dagger N^{s+1}/2 \rfloor\}) \\
&= \sum_{i=1}^{2}\sum_{j=1}^{\nu} \frac{|x \cdot \mathsf{hk}_{i,j}|}{|\mathcal{K}^*|} \le 2\nu \cdot \frac{M_x \cdot N'N^s}{\nu M^\dagger N^{s+1}/2} < \frac{M_x}{M^\dagger} = \varepsilon^\dagger_{\mathrm{uti}} \ .
\end{aligned}$$

Therefore:

$$\Delta(\mathsf{hashkg}(\Lambda), \mathsf{hashkg}(\Lambda) + x \cdot \mathsf{hk}) < \varepsilon^\dagger_{\mathrm{uti}} \ .$$

2-Universality. As for the case of DDH, we first study the 2-universality of the simple scheme which will be repeated ν times. Then we will conclude using the same argument, that is the parallel repetitions are independent. For the simple scheme, notice that conditioning on $\mathsf{projkg}^\dagger(\mathbf{hk}^\dagger)$ fixes the values of $\mathsf{hk}^\dagger_1$ and $\mathsf{hk}^\dagger_2$ modulo N', the order of $\mathfrak{g}$. So $\mathsf{hk}^\dagger_1$ and $\mathsf{hk}^\dagger_2$ are uniform in a coset of the subgroup of order N^s. Conditioning on $\mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}, \tau)$ fixes a relation between $\mathsf{hk}^\dagger_1$ and $\mathsf{hk}^\dagger_2$ modulo N^s depending on τ. However, as $\tau' \neq \tau$, this relation does not fix the value of $\mathsf{hash}^\dagger(\mathbf{hk}^\dagger, \mathfrak{b}', \tau')$, and the most likely value still has probability at most $\operatorname{spf}(N)$. □

We are now ready to present our inner-product functional encryption schemes.

7.3 Inner-Product Functional Encryption Schemes Based on the Decisional Composite Residuosity Assumption

In this section, we present the two schemes secure under the DCR assumption obtained by plugging the PHFs described in the previous section in Construction 5.3.1 and Construction 5.5.1. The first one reaches IND-IPFE-CPA security, the second one is an IND-TBIPFE-CCA secure tag-based IPFE scheme. The most interesting part about these construction is that they fix the biggest problem of the schemes based on the DDH assumption: the decryption is efficient. On the other hand, the schemes are non-modular, which is not as general as we would like. We recall however that [ALS16] presented a way to convert non-modular IPFE schemes into modular IPFE schemes, the only cost being that the key generation algorithm have to be stateful and remember up to ℓ different queries, which is still a reasonably low cost. We provide a comparison of those two schemes in Table 7.1, where we present the trade-offs between security and efficiency.

Security	IND-IPFE-CPA
mpk	$\ell(s\log(N)-1)$
msk	$\ell(\log(M)+s\log(N)-2)$
$sk_{\vec{y}}$	$\log(\ell)\log(M_y)(\log(M)+s\log(N)-2)$
$ct_{\vec{x}}$	$(\ell+1)(s\log(N)-1)$

Security	IND-TBIPFE-CCA
mpk	$\ell(2\nu+1)(s\log(N)-1)$
msk	$\ell\log(M)+(\nu+1)\ell(s\log(N)-1)+2\ell\nu(\log(M^\dagger)+\log(\nu))$
$sk_{\vec{y}}$	$\log(\ell)\log(M_y)(\log(M)+\log(M^\dagger)+\log(\nu)+2s\log(N)-3)$
$ct_{\vec{x}}$	$(\nu\ell+\ell+1)(s\log(N)-1)$

Table 7.1: Comparison of the two different DCR-based inner-product functional encryption schemes. The sizes are represented as a number of bits, since some elements are in $\mathbb{Z}$. We recall that in order to achieve IND-IPFE-CCA security from the IND-TBIPFE-CCA secure scheme, we just need to add a one-time signature to the ciphertext, and does not make the size of the keys grow.

7.3.1 Scheme Secure Against Chosen-Plaintext Attacks

Before presenting the scheme, let us first discuss the choice of parameters. We set $\mathcal{R}=\mathbb{Z}$. Contrary the DDH-based instantiation, the discrete logarithm problem in the subgroup generated by $\mathfrak{g}_\perp$ is easy: given $t\cdot\mathfrak{g}_\perp$, we can always efficiently recover t. However, to satisfy Condition 1, we need to choose $\mathcal{M}_y$ and $\mathcal{M}_x$ so that for any $\vec{y}\in\mathcal{M}_y$ and $\vec{x}\in\mathcal{M}_x$, $\langle\vec{y},\vec{x}\rangle$ is the same modulo $M_\perp=N^s$ and over the integers.

There are many ways to choose the parameters to satisfy this condition. We propose one possible way here.

Example 7.3.1 (Example of parameters for our DCR-based instantiation)**.** *Let M_y and M_x*

be positive integers such that $2M_yM_x + 1 \leq M_\perp = N^s$. *We set:*

$$\mathcal{M}_y := \{\vec{y} \in \mathbb{Z}^\ell \,:\, \|\vec{y}\| \leq M_y\}\,, \qquad \mathcal{M}_x := \{\vec{x} \in \mathbb{Z}^\ell \,:\, \|\vec{x}\| \leq M_x\}\,,$$
$$M := \ell \cdot 2^\kappa \cdot M_x \cdot |\Delta\mathcal{M}_x| \leq \ell \cdot 2^\kappa \cdot M_x \cdot (4 \cdot M_x)^\ell\,,$$

where $\|.\|$ *denotes the Euclidean norm, so that* $|\langle \vec{y}, \vec{x}\rangle| \leq M_yM_x$ *(when the inner-product is over the integers). For the last inequality, we use the rough inequality* $|\Delta\mathcal{M}_x| \leq (4 \cdot M_x)^\ell$.

Then, we fix M_y and M_x so that $2M_yM_x + 1 \leq M_\perp$. And we choose M so that M_x/M is negligible.

We are now ready to present our DCR-based IND-IPFE-CPA secure IPFE scheme, depicted on Figure 7.1. To make the scheme look mroe familiar, we use the multiplicative notation for $\mathbb{G}$: $\mathfrak{g}_1\mathfrak{g}_2 = |\mathfrak{g}_1\mathfrak{g}_2|$ where elements are represented in $\{-\frac{N^s-1}{2}, \ldots, \frac{N^s-1}{2}\}$.

- Let $N = pq$ be a product of two λ-bit random safe primes. We suppose that Condition 1 is satisfied. Let $u \xleftarrow{\$} \mathbb{Z}^*_{N^{s+1}}$, $\mathfrak{g} \leftarrow u^{2N^s}$, and $\mathfrak{g}_\perp \leftarrow 1 + N$.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Set $pp = (\kappa, \ell, N, \mathfrak{g})$. For $i = 1, \ldots, \ell$, set
$$\mathsf{hk}_i \xleftarrow{\$} \{0, \ldots, \lfloor MN^{s+1}/4\rfloor\} \in \mathbb{Z}\,, \qquad \mathfrak{hp}_i \leftarrow \mathfrak{g}^{\mathsf{hk}_i} \in \mathbb{G}\,.$$
Return $msk = \mathbf{hk} \in \mathbb{Z}^\ell$ and $mpk = \mathbf{\mathfrak{hp}} \in \mathbb{G}^\ell$.
- $\mathsf{KeyDer}(msk, \vec{y} \in \mathcal{M}_y) \mapsto sk_{\vec{y}}$: Set
$$\mathsf{hk}_{\vec{y}} \leftarrow \langle \vec{y}, \overrightarrow{\mathsf{hk}}\rangle \in \mathbb{Z}\,,$$
where the computation is done over $\mathbb{Z}$. Return $msk_{\vec{y}} = \mathsf{hk}_{\vec{y}} \in \mathbb{Z}$.
- $\mathsf{Encrypt}(mpk, \vec{x} \in \mathcal{M}_x) \mapsto ct_{\vec{x}}$: Sample $r \xleftarrow{\$} \{0, \ldots, \lfloor N/4\rfloor\}$ and set $\mathfrak{b} \leftarrow \mathfrak{g}^r$. Return (msk, mpk). For $i = 1, \ldots, \ell$, set
$$ct_i \leftarrow (1 + N)^{x_i} \cdot \mathfrak{hp}_i^r \in \mathbb{G}\,.$$
Return $ct_{\vec{x}} = (\mathfrak{b}, \overrightarrow{ct} \in \mathbb{G}^\ell)$.
- $\mathsf{Decrypt}(sk_{\vec{y}}, ct_{\vec{x}})$: Set
$$ct_{\langle \vec{x}, \vec{y}\rangle} \leftarrow \prod_{i=1}^{\ell} ct_i^{y_i} / \mathfrak{b}^{\mathsf{hk}_{\vec{y}}}\,.$$
Return $\frac{ct_{\langle \vec{x}, \vec{y}\rangle} - 1}{N} \bmod N^s$ (using the representation in $\{-\lfloor N^s/2\rfloor, \ldots, \lfloor N^s/2\rfloor\}$).

Figure 7.1: DCR-based inner-product functional encryption secure against chosen-plaintext attacks

Note that the sizes of our secret keys is slightly larger than those of [ALS16], due to our security reduction. In order to have better parameters, they do a more precise analysis of the computational entropy contained in the secret key and show that it is enough to hide the challenge vector. On the other hand, their scheme requires to sample discrete Gaussian

whereas we only need uniform values. Moreover, our scheme is length-flexible in the same sense as the cryptosystems of [DJ01; DJ03]. Namely, by fixing the parameter $s \in \mathbb{Z}^+$, one can obtain bigger or smaller sets M_x and M_y, which is not possible with the scheme presented by [ALS16]. Larger s however makes the scheme less efficient.

Applying Theorem 5.3.2 and Lemma 7.1.1, we immediately get the following security theorem.

Theorem 7.3.2. *Under the DCR assumption, the scheme $\mathcal{FE}$ depicted on Figure 7.1 is* IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the IND-IPFE-CPA *security of $\mathcal{FE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the DCR assumption, such that $\varepsilon_{\mathscr{A}} \leq 4s \cdot \varepsilon_{\mathscr{B}} + 16/\operatorname{spf}(N) + \ell \cdot |\Delta\mathcal{M}_x| \cdot M_x/M$.*

Using parameters from Example 7.3.1, we have the following security bound: $\varepsilon_{\mathscr{A}} \leq 4s \cdot \varepsilon_{\mathscr{B}} + 16/\operatorname{spf}(N) + 2^{-\kappa}$. Although there is an exponential loss in the security reduction of Theorem 5.3.2, we emphasize that there is no exponential loss using these parameters: the security loss is compensated by these well-chosen parameters. Most importantly, all the algorithms of the resulting scheme run in polynomial time (in the security parameter κ)[1] and the reduction to DCR is polynomial time. There is *no* complexity leveraging and we *do not* require subexponential assumption *nor* exponential-size keys or ciphertexts.

We also note that we could directly have another inner-product functional encryption scheme that is IND-IPFE-CPA secure by instantiating Construction 4.4.1 with the public key encryption scheme from [CS03] or [BCP03], but the resulting construction would have worse parameters than our construction from projective hash functions. These schemes are similar to the ElGamal public key encryption scheme, but in a composite group, which means that the message can be embeded in a subgroup where the discrete logarithm is easy to solve, as in our case with the construction from PHF.

7.3.2 Scheme Secure Against Chosen-Ciphertext Attacks

Let us now instantiate the framework for IND-TBIPFE-CCA secure tag-based inner-product functional encryption schemes with the DCR-based IPFE-CPA-friendly PHF defined in Section 7.2.1, and the DCR-based IPFE-CCA-friendly tag-based PHF defined in Section 7.2.2. We use the same parameters as for the IND-IPFE-CPA secure scheme in the previous section. We recall that we use here the multiplicative notation for $\mathbb{G}$: $\mathfrak{g}_1\mathfrak{g}_2 = |\mathfrak{g}_1\mathfrak{g}_2|$ where elements are represented in $\{-\frac{N^s-1}{2}, \ldots, \frac{N^s-1}{2}\}$. The resulting scheme is depicted on Figure 7.2.

