HAL open science

Diverse modules and zero-knowledge

Fabrice Benhamouda

To cite this version:

Fabrice Benhamouda. Diverse modules and zero-knowledge. Cryptography and Security [cs.CR]. PSL Research University; ENS Paris, 2016. English. NNT : . tel-01399476v1

HAL Id: tel-01399476

https://inria.hal.science/tel-01399476v1

Submitted on 18 Nov 2016 (v1), last revised 20 Mar 2017 (v2)

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

Diverse modules and zero-knowledge

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité Informatique

COMPOSITION DU JURY

M. FERREIRA ABDALLA Michel
École normale supérieure
Directeur de thèse

M. POINTCHEVAL David
École normale supérieure
Directeur de thèse

M. KILTZ Eike
Ruhr-Universität Bochum
Rapporteur

M. SHOUP Victor
New York University
Rapporteur

M. HOFHEINZ Dennis
Karlsruher Institut für Technologie
Membre du jury

M. JOUX Antoine
Université Pierre et Marie Curie
Membre du jury

M. REYZIN Leonid
Boston University
Membre du jury

Soutenue par **Fabrice**
BEN HAMOUDA--GUICHOUX
le 1er juillet 2016

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

Diverse modules and zero-knowledge

Fabrice Ben Hamouda--Guichoux

Thèse de doctorat dirigée par
Michel Ferreira Abdalla et David Pointcheval

Abstract

Smooth (or universal) projective hash functions were first introduced by Cramer and Shoup, at Eurocrypt'02, as a tool to construct efficient encryption schemes, indistinguishable under chosen-ciphertext attacks. Since then, they have found many other applications, including password-authenticated key exchange, oblivious transfer, blind signatures, and zero-knowledge arguments. They can be seen as implicit proofs of membership for certain languages. An important question is to characterize which languages they can handle.

In this thesis, we make a step forward towards this goal, by introducing diverse modules. A diverse module is a representation of a language, as a submodule of a larger module, where a module is essentially a vector space over a ring. Any diverse module directly yields a smooth projective hash function for the corresponding language, and almost all the known smooth projective hash functions are constructed this way.

Diverse modules are also valuable in their own right. Thanks to their algebraic structural properties, we show that they can be easily combined to provide new applications related to zero-knowledge notions, such as implicit zero-knowledge arguments (a lightweight alternative to non-interactive zero-knowledge arguments), and very efficient one-time simulation-sound (quasi-adaptive) non-interactive zero-knowledge arguments for linear languages over cyclic groups.

Résumé

Les *smooth* (ou *universal*) *projective hash functions* ont été introduites par Cramer et Shoup, à Eurocrypt'02, comme un outil pour construire des schémas de chiffrement efficaces et sûrs contre les attaques à chiffrés choisis. Depuis, elles ont trouvé de nombreuses applications, notamment pour la construction de schémas d'authentification par mot de passe, d'*oblivious transfer*, de signatures en blanc, et de preuves à divulgation nulle de connaissance. Elles peuvent êtres vues comme des preuves implicites d'appartenance à certains langages. Un problème important est de caractériser pour quels langages de telles fonctions existent.

Dans cette thèse, nous avançons dans la résolution de ce problème en proposant la notion de *diverse modules*. Un *diverse module* est une représentation d'un langage, comme un sous-module d'un module plus grand, un module étant un espace vectoriel sur un anneau. À n'importe quel *diverse module* est associée une *smooth projective hash function* pour le même langage. Par ailleurs, presque toutes les *smooth projective hash functions* actuelles sont construites de cette manière.

Mais les *diverse modules* sont aussi intéressants en eux-mêmes. Grâce à leur structure algébrique, nous montrons qu'ils peuvent facilement être combinés pour permettre de nouvelles applications, comme les preuves implicites à divulgation nulle de connaissance (une alternative légère aux preuves non-interactives à divulgation nulle de connaissance), ainsi que des preuves non-interactives à divulgation nulle de connaissance et *one-time simulation-sound* très efficaces pour les langages linéaires sur les groupes cycliques.