Applying Theorem 5.5.2 and Lemma 7.1.1, we immediately get the following security theorem.

Theorem 7.3.3. *Under the DCR assumption, the scheme $\mathcal{TBFE}$ depicted in Figure 7.2 is* IND-TBIPFE-CCA.

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the IND-TBIPFE-CCA *security of $\mathcal{TBFE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the DCR assumption, such that $\varepsilon_{\mathscr{A}} \leq 4s \cdot \varepsilon_{\mathscr{B}} + 16/\operatorname{spf}(N) + \ell \cdot |\Delta\mathcal{M}_x| \cdot M_x \cdot (1/M + 2/M^{\dagger}) + 2 \cdot q_{dec} \cdot |\Delta\mathcal{M}_x|/2^{\nu}$.*

[1] We recall that the length ℓ of the vectors is assumed to be polynomial in κ.

- Let $N = pq$ be a product of two λ-bit random safe primes. We suppose that Condition 1 is satisfied. Let $u \xleftarrow{\$} \mathbb{Z}^*_{N^{s+1}}$, $\mathfrak{g} \leftarrow u^{2N^s}$, and $\mathfrak{g}_\perp \leftarrow 1 + N$.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Set $pp = (\kappa, \ell, N, \mathfrak{g})$. For $i = 1, \ldots, \ell$, $j = 1, 2$, and $k = 1, \ldots, \nu$, set

$$\mathsf{hk}_i \xleftarrow{\$} \{0, \ldots, \lfloor MN^{s+1}/4 \rfloor\} \in \mathbb{Z}\,, \qquad \mathfrak{hp}_i \leftarrow \mathfrak{g}^{\mathsf{hk}_i} \in \mathbb{G}\,,$$
$$\mathsf{hk}^\dagger_{k,i,j} \xleftarrow{\$} \{0, \ldots, \left\lfloor \nu M^\dagger N^{s+1}/2 \right\rfloor\} \in \mathbb{Z}\,, \qquad \mathfrak{hp}^\dagger_{k,i,j} \leftarrow \mathfrak{g}^{\mathsf{hk}^\dagger_{k,i,j}} \in \mathbb{G}\,.$$

  Return $msk = (\mathbf{hk} \in \mathbb{Z}^\ell, \overrightarrow{\mathbf{hk}}^\dagger \in (\mathbb{Z}^{2\times\nu})^\ell)$ and $mpk = (\mathfrak{hp} \in \mathbb{G}^\ell, \overrightarrow{\mathfrak{hp}}^\dagger \in (\mathbb{G}^{2\times\nu})^\ell)$.
- $\mathsf{KeyDer}(msk, \overrightarrow{y} \in \mathcal{M}_y) \mapsto sk_{\overrightarrow{y}}$: Set

$$\mathsf{hk}_{\overrightarrow{y}} \leftarrow \langle \overrightarrow{y}, \overrightarrow{\mathsf{hk}} \rangle \in \mathbb{Z}\,, \qquad \mathbf{hk}^\dagger_{\overrightarrow{y}} \leftarrow \langle \overrightarrow{y}, \overrightarrow{\mathbf{hk}}^\dagger \rangle \in \mathbb{Z}^{2\times\nu}\,,$$

  where the computation is done over $\mathbb{Z}$. Return $msk_{\overrightarrow{y}} = (\mathsf{hk}_{\overrightarrow{y}} \in \mathbb{Z}, \mathbf{hk}^\dagger_{\overrightarrow{y}} \in \mathbb{Z}^{2\times\nu})$.
- $\mathsf{Encrypt}(\tau, mpk, \overrightarrow{x} \in \mathcal{M}_x) \mapsto ct_{\overrightarrow{x}}$: Sample $r \xleftarrow{\$} \{0, \ldots, \lfloor N/4 \rfloor\}$ and set $\mathfrak{b} \leftarrow \mathfrak{g}^r$. Return (msk, mpk). For $i = 1, \ldots, \ell$ and $k = 1, \ldots, \nu$, set

$$ct_i \leftarrow (1+N)^{x_i} \cdot \mathfrak{hp}_i^r \in \mathbb{G}\,, \qquad \boldsymbol{ct}^\dagger_{k,i} = \mathfrak{g}^{r \cdot \mathfrak{hp}^\dagger_{i,k,1} + r \cdot \tau \cdot \mathfrak{hp}^\dagger_{i,k,2}} \in \mathbb{G}\,.$$

  Return $ct_{\overrightarrow{x}} = (\mathfrak{b}, \overrightarrow{ct} \in \mathbb{G}^\ell, \overrightarrow{\boldsymbol{ct}}^\dagger \in (\mathbb{G}^\nu)^\ell)$.
- $\mathsf{Decrypt}(\tau, sk_{\overrightarrow{y}}, ct_{\overrightarrow{x}})$: Check that $\sum_{i=1}^\ell y_i \cdot ct^\dagger_{i,k} = \mathfrak{b}^{\mathsf{hk}^\dagger_{\overrightarrow{y},1,k} + \tau \cdot \mathsf{hk}^\dagger_{\overrightarrow{y},2,k}}$ for $k = 1, \ldots, \nu$; return $\perp$ if any check fails.

  Set

$$ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle} \leftarrow \prod_{i=1}^\ell ct_i^{y_i} / \mathfrak{b}^{\mathsf{hk}_{\overrightarrow{y}}}\,.$$

  Return $\frac{ct_{\langle \overrightarrow{x}, \overrightarrow{y} \rangle} - 1}{N} \bmod N^s$ (using the representation in $\{-\lfloor N^s/2 \rfloor, \ldots, \lfloor N^s/2 \rfloor\}$).

Figure 7.2: DCR-based tag-based inner-product functional encryption secure against chosen-ciphertext attacks

Using parameters from Example 7.3.1 and setting $\nu \geq \log_{\mathrm{spf}(N)}(2 \cdot q_{\mathrm{dec}} \cdot |\Delta\mathcal{M}_x|) = O(\mathrm{poly}(\kappa))$, we have the following security bound: $\varepsilon_{\mathscr{A}} \leq 4s \cdot \varepsilon_{\mathscr{B}} + 16/\mathrm{spf}(N) + 4 \cdot 2^{-\kappa}$. We note that in general, for security reasons, $\mathrm{spf}(N)$ has to be a really big integer. In this case, one think of ν as a really small constant, i.e., 1, 2, or 3 to be really safe, depending on the message space wanted, and on the number of queries the adversary could potentially make.

Similarly to what happens in our DCR-based IND-IPFE-CPA secure instantiation in Section 7.3.1, although there is an exponential loss in the security reduction of Theorem 5.5.2, we emphasize that there is no exponential loss using these parameters: the security loss is compensated by these well-chosen parameters.

Chapter 8

Instantiations based on the Learning With Errors Assumption

In this chapter, we instantiate the two generic constructions of inner-product functional encryption from public key encryption with a lattice-based public key encryption scheme. As usual with lattice-based cryptography, or noise-based cryptography, the analysis of the scheme is more complicated than in the two previous case of schemes based on the hardness of the discrete logarithm or on the hardness of the factorization.
We give 2 concrete instantiations that reach respectively s-IND-IPFE-CPA and IND-IPFE-CPA security, and compare their trade-offs between efficiency and security, and even between security and security: the scheme that reaches the stronger notion of security uses a stronger assumption (that is less likely to hold).
These schemes have the advantage of being resistant against quantum attacks, and have efficient decryption, however, they are non-modular. They were first presented in our papers [ABDP15b] (details are only found in the full version [ABDP15a]) and [ABCP16].

Contents

8.1 The Learning With Errors Assumption **108**
8.1.1 The Learning With Errors Assumption 108
8.1.2 Hardness Evidences 108
8.2 Public Key Encryption Schemes Secure under the Learning With Error Assumption **109**
8.2.1 Regev Public Key Encryption Scheme 109
8.2.2 Additional Properties of Regev Public Key Encryption Scheme 110
8.3 Inner-Product Functional Encryption Schemes Based on the Learning With Errors Assumption **116**
8.3.1 Scheme Selectively Secure Against Chosen-Plaintext Attacks 116
8.3.2 Scheme Adaptively Secure Against Chosen-Plaintext Attacks 119

8.1 The Learning With Errors Assumption

In this section, we recall the *learning with errors* (LWE) assumption and discuss its pros and cons.

8.1.1 The Learning With Errors Assumption

A quantum computer would improve efficiency of many known and used algorithms. This includes the algorithms that factors big numbers and find discrete logarithm. These problems becoming efficiently solvable would be a huge issue for public key cryptography, as most of the public key crypto is based on the hardness of either discrete logarithm or factorization. Recently, a lot of work is being put in trying to find alternative encryption schemes that would be easily implemented on classical computer, but would remain secure even if someone tries to break it with a quantum computer. The most trendy assumption in this case is the *learning with errors* assumption [Reg05]. Informally, it states that giving noisy inner-products with random vectors completely hide a secret vector $\mathbf{s}$ and looks uniformly random. This easily yields a secret key encryption scheme that is directly converted into a public key encryption scheme using its additive homomorphic properties. Its hardness is related to lattice problems, hence the nickname of lattice-based cryptography. Those problems have been studied for years, which is comforting for a cryptographic assumption. However, a lot of recent works tend to improve the attacks against lattice-based cryptography, in particular when the parameters are overstretched. Let us first define this assumption, before discussing its parameters.

Learning With Errors. Let n, q be integer parameters. For any noise distribution χ on $\mathbb{Z}_q$, and vector $\mathbf{s} \in \mathbb{Z}_q^n$, the oracle $\mathsf{LWE}_{q,n,\chi}(\mathbf{s})$ samples a fresh random n-dimensional vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, as well as noise $e \leftarrow \chi$, and returns $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$. The LWE assumption states that $\mathsf{LWE}_{q,n,\chi}(\mathbf{s})$ is computationally indistinguishable from the uniform distribution on $\mathbb{Z}_q$.

We would like to point out that in this assumption, the distribution (either LWE distribution or uniform distribution) is given as an oracle, and can be called any polynomial number of times. This is why some works prefer the notation with matrices and vectors. Some formulations also work in the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ of reals modulo 1. This in fact is equivalent to considering different noise distributions. In this work, we will focus on the noise distribution χ_σ, a discrete Gaussian over $\mathbb{Z}$ centered on 0 with standard deviation σ.

It makes no doubt that the assumption has now been adopted as a standard, and has been use to create encryption schemes [Reg05], key exchanges [DXL12; Pei14; ADPS15; ADPS16], identity-based encryption [ABB10a], bi-deniable encryption [OPW11], and even the first instances of fully homomorphic encryption [BV11a; GSW13] (The very first construction of FHE [Gen09] is not based on LWE, but on a similar assumption), which allows to compute on encrypted data without learning anything about it.

8.1.2 Hardness Evidences

Since its introduction in machine learning, this problem has seen a lot of interest as a cryptographic assumption. There are two main reasons to this success: the LWE assumption allowed the construction of primitives that have been the holy grail of cryptography for decades: fully homomorphic encryption; the second reason is because it is believed to be hard to solve for quantum computers. Let us first talk about the general hardness of LWE,

and then talk about the importance of the parameters and their impact on the hardness of the assumption.

Reductions from Lattice Problems. Regev showed in [Reg05] that there is a quantum reduction from the approximate shortest vector problem (GAP-SVP), a lattice problem believed to be hard, even for quantum computers, to LWE that is preserving the dimension n if the standard deviation σ is greater than $2\sqrt{n}$. Following that reductions, several works tried to dequantize the reduction, like the one of Peikert [Pei09] and more recently a fully classical reduction has been proven in [BLP+13]. Unfortunately, this reduction is not dimension preserving, making this result incomparable to Regev's one.

Modulus-to-Noise Ratio. When measuring the hardness of LWE, the two main parameters to consider are the dimension n of the vectors, and the modulus-to-noise ratio q/σ. The higher the dimension is, the harder it is to distinguish between the $\mathsf{LWE}_{q,n,\chi}(\mathbf{s})$ distribution, and the uniform distribution. Usually, the dimension $n = O(\kappa)$ is of the same order of magnitude as the security parameter κ. On the other hand, increasing the modulus-to-noise ratio q/σ decreases the difficulty of the problem. This is why in general it is better to aim at having a polynomial ratio $q/\sigma = O(\text{poly}(\kappa))$. Moreover, while there is no known attacks on using a superpolynomial modulus-to-noise ratio, there is less evidences of its hardness.

8.2 Public Key Encryption Schemes Secure under the Learning With Error Assumption

In this section, we present a variant of the original public key encryption scheme presented in [Reg05] and show that it can be used to instantiate Construction 4.3.1 and Construction 4.4.1. We also discuss the choice of parameters needed for the needed properties to be verified.

8.2.1 Regev Public Key Encryption Scheme

Choosing the Parameters. As in [PW08; KTX08], we use encryptions of messages in $\mathbb{Z}_p$ instead of bits in the case of the original encryption scheme. We let the message space be $M = \{0, \ldots, M_x\} \subseteq \mathbb{Z}_p$ for some integer M_x, n be an integer related to the security parameter, p be the modulus of the message space (for the IPFE schemes, we will require that $p > \ell M_x M_y$, which is big enough for the inner-product t not wrap around the modulus), $q > p$ be the modulus for the ciphertext and keys. Our public keys and ciphertexts consist of matrices and vectors over $\mathbb{Z}_q$. For every $v \in \mathbb{Z}_p$ (i.e., one entry of a message vector), define the "center" for v as $t(v) = v \cdot \left\lfloor \frac{q}{p} \right\rfloor \in \mathbb{Z}_q$. Let χ_σ denote an integer gaussian distribution over $\mathbb{Z}_q$ with standard deviation σ: $\chi_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(\mathbb{Z})}$. Let $m = m(n)$, and $\sigma = \sigma(n)$ and $\sigma' = \sigma'(n)$ be positive real Gaussian parameters. The following relations between parameters are required for correctness and security of the public key encryption scheme:

1. $m > (1 + \epsilon)(n + 1) \log q$;
2. $\sigma q > 2\sqrt{n}$;
3. $\frac{q}{2p} > \sigma\sqrt{2m\kappa}$.

The first one is needed to extract the computational entropy contained in the randomness of the encryption with the leftover hash lemma and is used in the security proof.
The second one is required for the hardness of the underlying LWE assumption.
And the last one is required for correctness, to ensure that the noise doesn't alter the message.
Other conditions are required for the additional properties. We will state those conditions when proving that the schemes satisfy the property, and we will recall all needed conditions when showing the concrete IPFE constructions. All operations are performed over $\mathbb{Z}_q$.

Construction 8.2.1 (Regev PKE Scheme)**.** *We recall the* Regev public-key encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$*:*

- $\mathsf{Setup}(1^\kappa) \mapsto (pp, sk, pk)$*. On input security parameter* κ *samples* $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$*,* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$*, and* $\mathbf{e} \xleftarrow{\$} \chi_\sigma^m$*, computes* $pk \leftarrow \boldsymbol{A}\mathbf{s}+\mathbf{e} \in \mathbb{Z}_q^m$*, and outputs public parameters* $pp = (\boldsymbol{A}, \chi_\sigma)$*, secret key* $sk = \mathbf{s}$*, and public key* pk*;*

- $\mathsf{Encrypt}(pp, pk, m) \mapsto ct$*. On input public parameter* pp*, public key* pk*, and plaintext* m*, samples* $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$*, computes*

$$ct_0 \leftarrow \boldsymbol{A}^\intercal \mathbf{r} \in \mathbb{Z}_q^n\,, \qquad\qquad ct_1 \leftarrow \langle pk, \mathbf{r}\rangle + t(x) \in \mathbb{Z}_q\,,$$

  where $t(x) = x \cdot \lfloor q/p \rfloor \in \mathbb{Z}_q$*, and outputs ciphertext* $ct = (ct_0, ct_1)$*;*

- $\mathsf{Decrypt}(pp, sk, ct) \mapsto m$ *or* $\perp$*. On input public parameters* pp*, secret key* sk*, and ciphertext* ct *computes*

$$d = ct_1 - \langle ct_0, \mathbf{s}\rangle,$$

  and returns the plaintext $x \in M$*, where* x *is such that* $d - t(x) \in \mathbb{Z}_q$ *is closest to* $0 \bmod q$*.*

We note that the scheme might be slightly optimized by sampling $\mathbf{r}$ from $\{-1,0,1\}^m$ instead of $\{0,1\}^m$, increasing the computational entropy of a ciphertext without increasing the noise at decryption time, but we prefer to keep the latter set for simplicity and to stay closer to the original scheme.

Correctness. In the decryption algorithm, $d = t(x) + \langle \mathbf{e}, vecr\rangle$, which means that the algorithm is correct if $\langle \mathbf{e}, \mathbf{r}\rangle$ is smaller than $\frac{q}{2p}$. $\langle \mathbf{e}, \mathbf{r}\rangle$ is a sum of m discrete Gaussians with standard deviation σ so it is subgaussian with standard deviation at most $\sigma\sqrt{m}$, and hence has magnitude less than $\sigma\sqrt{2m\kappa}$ with overwhelming probability. Our setting of parameters thus ensures that the decryption is correct.

8.2.2 Additional Properties of Regev Public Key Encryption Scheme

We now go through the additional properties defined in Section 4.2, and show that the Regev encryption scheme defined in Construction 8.2.1 satisfies these properties.

Structure. As required, we can split up the Setup algoritm into two parts:

- $\mathsf{SKGen}(1^\kappa)$ on input the security parameter and samples $\mathbf{s}$ from $\mathbb{Z}_q^n$ and outputs $sk = \mathbf{s}$;

- $\mathsf{PKGen}(sk, \tau)$ on input secret key $\mathbf{s}$, parameters τ, and samples $\mathbf{e} \xleftarrow{\$} \chi_\sigma^m$ and computes $pk \leftarrow \boldsymbol{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$. Then the algorithm outputs pk.

  Notice that, if τ describes an error distribution then $\mathbf{e}$ is sampled from this latter distribution.

As can we do with the $\mathsf{Encrypt}$ algorithm:

- $\mathsf{C}(\mathbf{r})$ on input a random vector $\mathbf{r} \in \{0,1\}^m$ outputs $ct_0 \leftarrow \boldsymbol{A}^\intercal \mathbf{r}$;
- $\mathsf{E}(pk, x; \mathbf{r})$ on input public key pk, message x and random vector $\mathbf{r}$, returns $ct_1 \leftarrow \langle pk, \mathbf{r}\rangle + t(x)$.

Semantic Security. Proof of the semantic security of this encryption scheme can be found in [Reg05]. It relies on the LWE assumption and on the leftover hash lemma, which is a statistical argument. It proceeds in two steps with a hybrid argument: first, we replace the public key by a uniformly random vector. This is indistinguishable from the security game thanks to the LWE assumption. Then we replace the ciphertext by a uniform vector. This is statistically indistinguishable thanks to the leftover hash lemma. Then we conclude by observing that the view is now independent of the challenge bit.

Linear Key Homomorphism. The first property comes from the fact that secret keys are elements uniformly sampled from the group $\mathbb{Z}_q^n$, so the secret key space is stable under addition. Moreover, $\sum \alpha_i \mathbf{s}_i$ is a correct secret key for $\sum \alpha_i(\boldsymbol{A}\mathbf{s}_i + \mathbf{e}_i)$ as long as $\sum \alpha_i \mathbf{e}_i$ remains small, which is true for small values of α_i. This is taken into account in our setting of parameters. The scheme will remain correct as long as σ' verifies the correctness condition, where $\sigma' = \sigma\|\vec{\alpha}\|$ is a bound on the standard deviation of the new error term.

Linear Ciphertext Homomorphism under Shared Randomness. It is easy to verify that:

$$\langle pk_1, \mathbf{r}\rangle + t(x_1) + \langle pk_2, \mathbf{r}\rangle + t(x_2) = \langle pk_1 + pk_2, \mathbf{r}\rangle + t(x_1 + x_2),$$

by the definition of the function t.

ℓ-Public-Key Reproducibility. To show that scheme of Construction 8.2.1 has *ℓ-public-key-reproducibility*, for any fixed constant ℓ, it is sufficient to show that there are error distributions with standard deviations $\sigma, \sigma', (\sigma_i)_{i\in[\ell]}$ such that $\mathsf{Exp}_{\mathcal{E},\kappa}^{\ell\text{-pk-rep-0}}(\mathscr{A})$ is indistinguishable from $\mathsf{Exp}_{\mathcal{E},\kappa}^{\ell\text{-pk-rep-1}}(\mathscr{A})$. The idea is that when taking a discrete Gaussian error e and multiplying it by a vector $\vec{\alpha}$, we obtain ℓ correlated Gaussians, so in order to obtain a spherical Gaussian distribution in the end, we add correctly correlated errors in order to unskew the distribution. However, we can only increase the noise by adding something, so we require that the final error distribution used in the scheme has standard deviation $\sigma \geq (1 + B\sqrt{\ell})\sigma'$, where B is a bound on the size of each α_i and σ' is the standard deviation of the error in the LWE assumption.

Theorem 8.2.2. *Construction 8.2.1 has* ℓ-public-key-reproducibility *in a statistical sense.*

Proof. Let τ be the description of an error distribution with standard deviation σ, and τ' an error distribution with standard deviation σ'. A simple calculation show that the covariance of $e \cdot \vec{\alpha}$ is $\sigma'^2 \vec{\alpha}\vec{\alpha}^\intercal$. Hence the choice of τ_i to unskew the Gaussian distribution. Let $\{\tau_i\}_{i\in[\ell]}$ describe sampling ℓ errors with covariance matrix $\sqrt{\Sigma}$, where

$\Sigma = \sigma^2 I_\ell - \sigma'^2 \vec{\alpha}\vec{\alpha}^\intercal$, where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_\ell)$. Notice that Σ is positive semi-definite if $\sigma > \sigma'(1 + B\sqrt{\ell})$ because α_i is smaller than B for any i.

Finally, let $\beta = \begin{pmatrix} I_\ell & \vec{\alpha} \end{pmatrix}$ and $\Sigma' = \begin{pmatrix} \Sigma & 0 \\ 0 & \sigma'^2 \end{pmatrix}$, where I_ℓ is the identity matrix.

Then, If $b = 0$, the errors appearing in pk_i come from the distribution $\chi_{\sigma I_\ell}$. If $b = 1$, the errors appearing in pk_i come from the distribution $\beta\chi_{\sqrt{\Sigma'}}$.