Acknowledgments

Écrire des remerciements de thèse est un exercice périlleux et je souhaite d'avance m'excuser de ne pas mentionner et remercier explicitement toutes les personnes qui m'ont aidé et encouragé à un moment ou un autre de ma vie et qui par leurs paroles, leurs conseils ou leurs critiques, ont éclairé mes réflexions et m'ont permis de progresser.

Je tiens tout d'abord à adresser mes plus vifs remerciements à mes deux directeurs de thèse Michel Abdalla et David Pointcheval qui m'ont incité à étudier la cryptographie, m'ont guidé pendant mon stage de M2 et ma thèse, m'ont offert de nombreuses opportunités de recherche et de collaboration, et ont consciencieusement relu ce manuscrit.

I would like to thank my two reviewers Eike Kiltz and Victor Shoup who read and commented very carefully my thesis. I know this is a long (and tedious) work and I am grateful for it. I am also thankful to the other committee members : Dennis Hofheinz, Antoine Joux, and Leonid Reyzin. Je tiens également à témoigner toute ma gratitude à Geoffroy Couteau et Michele Minelli pour leur relecture attentive de cette thèse.

It was my great pleasure to collaborate with many researchers in cryptography, during various visits and internships: Dan Page and Elisabeth Oswald for a very nice first research experience in cryptography, during my M1 internship on side-channel attacks and residue number systems at Bristol University (I thank David Naccache for recommending me to do an internship in the Bristol Cryptography Group); Marc Joye whom I visited twice and who always helped me a lot in difficult situations; Benoît Libert and his innumerable patents; Mihir Bellare for spending a lot of time discussing with me during my one-week visit at San Diego; Dan Boneh and Vinod Vaikuntanathan for their insightful discussions; Stephan Krenn and Jan Camenisch during my visit at IBM Zürich; Léo Ducas and Ronald Cramer for a very productive one-week visit at CWI; and the Cryptography Research Group of IBM T.J. Watson, in particular Tal Rabin who warmly welcomed me for three months and who offered me a postdoc position, but also Gilad Asharov, Craig Gentry, Shai Halevi, Charanjit Jutla, Hugo Krawczyk, and Mor Weiss.

I would also like to thank all my co-authors: Michel Abdalla, Fabien Allard, Antoine Amarilli, Michel Banâtre, Sonia Belaïd, Olivier Blazy, Florian Bourse, Jan Camenisch, Céline Chevalier, Paul Couderc, Geoffroy Couteau, Houda Ferradi, Rémi Géraud, Javier Herranz, Marc Joye, Stephan Krenn, Tancrède Lepoint, Benoît Libert, Helger Lipmaa, Vadim Lyubashevsky, Phil MacKenzie, Claire Mathieu, Robin Morisset, David Naccache, Gregory Neven, Alain Passelègue, Kenny Paterson, Krzysztof Pietrzak, David Pointcheval, Emmanuel Prouff, Pablo Rauzy, Adrian Thillard, Jean-François Verdonck, Damien Vergnaud, Hoeteck Wee, and Hang Zhou.

Je voudrais également remercier tous mes collègues de l'équipe Crypto de l'ENS, avec qui j'ai aussi collaboré, discuté et passé des moments très agréables : Michel Abdalla, Nuttapong Attrapadung, Sonia Belaïd, Olivier Blazy, Raphaël Bost, Florian Bourse, Yuanmi Chen, Céline Chevalier, Jérémy Chotard, Mario Cornejo, Geoffroy Couteau, Angelo De Caro, Rafaël Del Pino, Itai Dinur, Léo Ducas, Aurélien Dupin, Pierre-Alain Dupont, Pooya Farshim, Houda Ferradi, Pierre-Alain Fouque, Georg Fuchsbauer, Romain Gay, Rémi Géraud, Dahmun