We show that these two distribution are statistically close if $\sigma > \sigma'(1 + B\sqrt{\ell})$. Let us set $\beta' = \begin{pmatrix} I_\ell & \vec{\alpha} \\ \vec{\mu}^\intercal & 1 + \vec{\mu}^\intercal\vec{\alpha} \end{pmatrix}$ for some $\vec{\mu}$. Let $\Sigma_0 = \beta'\sqrt{\Sigma'}(\beta'\sqrt{\Sigma'})^\intercal$ be of the target form $\begin{pmatrix} \sigma^2 I_\ell & 0 \\ 0 & {\gamma_0}^2 \end{pmatrix}$. If Σ_0 has this form, it means that $\beta'\sqrt{\Sigma'}$ gives us ℓ uncorrelated errors distributed as χ_σ and another error distributed as χ_{γ_0} which we can drop because it is not correlated to the other. Then $\forall \mathbf{z}, \epsilon \leftarrow \chi_{\sqrt{\Sigma'}}$

$$\begin{aligned}
\mathbf{Pr}(\beta\epsilon = \mathbf{z}) &= \sum_s \mathbf{Pr}\left(\beta'\epsilon = \begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\intercal\mathbf{z} + s \end{pmatrix}\right) \\
&= \sum_s \mathbf{Pr}\left(\epsilon = \beta'^{-1}\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\intercal\mathbf{z} + s \end{pmatrix}\right) \\
&\propto \sum_s \rho_{\sqrt{\Sigma'}}\left(\beta'^{-1}\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\intercal\mathbf{z} + s \end{pmatrix}\right) \\
&\propto \sum_s \rho_{\beta'\sqrt{\Sigma'}}\left(\begin{pmatrix} \mathbf{z} \\ \vec{\mu}^\intercal\mathbf{z} + s \end{pmatrix}\right) \text{ by Lemma 2.4.5} \\
&\propto \sum_s \rho_{\sqrt{\Sigma_0}}(\mathbf{z})\rho_{\gamma_0}(\vec{\mu}^\intercal\mathbf{z} + s) \\
&\propto \rho_{\sqrt{\Sigma_0}}(\mathbf{z})\rho_{\gamma_0}(\vec{\mu}^\intercal\mathbf{z} + \mathbb{Z}) \\
&\propto \nu\rho_{\sqrt{\Sigma_0}}(\mathbf{z}) \text{ where } \nu \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right]
\end{aligned}$$

as long as $\gamma_0 > \sqrt{\frac{\ln(2\ell(1+1/\epsilon))}{\pi}}$ by Lemma 2.4.6 and Corollary 2.4.3 □

ℓ-Ciphertext Reproducibility. We show that the scheme of Construction 8.2.1 has *ℓ-ciphertext-reproducibility* for any fixed constant ℓ as long as $m \geq (n + \ell + 1)\log q + 2\log\frac{1}{\epsilon} + \Omega(1)$, by taking error distributions with standard deviations $\sigma', (\sigma_i)_{i\in[\ell]}$ as chosen for the ℓ-public-key-reproducibility, and by the following alternative encryption algorithm

$$\mathsf{E}'((\mathbf{s}_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}') = \langle \mathbf{s}_i, ct_0\rangle + \langle \mathbf{e}_i, \mathbf{r}'\rangle + t(x_i) \ .$$

as required. This is enough to show that $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\kappa}(\mathscr{A})$ is indistinguishable from $\mathsf{Exp}^{\ell\text{-ct-rep-1}}_{\mathcal{E},\kappa}(\mathscr{A})$.

Theorem 8.2.3. *Under the LWE assumption, Construction 8.2.1 has* ℓ-ciphertext-reproducibility.

Proof. We prove the theorem via a sequence of hybrid experiments.

Hybrid H_1: This is the $\mathsf{Exp}^{\ell\text{-ct-rep-0}}_{\mathcal{E},\mathcal{M},\kappa}(\mathscr{A})$, with the algorithms unfold.

proc Initialize$(\kappa, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda)$
$sk \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi_{\sigma'}^m, pk = \boldsymbol{A}sk + \mathbf{e}$
$pk_i = \boldsymbol{A}sk_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{\$} \{0,1\}^m$
$ct_0 = \boldsymbol{A}^\intercal \mathbf{r}, ct = \mathsf{E}(pk, a; \mathbf{r})$
$ct_i = \alpha_i ct + \mathsf{E}(pk_i, x_i; \mathbf{r})$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

Hybrid H_2: This is like H_1 except that pk is taken uniformly random in $\mathbb{Z}_q^m$.

proc Initialize$(\kappa, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda)$
$\boxed{pk \xleftarrow{\$} \mathbb{Z}_q^m}$
$pk_i = \boldsymbol{A}sk_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{\$} \{0,1\}^m$
$ct_0 = \boldsymbol{A}^\intercal \mathbf{r}, ct = \mathsf{E}(pk, a; \mathbf{r})$
$ct_i = \alpha_i ct + \mathsf{E}(pk_i, x_i; \mathbf{r})$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

The hardness of LWE guarantees that pk looks pseudo-random to the adversary. Moreover notice that sk is never used.

Hybrid H_3: This is like H_2 except that the ciphertext is computed with E' instead of E.

proc Initialize$(\kappa, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda)$
$pk \xleftarrow{\$} \mathbb{Z}_q^m$
$pk_i = \boldsymbol{A}sk_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{\$} \{0,1\}^m$
$ct_0 = \boldsymbol{A}^\intercal \mathbf{r}, ct = \mathsf{E}(pk, a; \mathbf{r})$
$ct_i = \alpha_i ct + \boxed{\mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r})}$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]})$

proc Finalize(b')
Return $(b' = b)$

Notice that $\mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}) = \mathsf{E}(pk_i, x_i; \mathbf{r})$ since $ct_0 = \boldsymbol{A}^\intercal \mathbf{r}$.

Hybrid H_4: This is like H_3 except that ct_0 and $\langle pk, \mathbf{r}\rangle$ are replaced with uniformly

random values.

$$\boxed{\begin{array}{l}
\underline{\textbf{proc Initialize}(\kappa, \mathcal{M})} \\
(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda) \\
pk \xleftarrow{\$} \mathbb{Z}_q^m, \boxed{u \xleftarrow{\$} \mathbb{Z}_q} \\
pk_i = \boldsymbol{A} sk_i + \mathbf{e}_i, \text{ for } \mathbf{e}_i \leftarrow \chi_{\sigma_i}^m \\
\mathbf{r} \xleftarrow{\$} \{0,1\}^m \\
\boxed{ct_0 \xleftarrow{\$} \mathbb{Z}_q^m, ct = u + t(a)} \\
ct_i = \alpha_i ct + \mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}) \\
\text{Return } (pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]}) \\
\\
\underline{\textbf{proc Finalize}(b')} \\
\text{Return } (b' = b)
\end{array}}$$

Let us define the following random variables:

- X is the random variable that takes uniform values of the form $(\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}, \mathbf{b} \in \mathbb{Z}_q^n)$.
- W is the random variable that takes uniform values of the form $\mathbf{r} \in \{0,1\}^m$.
- I is the random variable that takes values of the form $(\langle \mathbf{e}_1, \mathbf{r}\rangle, \ldots, \langle \mathbf{e}_\ell, \mathbf{r}\rangle)$, where $\mathbf{e}_i \leftarrow \chi^m$, $\mathbf{r} \in \{0,1\}^m$.

Then, by Lemma 2.4.7, we have that $\tilde{\mathbf{H}}_\infty(W|I) \geq \mathbf{H}_\infty(W) - (\ell-1)\log q = m - (\ell-1)\log q$. Now, notice that $H_X(W) = H_{(\boldsymbol{A},\mathbf{b})}(\mathbf{r}) = (\boldsymbol{A}^\intercal \mathbf{r}, \langle \mathbf{b}, \mathbf{r}\rangle)$ is a universal hash function and by applying the generalized leftover hash lemma (Lemma 2.4.8), we have that:

$$\Delta((H_X(W), X, I), (U, X, I)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\mathbf{H}}_\infty(W|I)} q^{n+1}} \ .$$

Then, if $m \geq (n+\ell+1)\log q + 2\log\frac{1}{\epsilon} + \Omega(1)$, the statistical distance between the two views is at most ϵ.

Hybrid H_5: This is like H_4 except that $\mathbf{r}$ is replaced by another random value $\mathbf{r}'$.

$$\boxed{\begin{array}{l}
\underline{\textbf{proc Initialize}(\kappa, \mathcal{M})} \\
(a, (\alpha_i, x_i, sk_i)_{i\in[\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda) \\
pk \xleftarrow{\$} \mathbb{Z}_q^m, u \xleftarrow{\$} \mathbb{Z}_q \\
pk_i = \boldsymbol{A} sk_i + \mathbf{e}_i, \text{ for } \mathbf{e}_i \leftarrow \chi_{\sigma_i}^m \\
\boxed{\mathbf{r}' \xleftarrow{\$} \{0,1\}^m} \\
ct_0 \xleftarrow{\$} \mathbb{Z}_q^m, ct = u + t(a) \\
ct_i = \alpha_i ct + \boxed{\mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}')} \\
\text{Return } (pk, (\alpha_i, pk_i, sk_i)_{i\in[\ell]}, ct_0, (ct_i)_{i\in[\ell]}) \\
\\
\underline{\textbf{proc Finalize}(b')} \\
\text{Return } (b' = b)
\end{array}}$$

These are exactly the same distribution as $\mathbf{r}$ is used nowhere else.

Hybrid H_6: This is like H_5 except that ct_0 is generated as $\boldsymbol{A}^\intercal \mathbf{r}$ and u is replaced by $\langle pk, \mathbf{r} \rangle$.

proc Initialize$(\kappa, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i \in [\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda)$
$\boxed{pk \xleftarrow{\$} \mathbb{Z}_q^m}$
$pk_i = \boldsymbol{A} sk_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\boxed{\mathbf{r} \xleftarrow{\$} \{0,1\}^m, \mathbf{r}' \xleftarrow{\$} \{0,1\}^m}$
$\boxed{ct_0 = \boldsymbol{A}^\intercal \mathbf{r}, ct = \mathsf{E}(pk, a; \mathbf{r})}$
$ct_i = \alpha_i ct + \mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}')$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i \in [\ell]}, ct_0, (ct_i)_{i \in [\ell]})$

proc Finalize(b')
Return $(b' = b)$

The change from H_6 to H_5 is the same as the change from H_3 to H_4, except that no information about $\mathbf{r}$ is leaked. So Lemma 2.4.7 gives us that the statistical distance between the two views is at most ϵ.

Hybrid H_7: This is the $\mathsf{Exp}_{\mathcal{E},\mathcal{M},\kappa}^{\ell\text{-ct-rep-1}}(\mathscr{A})$.

proc Initialize$(\kappa, \mathcal{M})$
$(a, (\alpha_i, x_i, sk_i)_{i \in [\ell]}) \xleftarrow{\$} \mathcal{M}(1^\lambda)$
$\boxed{sk \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi_{\sigma'}^m, pk = \boldsymbol{A} sk + \mathbf{e}}$
$pk_i = \boldsymbol{A} sk_i + \mathbf{e}_i$, for $\mathbf{e}_i \leftarrow \chi_{\sigma_i}^m$
$\mathbf{r} \xleftarrow{\$} \{0,1\}^m, \mathbf{r}' \xleftarrow{\$} \{0,1\}^m$
$ct_0 = \boldsymbol{A}^\intercal \mathbf{r}, ct = \mathsf{E}(pk, a; \mathbf{r})$
$ct_i = \alpha_i ct + \mathsf{E}'((sk_i, \mathbf{e}_i), x_i, ct_0; \mathbf{r}')$
Return $(pk, (\alpha_i, pk_i, sk_i)_{i \in [\ell]}, ct_0, (ct_i)_{i \in [\ell]})$

proc Finalize(b')
Return $(b' = b)$

Once again, the hardness of LWE guarantees that pk looks pseudo-random to the adversary. □

We would like to note that reusing the same randomness for different public keys for packing messages was already shown to be secure in [PVW08], but the requirements we made in the generic constructions are stronger.

We are now ready to use this encryption scheme to construct inner-product functional encryption schemes.