Goudarzi, Aurore Guillevic, Duong Hieu Phan, Sorina Ionica, Louiza Khati, Tancrède Lepoint, Baptiste Louf, Vadim Lyubashevsky, Pierrick Méaux, Thierry Mefenza, Michele Minelli, David Naccache, Phong Nguyen, Anca Nitulescu, Alain Passelègue, Miriam Paiola, Thomas Peters, David Pointcheval, Thomas Prest, Liz Quaglia, Răzvan Roşie, Sylvain Ruhault, Mario Strefler, Adrian Thillard, Mehdi Tibouchi, Damien Vergnaud, and Hoeteck Wee. Je tiens également à exprimer toute ma reconnaissance à l'équipe administrative du DI et au SPI, notamment Jacques Beigbeder, Lise-Marie Bivard, Isabelle Delais, Nathalie Gaudechoux, Joëlle Isnard, Valérie Mongiat et Ludovic Ricardou.

J'adresse mes sincères remerciements à la fondation CFM pour sa bourse de thèse généreuse et à ceux qui m'ont soutenu pour l'obtenir : mes deux directeurs de thèse, Ahmed Bouajjani, directeur adjoint de l'ED386, et Hubert Comon.

Je remercie tous mes amis, très spécialement Vaïa Machairas qui est toujours restée très proche malgré la distance et tous les informaticiens de la promotion 2009 de l'ENS, en particulier Antoine Amarilli, Yoann Bourse, Floriane Dardard, Marc Jeanmougin, Robin Morisset, Ludovic Patey et Pablo Rauzy. I would also like to thank Marty Simmons and Alan Green for their friendship and their warm welcome during my two visits to the US.

Je témoigne toute ma reconnaissance à Michel Bourdais, mon professeur de maths de sixième-cinquième qui a éveillé mon vif intérêt pour les mathématiques. Je tiens aussi à remercier les professeurs de physique Sophie Larasse et Pascal Brasselet, pour m'avoir ouvert la voie de la recherche et m'avoir permis de présenter mes premiers exposés en public lors des Olympiades de Physique sur un écoulement pas si simple, avec Philippe-Henri Blais et Xavier Le Gall.

J'aimerais également remercier de tout mon cœur mes parents qui m'ont toujours aimé, soutenu et tout fait pour me permettre de m'épanouir. Merci aussi à ma sœur Caroline et mon frère Matthieu.

Enfin, je souhaiterais tout particulièrement remercier Hang Zhou qui me rend heureux en partageant ma vie depuis plus de cinq ans.

Preface

The main goal of this thesis is to show the power of smooth (or universal) projective hash functions and promote their use in many cryptographic protocols. It is mostly aimed at cryptographers with some knowledge in protocol design. I hope they will find new tools or at least a different viewpoint on some tools which they can apply to new problems. Nevertheless, this thesis should be accessible to any person with a computer science background, despite not including a general introduction to cryptography or its history. I rather tried to write it both as a tutorial for diverse modules and as a reference thereof. This thesis is therefore definitely not a collection of papers, but a major rewriting of some of my papers and also includes a significant portion of unpublished work. In particular, it only covers a small fraction of my work during my PhD thesis, to keep it neat, coherent, and relatively easy to grasp. Furthermore, I have decided to use a more abstract approach and deliberately tried to limit awfully complex notation using coordinates.

Diverse modules summarize my vision of smooth projective hash functions, which are a powerful tool introduced by Cramer and Shoup in [CS02]. They can be seen as a particular case of diverse groups, the main (and essentially only known) way to construct smooth projective hash functions, which were also introduced by Cramer and Shoup in the same paper. What makes them so powerful, at least from my point of view, is that diverse modules have additional structural properties compared to groups. Thanks to these structural properties, we can use a more algebraic viewpoint and more algebraic constructions.

This search for structure and use of algebraic methods is actually a feature in most of my thesis and even before, starting with my work with Michel Abdalla and David Pointcheval on tight forward-secure signatures [ABP13] where I mainly analyzed the structure of residues modulo composite numbers. My work with Michel Abdalla, Alain Passelègue, and Kenny Paterson on pseudorandom functions [ABPP14; ABP15a; ABP15b] culminated in a completely algebraic framework, which shows the equivalence of some security definition with the linear independence of some polynomials. And finally, even more recently, my work on the randomness complexity of private circuits with Sonia Belaïd, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud [BBP+16] contains in its heart an algebraic characterization of security based again on linear algebra.

Contents

Abstract iii

Résumé v

Acknowledgments vii

Preface ix

1 Introduction 1
- 1.1 Proofs in Cryptography . . . 2
  - 1.1.1 Provable Security and Mathematical Proofs . . . 2
  - 1.1.2 Arguments and Proofs . . . 2
- 1.2 Projective Hash Functions and Applications . . . 4
  - 1.2.1 Projective Hash Functions . . . 4
  - 1.2.2 Applications . . . 5
  - 1.2.3 Languages . . . 6
- 1.3 Our Results . . . 7
  - 1.3.1 Diverse Modules and Diverse Vector Spaces . . . 7
  - 1.3.2 Operations on Diverse Modules . . . 7
  - 1.3.3 Applications Related to Zero-knowledge . . . 8
  - 1.3.4 Associated Personal Publications . . . 9
  - 1.3.5 Organization . . . 9
- 1.4 Our Other Contributions . . . 9
  - 1.4.1 Other Contributions on Projective Hash Functions . . . 10
  - 1.4.2 Pseudorandom Functions . . . 11
  - 1.4.3 Randomness Complexity of Private Circuits . . . 12
  - 1.4.4 Cryptosystems Based on Residue Symbols . . . 13
  - 1.4.5 Lattice-Based Zero-Knowledge Arguments . . . 13
  - 1.4.6 Forward-Secure Signature Schemes . . . 13
  - 1.4.7 Security Proof of J-PAKE . . . 13
  - 1.4.8 Coppersmith Methods and Analytic Combinatorics . . . 14
- Personal Publications . . . 14
  - Journal Papers . . . 14
  - Conference Papers . . . 14
  - Manuscripts . . . 16
  - Patent Applications . . . 16

2 Preliminaries 17
- 2.1 Notation and Preliminaries . . . 18
  - 2.1.1 General Notation . . . 18
  - 2.1.2 Preliminaries on Provable Security . . . 20

2.1.3 Statistical and Computational Indistinguishability . . . 22
2.1.4 Proof by Games or Hybrid Arguments . . . 26
2.1.5 Cyclic Groups, Bilinear Groups, and Multilinear Groups . . . 26
2.2 Cryptographic Primitives . . . 28
2.2.1 Collision-Resistant Hash Function Families . . . 28
2.2.2 Encryption . . . 29
2.2.3 Randomness Extractors and Min Entropy . . . 34
2.3 Languages . . . 35
2.3.1 Languages for Projective Hash Functions . . . 36
2.3.2 Hard-Subset-Membership Languages . . . 37
2.3.3 Language for Zero-Knowledge Arguments . . . 39
2.4 Zero-Knowledge Arguments . . . 39
2.4.1 Overview . . . 39
2.4.2 Formal Definitions of Zero-Knowledge Arguments . . . 42
2.4.3 Non-Interactive Zero-Knowledge Arguments . . . 47
2.5 Projective Hash Functions . . . 48
2.5.1 Projective Hash Functions (PHFs) . . . 48
2.5.2 Smooth Projective Hash Functions (SPHFs) . . . 49
2.5.3 Simple Applications of SPHFs . . . 50

3 Diverse Vector Spaces **55**
3.1 First Examples, Definition, and Link with SPHFs . . . 56
3.1.1 Step-by-Step Overview . . . 56
3.1.2 Definition . . . 62
3.2 Conjunctions and Disjunctions . . . 68
3.2.1 Conjunctions . . . 68
3.2.2 Disjunctions . . . 69
3.3 Application to Non-Interactive Zero-Knowledge Arguments . . . 76
3.3.1 Overview . . . 76
3.3.2 Construction . . . 78
3.3.3 Completeness and Security . . . 79
3.4 More Examples . . . 80
3.4.1 Matrix Decisional Diffie-Hellman Assumptions (MDDH) . . . 80
3.4.2 Cramer-Shoup Encryption . . . 82
3.4.3 Encryption of Plaintexts Satisfying a System of Quadratic Equations . 83