8.3 Inner-Product Functional Encryption Schemes Based on the Learning With Errors Assumption

In this section, we present the two schemes secure under the LWE assumption obtained by plugging Regev public key encryption scheme described in the previous section in Construction 4.3.1 and Construction 4.4.1. The first one reaches selective security against chosen-plaintext attacks, while the second one reaches adaptive security. The price to pay for adaptivity in this case is a bit heavy, because we need to rely on a stronger assumption where the ratio between the modulus and the standard deviation of the error is superpolynomial. We would like to point out that for the case of adaptive security, [ALS16] reaches better parameter than us and can use a polynomial modulus-to-noise ratio at the price of the genericity of the framework. The main difference between their construction and ours is that they base their IPFE scheme on the Dual Regev public key encryption scheme, which is the same as the one we presented, but interverting the roles of the secret key and the encryption randomness. Then, they use a more subtle statistical argument to conclude their proof, instead of our argument that switches secret keys to change the challenge message. As the constructions presented in Section 7.3, our two IPFE schemes secure under the LWE assumption have efficient decryption, unlike the schemes of Section 6.4, but suffer the same downside: they are non-modular. We provide a comparison of those two schemes in Table 8.1, where we present the trade-offs they offer between security and efficiency.
We note that no construction from projective hash functions is given in this chapter. The main reason is because we don't currently know how to build PHFs with the required properties for reaching IND-IPFE-CCA security. There have been some proposition of projective hash functions from LWE [KV09; BCDP13] that we might be able to use to construct adaptive IND-IPFE-CPA secure scheme, but we felt that the construction from public key encryption was conceptually simpler. The complexity of building PHFs based on lattices already shows up when trying to define a language for your words: the set of ciphertexts that are output by encrypting 0 is different to the set of ciphertexts that decrypts to 0. That is because the decryption must be correct for all ciphertexts returned by encrypt, and adding a really small noise to this ciphertext shouldn't change the decryption value.

8.3.1 Scheme Selectively Secure Against Chosen-Plaintext Attacks

The first scheme is the less secure one. However, it has a pretty good efficiency compared to other schemes that are based on lattices, like fully homomorphic encryption or attribute-based encryption. All operations used are pretty fast, the main downside for the efficiency is that the vectors and matrices can be huge depending on the choice of parameters. If the message space needs to be big, we recommend using schemes based on the DCR assumption, because the message space is set by the security parameter and is very big. One noticeable advantage of this scheme compared to those based on other assumptions is that it is resistant against quantum attacks. We assume that the conditions required are verified:

1. $m \geq (n + \ell + 1) \log q + 2 \log \frac{1}{\epsilon} + \Omega(1)$;
2. $\sigma \geq (1 + M_x \sqrt{\ell})\sigma'$;
3. $\sigma' q > 2\sqrt{n}$;
4. $p > \ell M_x M_y$;

Security	Conditions on the parameters
s-IND-IPFE-CPA	1. $m \geq (n+\ell+1)\log q + 2\log\frac{1}{\epsilon} + \Omega(1)$; 2. $\sigma \geq (1+M_x\sqrt{\ell})\sigma'$; 3. $\sigma' q > 2\sqrt{n}$; 4. $p > \ell M_x M_y$; 5. $\frac{q}{2p} > \sigma M_y\sqrt{2\ell m\kappa}$.
IND-IPFE-CPA	1. $m \geq (n+\ell+2)\log q + 2\log\frac{1}{\epsilon} + \Omega(1)$; 2. $T = M_x \cdot \kappa^{\omega(1)}$; 3. $\sigma \geq (1+T\sqrt{\ell})\sigma'$; 4. $\gamma_0 > \sqrt{\frac{\ln(2\ell(1+1/\epsilon))}{\pi}}$; 5. $\sigma' q > 2\sqrt{n}$; 6. $p > \ell M_x M_y$; 7. $\frac{q}{2p} > \sigma M_y^2\ell\sqrt{2m\kappa}$.

Security	s-IND-IPFE-CPA	IND-IPFE-CPA
mpk	$m\ell$	$m\ell$
msk	$n\ell$	$(n+1)\ell$
$sk_{\vec{y}}$	n	$n+1$
$ct_{\vec{x}}$	$n+\ell$	$n+\ell+1$

Table 8.1: Comparison of the two different LWE-based inner-product functional encryption schemes. The sizes are given as number of elements in $\mathbb{Z}_q$. We note that the second element of the secret key for the IND-IPFE-CPA secure scheme is conceptually an integer, but we count it as 1 element in $\mathbb{Z}_q$ since it is smaller than q with our setting of parameters.

5. $\frac{q}{2p} > \sigma M_y\sqrt{2\ell m\kappa}$.

Our s-IND-IPFE-CPA secure inner-product functional encryption scheme based on LWE is depicted on Figure 8.1.

Correctness. The proof of correctness follows the same path as the one for the basic public key encryption scheme. This time, the final error has standard deviation at most $\sigma M_y\sqrt{m\ell}$. The correctness then follows from the setting of parameters.

Remark 8.3.1 (Optimization of the message space.)**.** *It is possible to gain some efficiency and reduce the parameters size by reducing the bound M_y on each coordinate of the key*

- Let n, m, p, q be integer parameters and σ a positive real parameter such that they verify the conditions required. Let $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ be a uniformly random matrix.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Set $pp = (\kappa, \ell, n, m, p, q, \boldsymbol{A})$. For $i = 1, \ldots, \ell$, set

$$(\mathbf{s}_i, \mathbf{e}_i) \xleftarrow{\$} \mathbb{Z}_q^n \times \chi_\sigma^m \,, \qquad \mathbf{b}_i \leftarrow \boldsymbol{A}\mathbf{s}_i + \mathbf{e}_i \in \mathbb{Z}_q^m \,.$$

  Return $msk = (\mathbf{s}_i)_{i \in [\ell]}$ and $mpk = (\mathbf{b}_i)_{i \in [\ell]}$;
- $\mathsf{KeyDer}(msk, \mathbf{y} \in \mathcal{M}_y) \mapsto sk_{\vec{y}}$: Return

$$sk_{\mathbf{y}} \leftarrow \sum_{i \in [\ell]} y_i \mathbf{s}_i \in \mathbb{Z}_q^n \,;$$

- $\mathsf{Encrypt}(mpk, \mathbf{x} \in \mathcal{M}_x) \mapsto ct_{\mathbf{x}}$: Pick $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$ and set

$$ct_0 \leftarrow \boldsymbol{A}^\intercal \mathbf{r} \in \mathbb{Z}_q^n \,.$$

  For $i = 1, \ldots, \ell$, set

$$ct_i \leftarrow \mathbf{b}_i^\intercal \mathbf{r} + t(x_i) \in \mathbb{Z}_q \,,$$

  where $t(v) = v \cdot \lfloor q/p \rfloor \in \mathbb{Z}_q$. Return $ct_{\mathbf{x}} = (ct_0, (ct_i)_{i \in [\ell]})$;
- $\mathsf{Decrypt}(sk_{\mathbf{y}}, ct_{\mathbf{x}})$: Set

$$ct_{\langle \mathbf{x}, \mathbf{y} \rangle} \leftarrow \sum_{i \in [\ell]} y_i ct_i - ct_0^\intercal sk_{\mathbf{y}} \in \mathbb{Z}_q.$$

  Return the plaintext m, where m is such that $d - t(m) \in \mathbb{Z}_q$ is closest to $0 \mod q$.

Figure 8.1: LWE-based inner-product functional encryption secure against selective chosen-plaintext attacks

vectors $\mathbf{y}$*. In order to achieve this, we decompose each coordinate of* $\mathbf{y}$ *in binary and for each coordinate of* $\mathbf{x}$ *we encrypt it multiplied by each of the powers of two. Then the inner-product will reconstruct* $\mathbf{y}$ *and the scheme will be correct. This makes* ℓ *grow by* $\log M_y$ *and decreases* M_y *to* 1*, which greatly improves on the correctness condition, allowing us to use better parameters.*
A bit more formally, the new scheme consists in writing $y_i = \sum_{j \in \{0,\ldots,\gamma\}} 2^j y_{i,j}$ *for each coordinate* y_i *of* $\mathbf{y}$*, where* $\gamma = \log M_y$*, and use the IPFE schemes replacing vectors:*

$$\mathbf{y} \mapsto (y_{i,j})_{i \in \ell, j \in \{0,\ldots,\gamma\}} \,, \qquad \mathbf{x} \mapsto (2^j x_i)_{i \in \ell, j \in \{0,\ldots,\gamma\}} \,.$$

It is easy to see that $\sum_{i \in \ell, j \in \{0,\ldots,\gamma\}} y_{i,j} 2^j x_i = \sum_{i \in \ell, j \in \{0,\ldots,\gamma\}} y_i x_i = \langle \mathbf{x}, \mathbf{y} \rangle$.

Security.

By using Theorem 4.3.3, we immediately get the following security theorem for our construction.

Theorem 8.3.2. *Under the LWE assumption with parameters n, q, σ', the scheme $\mathcal{FE}$ depicted on Figure 8.1 is* s-IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the s-IND-IPFE-CPA *security of $\mathcal{FE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the LWE assumption, such that $\varepsilon_{\mathscr{A}} \leq \varepsilon_{\mathscr{B}}$.*

8.3.2 Scheme Adaptively Secure Against Chosen-Plaintext Attacks

Our second scheme reaches adaptive security, which usually is a hard problem when dealing with lattice-based cryptography, because of the noisy nature of the schemes. It is more costly than the previous scheme that was only selectively secure, but if adaptive security is a necessity, it is still more efficient than using the previous scheme and increasing the security paramater to do complexity leveraging. We assume that the conditions required are verified:

1. $m \geq (n + \ell + 2) \log q + 2 \log \frac{1}{\epsilon} + \Omega(1)$;
2. $T = M_x \cdot \kappa^{\omega(1)}$;
3. $\sigma \geq (1 + T\sqrt{\ell})\sigma'$;
4. $\gamma_0 > \sqrt{\frac{\ln(2\ell(1+1/\epsilon))}{\pi}}$;
5. $\sigma' q > 2\sqrt{n}$;
6. $p > \ell M_x M_y$;
7. $\frac{q}{2p} > \sigma M_y^2 \ell \sqrt{2m\kappa}$.

Our IND-IPFE-CPA secure inner-product functional encryption scheme based on LWE is depicted on Figure 8.2.

Correctness. The proof of correctness follows the same path as the one for the basic public key encryption scheme. This time, the final error has standard deviation at most $\sigma M_y \sqrt{m\ell} + \sigma' \ell T M_y$, which is less than $\sigma M_y^2 \ell \sqrt{m}$ because of the relation between σ and σ'. The correctness then follows from the setting of parameters.

Remark 8.3.1 still applies in this case, giving the same improvements with the same construction.

Security.

By using Theorem 4.4.2, we immediately get the following security theorem for our construction.

Theorem 8.3.3. *Under the LWE assumption with parameters n, q, σ', the scheme $\mathcal{FE}$ depicted on Figure 8.2 is* IND-IPFE-CPA *secure.*

More precisely, if there exists an attacker $\mathscr{A}$ that has advantage $\varepsilon_{\mathscr{A}}$ in breaking the IND-IPFE-CPA *security of $\mathcal{FE}$, then there exists an attacker $\mathscr{B}$ that runs in approximately the same time and that has advantage $\varepsilon_{\mathscr{B}}$ in breaking the LWE assumption, such that $\varepsilon_{\mathscr{A}} \leq \varepsilon_{\mathscr{B}}$.*

- Let n, m, p, q be integer parameters and σ a positive real parameter such that they verify the conditions required. Let $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ be a uniformly random matrix.
- $\mathsf{Setup}(1^\kappa, 1^\ell) \mapsto (mpk, msk)$: Set $pp = (\kappa, \ell, n, m, p, q, \boldsymbol{A})$, and sample

$$(\mathbf{s}_0, \mathbf{e}_0) \xleftarrow{\$} \mathbb{Z}_q^n \times \chi_{\gamma_0}^m \,, \qquad \mathbf{b}_0 \leftarrow \boldsymbol{A}\mathbf{s}_0 + \mathbf{e}_0 \in \mathbb{Z}_q^m \,.$$

  For $i = 1, \ldots, \ell$, set

$$(t_i, \mathbf{s}_i, \mathbf{e}_i) \xleftarrow{\$} \{0, \ldots, T\} \times \mathbb{Z}_q^n \times \chi_\sigma^m \,, \qquad \mathbf{b}_i \leftarrow \boldsymbol{A}(t_i \cdot \mathbf{s}_0 + \mathbf{s}_i) + \mathbf{e}_i \in \mathbb{Z}_q^m \,.$$

  Return $msk = (\mathbf{s}_i, t_i)_{i \in [\ell]}$ and $mpk = (\mathbf{b}_0, \mathbf{b}_i)_{i \in [\ell]}$;
- $\mathsf{KeyDer}(msk, \mathbf{y} \in \mathcal{M}_y) \mapsto sk_{\vec{y}}$: Set

$$\mathbf{s}_\mathbf{y} \leftarrow \sum_{i \in [\ell]} y_i \mathbf{s}_i \in \mathbb{Z}_q^n \,, \qquad t_\mathbf{y} \leftarrow \sum_{i \in [\ell]} y_i t_i \in \mathbb{Z} \,.$$

  Return $sk_\mathbf{y} = (\mathbf{s}_\mathbf{y}, t_\mathbf{y})$
- $\mathsf{Encrypt}(mpk, \mathbf{x} \in \mathcal{M}_x) \mapsto ct_\mathbf{x}$: Pick $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$ and set

$$ct_0 \leftarrow \boldsymbol{A}^\intercal \mathbf{r} \in \mathbb{Z}_q^n \,, \qquad ct_1 \leftarrow \mathbf{b}_0^\intercal \mathbf{r} \in \mathbb{Z}_q \,.$$

  For $i = 1, \ldots, \ell$, set

$$ct_{2,i} \leftarrow \mathbf{b}_i^\intercal \mathbf{r} + t(x_i) \in \mathbb{Z}_q \,,$$

  where $t(v) = v \cdot \lfloor q/p \rfloor \in \mathbb{Z}_q$. Return $ct_\mathbf{x} = (ct_0, ct_1, (ct_{2,i})_{i \in [\ell]})$;
- $\mathsf{Decrypt}(sk_\mathbf{y}, ct_\mathbf{x})$: Set

$$ct_{\langle \mathbf{x}, \mathbf{y} \rangle} \leftarrow \sum_{i \in [\ell]} y_i ct_{2,i} - t_\mathbf{y} ct_1 - ct_0^\intercal sk_\mathbf{y} \in \mathbb{Z}_q.$$

  Return the plaintext m, where m is such that $d - t(m) \in \mathbb{Z}_q$ is closest to $0 \mod q$.

Figure 8.2: LWE-based inner-product functional encryption secure against adaptive chosen-plaintext attacks

Chapter 9

Conclusion and Open Questions

9.1 Conclusion

In this thesis, we introduced the notion of *inner-product functional encryption* (IPFE) which is the simplest non-trivial example of functional encryption that allows partial decryption of the plaintext. IPFE finds many theoretical and practical applications, and can be built efficiently from standard assumption.

We analyzed the different notions of security for IPFE, and showed how to generically build a partial function hiding secret key IPFE scheme from a standard public key modular IPFE scheme. This result can find some nice direct applications, and can also be used to build a secret key IPFE that is function hiding against bounded collusions of adversaries.

We presented 4 generic constructions based on 2 different albeit similar frameworks: constructions from public key encryption and from projective hash functions. The first framework is a bit more general as it encompasses some lattice-based IPFE schemes, while the second is more powerful as it allows the construction of schemes secure against chosen-ciphertext attacks, at the cost of efficiency.

We proposed 3 instantiations based on the decisional Diffie-Hellman assumption. All of them suffer the same drawback of having inefficient decryption restricting its input to polynomial size. However, the schemes are very efficient and simple, and compute modular inner-products. One of the schemes is secure against chosen-ciphertext attacks.

We also proposed 2 instantiations based on the decisional composite residuosity in order to avoid the problem of inefficient decryption. The sizes of the ciphertexts are larger than the constructions based on DDH, but the message space is flexible, and these schemes have very good asymptotical efficiency for very large message spaces. One of these schemes is also secure against chosen-ciphertext attacks, with a lower incremental cost than the scheme based on DDH.

Lastly, we proposed 2 instantiations based on the learning with errors assumptions. Those also avoid the problem of inefficient decryption, but the security against adaptive adversary is costly compared to the schemes based on the other assumptions. However, this scheme

is really interesting because it is resistant against attacks from quantum computers, and it might be used in conjunction with other lattice-based tools in order to create more powerful primitives.

9.2 Open Questions

Here, I give a few open problems and directions that might be interesting for future works. The first question is also the most obvious.

Question 9.1. *Can we build a modular inner-product functional encryption with efficient decryption and stateless algorithms?*

This would combine the best of all possible worlds. But interestingly enough, all our constructions have some drawbacks that cannot make them fit the nice definition of IPFE.

Question 9.2. *Can we build an inner-product functional encryption secure against chosen-ciphertext attacks under the learning with errors assumption?*

As noted previously, attempts at building projective hash functions from lattices don't give primitives powerful enough to build CCA secure schemes. However, there are some instances of lattice-based CCA schemes using different approaches.

Question 9.3. *Can we build a secret key function hiding inner-product functional encryption without pairings?*

Or, more generally:

Question 9.4. *How far can we go, only using the decisional Diffie-Hellman assumption?*

This approach is different than the one that tries to build functional encryption for all circuits: instead of picking a problem and trying to solve it at all cost, we take one tool and see what we can get from it. Our work is partly based on this question. Partial function hiding is another example of trying to squeeze every drop we can out of the simplest standard assumptions.

Question 9.5. *How expressive can we make functional encryption using standard assumptions?*

We know that using pairings, we can increase the range of possibilities to bilinear functions, or function hiding, or multi-inputs. On the other hand, recent works show that it is plausible that multilinear map with a degree slightly bigger than 2 would imply functional encryption for all circuits, and indistinguishability obfuscation. So it would seem like we've reached the maximum we can without reaching functional encryption for all circuits. There is still room for improvement, for example one could try and build functional encryption for some randomized functionality.

Notation

General Mathematical Notations	
$\mathbb{N}$	The set of natural numbers
$\mathbb{Z}$	The set of integers
$\mathbb{R}$	The set of reals
log	The base-2 logarithm
ln	The natural logarithm
$[n]$	The numbers from 1 to n
$\vec{x}, \mathbf{x}$	A (column) vector
$\boldsymbol{A}$	A matrix
$\langle \cdot, \cdot \rangle$	The inner-product operation
$\lvert \vec{x} \rvert$	The size of a vector
$\lvert S \rvert$	The cardinality of a set
Δ	The statistical distance
$\propto$	Is proportional to
o, O, ω, Ω	Asymptotic notations
$\mathscr{A}, \mathscr{B}, \cdots$	Algorithms, adversaries
General Cryptographic Notations	
κ	The security parameter
pk	A public key
sk	A secret key
ct	A ciphertext
mpk	A master public key
msk	A master secret key
$sk_{\mathbf{y}}$	A user secret key
Notations Specific to Chapter 5	
$\mathbf{P}$	A subset membership problem
$\mathcal{L}$	An NP language
$\mathfrak{b}$	A word
w	A witness
hk	A hashing key
$\mathfrak{hp}$	A projection key
$\mathfrak{H}$	A hash
Notations Specific to Chapter 6	
q	a prime number
$\mathbb{G}$	A cyclic group
$\mathfrak{g}$	A generator
Notations Specific to Chapter 7	
p, q	Safe primes
N	Product of safe primes
$\mathrm{spf}(N)$	Smallest prime factor of N
$\left(\frac{a}{N}\right)$	Jacobi symbol of a
$\mathbb{J}_N$	Elements of Jacobi symbol 1
$\mathbb{QR}_N$	Squares of $\mathbb{Z}_N^*$
$\mathbb{QR}_N^*$	The group of signed quadratic residues
Notations Specific to Chapter 8	
Λ	A lattice
$\eta_\epsilon(\Lambda)$	The smoothing parameter of Λ
ρ	The Gaussian function
χ	A noise distribution
σ	A Gaussian parameter
χ_σ	A discrete Gaussian distribution

Abbreviations

Primitives	
PKE	Public Key Encryption
FE	Functional Encryption
IPFE	Inner-Product Functional Encryption
TBIPFE	Tag-Based Inner-Product Functional Encryption
PHF	Projective Hash Function
OTS	One-Time Signature
CRH	Collision Resistant Hash Function
Security Notions	
s-IND-CPA	Indistinguishability under Selective Chosen-Plaintext Attacks (for PKE schemes)
s-IND-IPFE-CPA	Indistinguishability under Selective Chosen-Plaintext Attacks (for IPFE schemes)
IND-IPFE-CPA	Indistinguishability under Chosen-Plaintext Attacks (for IPFE schemes)
IND-IPFE-CCA	Indistinguishability under Chosen-Ciphertext Attacks (for IPFE schemes)
IND-TBIPFE-CCA	Indistinguishability under Chosen-Ciphertext Attacks (for TBIPFE schemes)
SIM	Simulation Security
NA-SIM	Non-Adaptive Simulation Security
Assumptions	
DDH	Decisional Diffie-Hellman
MDDH	Matrix Decisional Diffie-Hellman
DCR	Decisional Composite Residuosity
LWE	Learning With Errors

List of Illustrations

Figures

2.1 s-IND-CPA security of a PKE scheme. 12

3.1 Illustration of an Inner-Product Functional Encryption scheme. 18
3.2 Illustration of the IND-IPFE-CPA security game. 20
3.3 IND-IPFE-CPA security of an IPFE scheme. 20
3.4 s-IND-IPFE-CPA security of an IPFE scheme. 23
3.5 NA-SIM security of an IPFE scheme. 26
3.6 Illustration of the IND-IPFE-CCA security game. 28
3.7 IND-IPFE-CCA security of an IPFE scheme. 29
3.8 IND-TBIPFE-CCA security of a TBIPFE scheme. 30
3.9 Function hiding of an IPFE scheme. 33
3.10 Function hiding of a secret key IPFE scheme. 34
3.11 Partial function hiding of a secret key IPFE scheme. 36

4.1 ℓ-public-key-reproducibility of a PKE scheme. 45
4.2 ℓ-ciphertext-reproducibility of a PKE scheme. 46
4.3 Illustration for randomness reuse. 48

5.1 A zero knowledge proof using projective hash functions. **b** is in the language. 65
5.2 A zero knowledge proof using projective hash functions. **b** is not in the language. 65
5.3 Illustration of key homomorphism. 67
5.4 Construction of Inner-Product Functional Encryption from Key Homomorphic Projective Hash Function. 70
5.5 Strategy for the proof of Theorem 5.3.2. 73
5.6 Tag-Based Projective Hash Functions. 75
5.7 Encrypt algorithm for IND-IPFE-CCA secure IPFE schemes. 78

6.1 DDH-based inner-product functional encryption secure against selective chosen-plaintext attacks 93
6.2 DDH-based inner-product functional encryption secure against chosen-plaintext attacks 94
6.3 DDH-based tag-based inner-product functional encryption secure against chosen-ciphertext attacks 96

7.1 DCR-based inner-product functional encryption secure against chosen-plaintext attacks 104
7.2 DCR-based tag-based inner-product functional encryption secure against chosen-ciphertext attacks 106

8.1 LWE-based inner-product functional encryption secure against selective chosen-plaintext attacks . . . 118
8.2 LWE-based inner-product functional encryption secure against adaptive chosen-plaintext attacks . . . 120

Tables

6.1 Comparison of DDH-based IPFE schemes. . . . 93

7.1 Comparison of the DCR-based IPFE schemes. . . . 103

8.1 Comparison of the LWE-based IPFE schemes. . . . 117

Bibliography

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. "Efficient Lattice (H)IBE in the Standard Model". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 553–572 (cit. on pp. 2, 108).

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. "Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 98–115 (cit. on p. 2).

[ABCP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. *Better Security for Functional Encryption for Inner Product Evaluations*. Cryptology ePrint Archive, Report 2016/011. http://eprint.iacr.org/2016/011. 2016 (cit. on pp. 4, 5, 17, 41, 107).

[ABDP15a] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. *Simple Functional Encryption Schemes for Inner Products*. Cryptology ePrint Archive, Report 2015/017. http://eprint.iacr.org/2015/017. 2015 (cit. on pp. 4, 107).