4 Diverse Modules **85**
4.1 Universality and Smoothness . . . 86
4.1.1 Motivation . . . 86
4.1.2 Universal Projective Hash Functions . . . 86
4.1.3 Weakly Universal Projective Hash Functions . . . 87
4.2 Diverse Modules (DMs) . . . 88
4.2.1 Graded Rings . . . 89
4.2.2 Diverse Modules, Universal PHFs, and Tools for Composite Order . . 92
4.2.3 Link with Diverse Groups . . . 102
4.3 Conjunctions and Disjunctions . . . 103
4.3.1 Conjunctions . . . 103

4.3.2 Disjunctions . . . 103
4.4 t-Universality, t-Smoothness, and t-Soundness . . . 111
4.4.1 t-Universality and t-Smoothness . . . 112
4.4.2 t-Soundness . . . 114
4.4.3 Construction of t-Sound Tag-CS-DMs and Tag-CS-DVSs . . . 115

5 Pseudorandomness **123**
5.1 Pseudorandom Projective Hash Functions and Diverse Vector Spaces . . . 124
5.1.1 Definition . . . 124
5.1.2 Construction from Hard-Subset-Membership Languages . . . 124
5.1.3 Construction from MDDH . . . 126
5.2 Mixed Pseudorandomness . . . 129
5.2.1 Definition . . . 129
5.2.2 GL Disjunctions of a GL-DVS and a Pr-DVS . . . 130
5.2.3 CS/KV Disjunctions of a DVS and a Pr-DVS . . . 133

6 Applications of Diverse Modules **137**
6.1 Honest-Verifier Zero-Knowledge Arguments . . . 138
6.1.1 Two Dual Constructions From DMs . . . 138
6.1.2 Extensions and Comparisons . . . 141
6.2 Non-Interactive Zero-Knowledge Arguments (NIZK) . . . 145
6.2.1 First Constructions . . . 145
6.2.2 t-Time Simulation-Soundness . . . 147
6.2.3 Concrete Instantiation and Comparison . . . 152
6.2.4 Application: Threshold Cramer-Shoup-like Encryption Scheme . . . 153
6.3 Trapdoor Smooth Projective Hashing and Implicit Zero-Knowledge . . . 160
6.3.1 Overview . . . 160
6.3.2 Trapdoor Smooth Projective Hash Functions (TSPHFs) . . . 164
6.3.3 Implicit Zero-Knowledge Arguments (iZK) . . . 170

7 Conclusion and Open Questions **177**
7.1 Conclusion . . . 177
7.2 Open Questions . . . 178

Notation **181**

Abbreviations **183**

List of Illustrations **185**
Figures . . . 185
Tables . . . 185

Bibliography **187**

Chapter 1

Introduction

For a long time, the main purpose of cryptography was to allow two people who agreed with each other on a secret bit string, called a *secret key*, to *confidentially* communicate, using *(secret-key) encryption*.

Since the breakthrough discoveries of *key exchange* [DH76], *public-key encryption*, and *digital signatures* [RSA78], the scope of cryptography has been vastly broadened. Key exchange and public key encryption enable to confidentially talk to any person or entity (like websites), even without knowing him or it beforehand, while digital signatures provide authentication, or in other words, enable to ensure that a message comes from the right user or entity. Contrary to written signatures, digital signatures cannot be forged and can be easily checked by a computer.

Nowadays, cryptography is omnipresent in our day-to-day life, although often very hidden: credit cards, access badges, online shopping, biometric passports, smartphones, emails, and even access to most websites (following the trend of HTTPS everywhere) all utilize cryptographic protocols. Most of these uses however only rely on encryption, key exchange, and authentication via digital signatures.