[ABDP15b] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. "Simple Functional Encryption Schemes for Inner Products". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33 (cit. on pp. 2, 3, 5, 17, 41, 85, 107).

[ABDP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. *Better Security for Functional Encryption for Inner Product Evaluations*. Cryptology ePrint Archive, Report 2016/011. http://eprint.iacr.org/2016/011. 2016 (cit. on p. 95).

[ABP+17] Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. "Efficient Public Trace and Revoke from Standard Assumptions: Extended Abstract". In: *ACM CCS 17*. ACM Press, 2017, pp. 2277–2293 (cit. on p. 3).

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Disjunctions for Hash Proof Systems: New Constructions and Applications". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 69–100. DOI: 10.1007/978-3-662-46803-6_3 (cit. on pp. 74, 75, 90).

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. "From Selective to Adaptive Security in Functional Encryption". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 657–677. DOI: 10.1007/978-3-662-48000-7_32 (cit. on p. 23).

[ABV+12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. "Functional Encryption for Threshold Functions (or Fuzzy IBE) from Lattices". In: *PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. LNCS. Springer, Heidelberg, May 2012, pp. 280–297 (cit. on p. 2).

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. "Smooth Projective Hashing for Conditionally Extractable Commitments". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 671–689 (cit. on p. 64).

[ADPS15] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. *Post-quantum key exchange - a new hope*. Cryptology ePrint Archive, Report 2015/1092. http://eprint.iacr.org/2015/1092. 2015 (cit. on p. 108).

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. *NewHope without reconciliation*. Cryptology ePrint Archive, Report 2016/1157. http://eprint.iacr.org/2016/1157. 2016 (cit. on p. 108).

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. "Multi-input Inner-Product Functional Encryption from Pairings". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Heidelberg, May 2017, pp. 601–626 (cit. on p. 3).

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Functional Encryption: New Perspectives and Lower Bounds". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 500–518. DOI: 10.1007/978-3-642-40084-1_28 (cit. on pp. 2, 25).

[ALS15] Shweta Agrawal, Benoît Libert, and Damien Stehlé. *Fully Secure Functional Encryption for Inner Products, from Standard Assumptions*. Cryptology ePrint Archive, Report 2015/608. http://eprint.iacr.org/2015/608. 2015 (cit. on p. 26).

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. "Fully Secure Functional Encryption for Inner Products, from Standard Assumptions". In: *CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. LNCS. Springer, Heidelberg, Aug. 2016, pp. 333–362. DOI: 10.1007/978-3-662-53015-3_12 (cit. on pp. 3, 4, 19, 37, 66, 71, 95, 103–105, 116).

[AP14] Jacob Alperin-Sheriff and Chris Peikert. "Faster Bootstrapping with Polynomial Error". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 297–314. DOI: 10.1007/978-3-662-44371-2_17 (cit. on pp. 2, 15).

[AR17] Shweta Agrawal and Alon Rosen. "Functional Encryption for Bounded Collusions, Revisited". In: *TCC 2017, Part I*. LNCS. Springer, Heidelberg, Mar. 2017, pp. 173–205 (cit. on p. 3).

[AS17] Prabhanjan Ananth and Amit Sahai. "Projective Arithmetic Functional Encryption and Indistinguishability Obfuscation from Degree-5 Multilinear Maps". In: *EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. LNCS. Springer, Heidelberg, May 2017, pp. 152–181 (cit. on p. 3).

[BB04] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 443–459 (cit. on p. 2).

[BBL17] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. "CCA-Secure Inner-Product Functional Encryption from Projective Hash Functions". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Heidelberg, Mar. 2017, pp. 36–66 (cit. on pp. 4, 5, 17, 63, 85, 97).

[BBS03] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. "Randomness Re-use in Multi-recipient Encryption Schemeas". In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, Heidelberg, Jan. 2003, pp. 85–99 (cit. on p. 48).

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 41–55 (cit. on p. 86).

[BCDP13] Olivier Blazy, Céline Chevalier, Léo Ducas, and Jiaxin Pan. *Exact Smooth Projective Hash Function based on LWE*. Cryptology ePrint Archive, Report 2013/821. http://eprint.iacr.org/2013/821. 2013 (cit. on p. 116).

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. *Practical Functional Encryption for Quadratic Functions with Applications to Predicate Encryption*. Cryptology ePrint Archive, Report 2017/151. http://eprint.iacr.org/2017/151. 2017 (cit. on p. 3).

[BCP03] Emmanuel Bresson, Dario Catalano, and David Pointcheval. "A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications". In: *ASIACRYPT 2003*. Ed. by Chi-Sung Laih. Vol. 2894. LNCS. Springer, Heidelberg, Nov. 2003, pp. 37–54. DOI: 10.1007/978-3-540-40061-5_3 (cit. on p. 105).

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. "Relations Among Notions of Security for Public-Key Encryption Schemes". In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Heidelberg, Aug. 1998, pp. 26–45 (cit. on p. 27).

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 213–229 (cit. on p. 2).

[BF13] Manuel Barbosa and Pooya Farshim. "On the Semantic Security of Functional Encryption Schemes". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 143–161. DOI: 10.1007/978-3-642-36362-7_10 (cit. on p. 2).

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. "Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 533–556. DOI: 10.1007/978-3-642-55220-5_30 (cit. on p. 44).

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. "Function-Hiding Inner Product Encryption". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 470–491. DOI: 10.1007/978-3-662-48797-6_20 (cit. on pp. 3, 32).

[BJL16] Fabrice Benhamouda, Marc Joye, and Benoît Libert. "A New Framework for Privacy-Preserving Aggregation of Time-Series Data". In: *ACM Trans. Inf. Syst. Secur.* 18.3 (Mar. 2016). ISSN: 1094-9224/2016. DOI: 10.1145/2873069 (cit. on p. 67).

[BL17] Johannes Blömer and Gennadij Liske. *Subtleties in Security Definitions for Predicate Encryption with Public Index*. Cryptology ePrint Archive, Report 2017/453. http://eprint.iacr.org/2017/453. 2017 (cit. on p. 27).

[Ble98] Daniel Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Heidelberg, Aug. 1998, pp. 1–12 (cit. on p. 27).

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 575–584 (cit. on p. 109).

[BO13] Mihir Bellare and Adam O'Neill. "Semantically-Secure Functional Encryption: Possibility Results, Impossibility Results and the Quest for a General Definition". In: *CANS 13*. Ed. by Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab. Vol. 8257. LNCS. Springer, Heidelberg, Nov. 2013, pp. 218–234. DOI: 10.1007/978-3-319-02937-5_12 (cit. on pp. 2, 25).

[BP16] Zvika Brakerski and Renen Perlman. "Lattice-Based Fully Dynamic Multi-key FHE with Short Ciphertexts". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 190–213. DOI: 10.1007/978-3-662-53018-4_8 (cit. on p. 44).

[BPMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. "FHE Circuit Privacy Almost for Free". In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 62–89. DOI: 10.1007/978-3-662-53008-5_3 (cit. on p. 4).

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. "Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions". In: *TCC 2012*. Ed. by Ronald Cramer. Vol. 7194. LNCS. Springer, Heidelberg, Mar. 2012, pp. 94–111 (cit. on p. 64).

[BR06] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 409–426 (cit. on p. 10).

[BS15] Zvika Brakerski and Gil Segev. "Function-Private Functional Encryption in the Private-Key Setting". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 306–324. DOI: 10.1007/978-3-662-46497-7_12 (cit. on p. 32).

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges". In: *TCC 2011*. Ed. by Yuval Ishai. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011, pp. 253–273 (cit. on pp. 2, 12, 25).

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient Fully Homomorphic Encryption from (Standard) LWE". In: *52nd FOCS*. Ed. by Rafail Ostrovsky. IEEE Computer Society Press, Oct. 2011, pp. 97–106 (cit. on p. 108).

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 505–524 (cit. on p. 2).

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. "Circuit-ABE from LWE: Unbounded Attributes and Semi-adaptive Security". In: *CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. LNCS. Springer, Heidelberg, Aug. 2016, pp. 363–384. DOI: 10.1007/978-3-662-53015-3_13 (cit. on p. 2).

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. "Universally Composable Password-Based Key Exchange". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 404–421 (cit. on p. 64).

[CM15] Michael Clear and Ciaran McGoldrick. "Multi-identity and Multi-key Leveled FHE from Learning with Errors". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 630–656. DOI: 10.1007/978-3-662-48000-7_31 (cit. on p. 44).

[CO17] Wutichai Chongchitmate and Rafail Ostrovsky. "Circuit-Private Multi-key FHE". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Heidelberg, Mar. 2017, pp. 241–270 (cit. on p. 44).

[CS01] Ronald Cramer and Victor Shoup. *Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption*. Cryptology ePrint Archive, Report 2001/085. http://eprint.iacr.org/2001/085. 2001 (cit. on p. 89).

[CS02] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 45–64 (cit. on pp. 13, 64, 66, 68, 76, 77, 100).

[CS03] Jan Camenisch and Victor Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 126–144 (cit. on p. 105).

[CS98] Ronald Cramer and Victor Shoup. "A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack". In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Heidelberg, Aug. 1998, pp. 13–25 (cit. on pp. 29, 66).

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. "Functional Encryption for Inner Product with Full Function Privacy". In: *PKC 2016, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9614. LNCS. Springer, Heidelberg, Mar. 2016, pp. 164–195. DOI: 10.1007/978-3-662-49384-7_7 (cit. on pp. 3, 32).

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on pp. 1, 86).

[DIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. "On the Achievability of Simulation-Based Security for Functional Encryption". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 519–535. DOI: 10.1007/978-3-642-40084-1_29 (cit. on p. 25).

[DJ01] Ivan Damgård and Mats Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System". In: *PKC 2001*. Ed. by Kwangjo Kim. Vol. 1992. LNCS. Springer, Heidelberg, Feb. 2001, pp. 119–136 (cit. on pp. 98, 100, 105).

[DJ03] Ivan Damgård and Mads Jurik. "A Length-Flexible Threshold Cryptosystem with Applications". In: *ACISP 03*. Ed. by Reihaneh Safavi-Naini and Jennifer Seberry. Vol. 2727. LNCS. Springer, Heidelberg, July 2003, pp. 350–364. DOI: 10.1007/3-540-45067-X_30 (cit. on p. 105).

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 523–540 (cit. on pp. 15, 16).

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. *A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem*. Cryptology ePrint Archive, Report 2012/688. http://eprint.iacr.org/2012/688. 2012 (cit. on p. 108).

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 129–147. DOI: 10.1007/978-3-642-40084-1_8 (cit. on pp. 86, 89).

[ElG85] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *IEEE Transactions on Information Theory* 31 (1985), pp. 469–472 (cit. on pp. 86, 87).

[FN94] Amos Fiat and Moni Naor. "Broadcast Encryption". In: *CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 480–491 (cit. on p. 2).

[FS97] Roger Fischlin and Claus-Peter Schnorr. "Stronger Security Proofs for RSA and Rabin Bits". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 267–279 (cit. on p. 98).

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, May 2009, pp. 169–178 (cit. on pp. 2, 108).

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th FOCS*. IEEE Computer Society Press, Oct. 2013, pp. 40–49 (cit. on p. 2).

[GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. "Bounded Ciphertext Policy Attribute Based Encryption". In: *ICALP 2008, Part II*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Vol. 5126. LNCS. Springer, Heidelberg, July 2008, pp. 579–591 (cit. on p. 2).

[GL03] Rosario Gennaro and Yehuda Lindell. "A Framework for Password-Based Authenticated Key Exchange". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. http://eprint.iacr.org/2003/032.ps.gz. Springer, Heidelberg, May 2003, pp. 524–543 (cit. on p. 64).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information". In: *14th ACM STOC*. ACM Press, May 1982, pp. 365–377 (cit. on p. 98).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption". In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299 (cit. on pp. 21, 98).

[GMW15] Romain Gay, Pierrick Méaux, and Hoeteck Wee. "Predicate Encryption for Multi-dimensional Range Queries from Lattices". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 752–776. DOI: 10.1007/978-3-662-46447-2_34 (cit. on p. 2).

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 218–229 (cit. on p. 2).

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". In: *ACM CCS 06*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. Available as Cryptology ePrint Archive Report 2006/309. ACM Press, Oct. 2006, pp. 89–98 (cit. on p. 2).

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5 (cit. on pp. 2, 108).

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Functional Encryption with Bounded Collusions via Multi-party Computation". In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 162–179 (cit. on p. 2).

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Attribute-based encryption for circuits". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 545–554 (cit. on p. 2).

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Predicate Encryption for Circuits from LWE". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 503–523. DOI: 10.1007/978-3-662-48000-7_25 (cit. on p. 2).

[HK09] Dennis Hofheinz and Eike Kiltz. "The Group of Signed Quadratic Residues and Applications". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 637–653 (cit. on p. 98).

[HKS13] Dennis Hofheinz, Eike Kiltz, and Victor Shoup. "Practical Chosen Ciphertext Secure Encryption from Factoring". In: *Journal of Cryptology* 26.1 (Jan. 2013), pp. 102–118. DOI: 10.1007/s00145-011-9115-0 (cit. on p. 98).

[JOA96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996, p. 780 (cit. on p. 14).

[Kal05] Yael Tauman Kalai. "Smooth Projective Hashing and Two-Message Oblivious Transfer". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 78–95 (cit. on p. 64).

[Kil06] Eike Kiltz. "Chosen-Ciphertext Security from Tag-Based Encryption". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Heidelberg, Mar. 2006, pp. 581–600 (cit. on p. 30).

[KKS17] Sungwook Kim, Jinsu Kim, and Jae Hong Seo. *A New Approach for Practical Function-Private Inner Product Encryption*. Cryptology ePrint Archive, Report 2017/004. http://eprint.iacr.org/2017/004. 2017 (cit. on p. 32).

[KLM+16] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. *Function-Hiding Inner Product Encryption is Practical*. Cryptology ePrint Archive, Report 2016/440. http://eprint.iacr.org/2016/440. 2016 (cit. on p. 32).

[KM14] Franziskus Kiefer and Mark Manulis. "Distributed Smooth Projective Hashing and Its Application to Two-Server Password Authenticated Key Exchange". In: *ACNS 14*. Ed. by Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay. Vol. 8479. LNCS. Springer, Heidelberg, June 2014, pp. 199–216. DOI: 10.1007/978-3-319-07536-5_13 (cit. on p. 64).

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 146–162 (cit. on p. 2).

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. "Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems". In: *ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Vol. 5350. LNCS. Springer, Heidelberg, Dec. 2008, pp. 372–389 (cit. on p. 109).

[Kur02] Kaoru Kurosawa. "Multi-recipient Public-Key Encryption with Shortened Ciphertext". In: *PKC 2002*. Ed. by David Naccache and Pascal Paillier. Vol. 2274. LNCS. Springer, Heidelberg, Feb. 2002, pp. 48–63 (cit. on p. 48).

[KV09] Jonathan Katz and Vinod Vaikuntanathan. "Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 636–652 (cit. on pp. 64, 116).

[KY09] Jonathan Katz and Arkady Yerukhimovich. "On Black-Box Constructions of Predicate Encryption from Trapdoor Permutations". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 197–213 (cit. on p. 2).

[Lin16a] Huijia Lin. "Indistinguishability Obfuscation from Constant-Degree Graded Encoding Schemes". In: *EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 28–57. DOI: 10.1007/978-3-662-49890-3_2 (cit. on p. 3).

[Lin16b] Huijia Lin. *Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs*. Cryptology ePrint Archive, Report 2016/1096. http://eprint.iacr.org/2016/1096. 2016 (cit. on p. 3).

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 62–91 (cit. on p. 2).

[LT17] Huijia Lin and Stefano Tessaro. *Indistinguishability Obfuscation from Trilinear Maps and Block-Wise Local PRGs*. Cryptology ePrint Archive, Report 2017/250. http://eprint.iacr.org/2017/250. 2017 (cit. on p. 3).

[LV16] Huijia Lin and Vinod Vaikuntanathan. "Indistinguishability Obfuscation from DDH-Like Assumptions on Constant-Degree Graded Encodings". In: *57th FOCS*. Ed. by Irit Dinur. IEEE Computer Society Press, Oct. 2016, pp. 11–20. DOI: 10.1109/FOCS.2016.11 (cit. on p. 3).

[MR07] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *SIAM J. Comput.* 37.1 (Apr. 2007), pp. 267–302. ISSN: 0097-5397. DOI: 10.1137/S0097539705447360. URL: http://dx.doi.org/10.1137/S0097539705447360 (cit. on p. 15).

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. "Alternatives to Non-malleability: Definitions, Constructions, and Applications (Extended Abstract)". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Heidelberg, Feb. 2004, pp. 171–190 (cit. on p. 29).

[MW16] Pratyay Mukherjee and Daniel Wichs. "Two Round Multiparty Computation via Multi-key FHE". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 735–763. DOI: 10.1007/978-3-662-49896-5_26 (cit. on p. 44).

[NP15] Mridul Nandi and Tapas Pandit. *Generic Conversions from CPA to CCA secure Functional Encryption*. Cryptology ePrint Archive, Report 2015/457. http://eprint.iacr.org/2015/457. 2015 (cit. on p. 27).

[ONe10] Adam O'Neill. *Definitional Issues in Functional Encryption*. Cryptology ePrint Archive, Report 2010/556. http://eprint.iacr.org/2010/556. 2010 (cit. on pp. 12, 25, 26).

[OPW11] Adam O'Neill, Chris Peikert, and Brent Waters. "Bi-Deniable Public-Key Encryption". In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 525–542 (cit. on p. 108).

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. "Attribute-based encryption with non-monotonic access structures". In: *ACM CCS 07*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM Press, Oct. 2007, pp. 195–203 (cit. on p. 2).

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. "Homomorphic Encryption and Signatures from Vector Decomposition". In: *PAIRING 2008*. Ed. by Steven D. Galbraith and Kenneth G. Paterson. Vol. 5209. LNCS. Springer, Heidelberg, Sept. 2008, pp. 57–74 (cit. on p. 37).

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. "Hierarchical Predicate Encryption for Inner-Products". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 214–231 (cit. on pp. 2, 37).

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Heidelberg, May 1999, pp. 223–238 (cit. on pp. 98, 100).

[Pei09] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem: extended abstract". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, May 2009, pp. 333–342 (cit. on p. 109).

[Pei14] Chris Peikert. *Lattice Cryptography for the Internet*. Cryptology ePrint Archive, Report 2014/070. http://eprint.iacr.org/2014/070. 2014 (cit. on p. 108).

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. "A Framework for Efficient and Composable Oblivious Transfer". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Springer, Heidelberg, Aug. 2008, pp. 554–571 (cit. on p. 115).

[PW08] Chris Peikert and Brent Waters. "Lossy trapdoor functions and their applications". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 187–196 (cit. on p. 109).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93 (cit. on pp. 108, 109, 111).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signature and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126 (cit. on pp. 1, 98).

[SW05] Amit Sahai and Brent R. Waters. "Fuzzy Identity-Based Encryption". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 457–473 (cit. on p. 2).

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. "Efficient Functional Encryption for Inner-Product Values with Full-Hiding Security". In: *ISC 2016*. Ed. by Matt Bishop and Anderson C. A. Nascimento. Vol. 9866. LNCS. Springer, Heidelberg, Sept. 2016, pp. 408–425. DOI: 10.1007/978-3-319-45871-7_24 (cit. on pp. 3, 32).

[Wat09] Brent Waters. "Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 619–636 (cit. on p. 2).

[Wat11] Brent Waters. "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization". In: *PKC 2011*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. LNCS. Springer, Heidelberg, Mar. 2011, pp. 53–70 (cit. on p. 2).

[Wee12] Hoeteck Wee. "Dual Projective Hashing and Its Applications - Lossy Trapdoor Functions and More". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 246–262 (cit. on p. 64).

[Wee16] Hoeteck Wee. "KDM-Security via Homomorphic Smooth Projective Hashing". In: *PKC 2016, Part II*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9615. LNCS. Springer, Heidelberg, Mar. 2016, pp. 159–179. DOI: 10.1007/978-3-662-49387-8_7 (cit. on p. 64).

[Yao82] Andrew Chi-Chih Yao. "Protocols for Secure Computations (Extended Abstract)". In: *23rd FOCS*. IEEE Computer Society Press, Nov. 1982, pp. 160–164 (cit. on p. 2).

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)". In: *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167 (cit. on p. 2).

[Yun84] Moti Yung. "Cryptoprotocols: Subscription to a Public Key, the Secret Blocking and the Multi-Player Mental Poker Game (Extended Abstract)". In: *CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. LNCS. Springer, Heidelberg, Aug. 1984, pp. 439–453 (cit. on p. 2).

Résumé

Le chiffrement fonctionnel est une technique émergente en cryptographie dans laquelle une autorité toute puissante est capable de distribuer des clés permettant d'effectuer des calculs sur des données chiffrées de manière contrôlée. La mode dans ce domaine est de construire des schémas qui sont aussi expressifs que possible, c'est-à-dire du chiffrement fonctionnel qui permet l'évaluation de n'importe quel circuit. Ces contributions délaissent souvent l'efficacité ainsi que la sécurité. Elles reposent sur des hypothèses fortes, très peu étudiées, et aucune construction n'est proche d'être pratique.

Le but de cette thèse est d'attaquer ce défi sous un autre angle: nous essayons de construire des schémas de chiffrement fonctionnel les plus expressifs que nous le pouvons en se basant sur des hypothèses standards, tout en conservant la simplicité et l'efficacité des constructions.

C'est pourquoi nous introduisons la notion de chiffrement fonctionnel pour l'évaluation de produits scalaires, où les messages sont des vecteurs $\vec{x}$, et l'autorité peut transmettre des clés correspondants à des vecteurs $\vec{y}$ qui permettent l'évaluation du produit scalaire $\langle \vec{x}, \vec{y} \rangle$. Cette fonctionnalité possède immédiatement des applications directes, et peut aussi être utilisé dans d'autres constructions plus théoriques, le produit scalaire étant une opération couramment utilisée.

Enfin, nous présentons deux structures génériques pour construire des schémas de chiffrement fonctionnels pour le produit scalaire, ainsi que des instanciations concrètes dont la sécurité repose sur des hypothèses standards. Nous comparons aussi les avantages et inconvénients de chacune d'entre elles.

Abstract

Functional encryption is an emerging framework in which a master authority can distribute keys that allow some computation over encrypted data in a controlled manner. The trend on this topic is to try to build schemes that are as expressive possible, i.e., functional encryption that supports any circuit evaluation. These results are at the cost of efficiency and security. They rely on recent, not very well studied assumptions, and no construction is close to being practical.

The goal of this thesis is to attack this challenge from a different angle: we try to build the most expressive functional encryption scheme we can get from standard assumption, while keeping the constructions simple and efficient.

To this end, we introduce the notion of functional encryption for inner-product evaluations, where plaintexts are vectors $\vec{x}$, and the trusted authority delivers keys for vectors $\vec{y}$ that allow the evaluation of the inner-product $\langle \vec{x}, \vec{y} \rangle$. This functionality already offers some direct applications, and it can also be used for theoretical constructions, as inner-product is a widely used operation.

Finally, we present two generic frameworks to construct inner-product functional encryption schemes, as well as some concrete instantiations whose security relies on standard assumptions. We also compare their pros and cons.

Mots Clés

cryptographie, chiffrement fonctionnel, produit scalaire, sécurité prouvée, constructions génériques, *projective hash functions*.

Keywords

cryptography, functional encryption, inner-product, provable security, generic constructions, projective hash functions.