But cryptography is even broader than these three important practical primitives. In this thesis, we consider cryptographic protocols (or cryptosystems), around the counter-intuitive concept of *zero-knowledge*. Zero-knowledge proofs enable a user, Alice, to prove to another user, Bob, that some statement (for example Goldbach's conjecture, a famous open problem in mathematics) is true without revealing anything else. After running the corresponding protocol, Bob will only be convinced that the statement is true but will have learned nothing else. Although such protocols are still not mainstream, they are starting to be deployed in the real world, as for electronic voting in [Helios].

The main objects of study in this thesis, namely *diverse modules* and *smooth (or universal) projective hash functions*, can be considered even more intriguing: they enable to prove some fact implicitly, still with a form of zero-knowledge. Concretely, using smooth projective hash functions, Bob could send to Alice a treasure map in such a way that Alice could read it, only if she knows a proof of Goldbach's conjecture. But Bob would have no idea whether Alice was able to read the treasure map (and find the treasure) or not.[1] Diverse modules can be seen as an algebraic framework for projective hash functions. In this thesis, we introduce the notion of diverse modules, show many tools to manipulate them, and present various applications related to zero-knowledge.

[1] Except by physically going to the place where the treasure is buried.

Towards this goal, we first introduce in more detail the notions of proofs in cryptography, projective hash functions, and diverse modules. We then sketch the main ideas and concepts in this thesis. Finally, we briefly summarize our other contributions that were not included in this thesis.

1.1 Proofs in Cryptography

In modern cryptography, proofs play an important dual role. Not only are they at the heart of provable security, but they are also both an important subject of study of cryptography and an important tool to construct advanced cryptosystems or cryptographic protocols.

1.1.1 Provable Security and Mathematical Proofs

In [GM82], Goldwasser and Micali laid the foundations of provable security. A cryptosystem or cryptographic protocol is provably secure if any adversary breaking the security of the system can be transformed into an adversary breaking a problem which is supposed to be hard. This transformation is called a reduction. Not only does this approach enable to rule out many insecure protocols, but it also enables to focus cryptanalysis, the research of attacks against cryptosystems, on a small number of simple well defined problems. These problems include computing the factorization and discrete logarithms (inverse of exponentiation) in some specific cyclic groups, together with associated weaker problems.

More generally, in mathematics and theoretical computer science, the main goal is to find new results and to mathematically prove them. As such, mathematical proofs are an important tool.

1.1.2 Arguments and Proofs

P versus NP. But proofs are not only a tool but an object of study in theoretical computer science. One of the most important open problems in computer science is actually the problem of P versus NP. It basically asks whether being able to efficiently check a proof of a statement, is equivalent to being able to efficiently check if a statement is true or false. It is one of the seven Millennium Prize Problems of the Clay Mathematics Institute.

More formally, we define a language $\mathscr{L}$ to be a set of bit strings. Bit strings can be used to encode almost anything, including mathematical statements. We can for example consider the language of the mathematical statements which are provable in the Zermelo–Fraenkel set theory.

On the one hand, the class P is the class of languages $\mathscr{L}$, such that there exists an algorithm that takes as input a bit string x and that can decide in polynomial time (in the size of x), whether $x \in \mathscr{L}$. We generally consider this class as the class of easy-to-decide languages and call polynomial-time algorithms, efficient algorithms.

On the other hand, the class NP is the class of languages $\mathscr{L}$, such that there exists an algorithm, that takes as input two bit strings x and w and that can decide in polynomial time (in the size of x), whether w is a valid proof or witness that $x \in \mathscr{L}$. We suppose that for any word $x \in \mathscr{L}$, there exists such a witness w, while otherwise no such witness exists. For example, the language of provable mathematical statements (in the Zermelo-Fraenkel set theory) with polynomial-size proofs is in NP. But this language is not known to be in P. If it were, there would be no reason to give proofs of mathematical statements as finding a

proof would be roughly as efficient as checking it! More generally, while the class P is clearly included in NP, finding whether NP is included in P is a major open question, but most researchers strongly believe that $\mathrm{P} \subsetneq \mathrm{NP}$.

Interactive proofs. We can see an NP proof or witness as a mathematical proof, or equivalently as a lecture, where the teacher (or prover) shows a proof of the statement "$x \in \mathscr{L}$" to the students (or verifiers) who check the proof without asking any question. But as we know, lectures are often interactive and this interactivity helps the students to better understand the course.[2]

In two independent seminal papers, that won a Gödel prize, Babai [Bab85] and Goldwasser, Micali, and Rackoff [GMR85] introduced the notion of interactive proofs or Arthur-Merlin games.[3] In an interactive proof, an all-powerful prover tries to convince a polynomial-time verifier that $x \in \mathscr{L}$, by interacting with him, through a protocol. If $x \in \mathscr{L}$, the prover should have a strategy to convince systematically the verifier (*completeness* property), while otherwise, the prover should not be able to convince the verifier with probability more than $1/2$ (*soundness* property). The verifier is indeed probabilistic: he is allowed to draw random coins and act depending on them. Otherwise, the class of languages handled by interactive proofs, denoted by IP, would just be the class P. On the contrary, in [Sha92], Shamir showed that IP is equal to the class PSPACE, which corresponds to algorithms that use a polynomial amount of space to store intermediate variables. In particular, the class PSPACE contains the class NP.

Zero-knowledge. As pointed out by Goldwasser, Micali, and Rackoff in their seminal paper [GMR85], an important question about interactive proofs in cryptography is whether the prover reveals more information (or knowledge) to the verifier than the fact that $x \in \mathscr{L}$. Indeed, in cryptography, we often want to hide information. A proof that does not reveal any information to the verifier besides the membership of the word to the language is called zero-knowledge.

Using a zero-knowledge proof, a person, called Alice, could for example prove that the Goldbach's conjecture is true (or false) to another person, called Bob. Bob would be sure that the conjecture is true, but would have absolutely no idea of how to mathematically prove it, and would be unable to claim the discovery of the proof before Alice.

This concept might seem very counter-intuitive and impossible to achieve. Yet in [GMW91], Goldreich, Micali, and Wigderson constructed zero-knowledge proofs for any language in NP, under a very weak assumption, namely the existence of one-way functions. One-way functions are functions that can efficiently be evaluated on any input but that are hard to invert. This assumption is necessary to prove the existence of almost any cryptosystem (at least from a theoretical point of view).[4]

Furthermore, in this construction, the prover runs in polynomial time if he is given a witness w for the word x. Concretely, in the example of the Goldbach's conjecture, this means

[2] This way of presenting interactive proofs is inspired by Micali's talk "Proofs, Secrets, and Computation", given on Tuesday 26 May 2015, at Paris 6, France.

[3] There is a subtle difference between the two: in Arthur-Merlin games, the random coins used by the verifier are public, while in interactive proofs, they are private. However, in [GS86], Goldwasser and Sipser showed that they are basically equivalent, up to a loss of two rounds of interaction.

[4] We remark that the existence of one-way function implies that P is distinct from NP.

that if Alice has a mathematical proof of this conjecture, she can easily do a zero-knowledge proof to Bob that the conjecture is true.

In this thesis, we always assume that the prover runs in polynomial time, when given access to a witness. The efficiency of the prover is indeed important to be able to use the resulting proof in real-world cryptographic protocols.

Applications. Zero-knowledge proofs have many applications in cryptography. They enable to ensure that parties in a cryptosystem behaved honestly, and did not cheat during the protocol. Most complex cryptographic systems from multi-party computation (where several users, each holding a different input, want to compute a function of all these inputs) to electronic voting (e-voting) or electronic cash [Cha82] (e-cash) often use forms of zero-knowledge proofs or variants thereof.

Variants. We consider many variants of zero-knowledge proofs throughout this thesis.

Arguments. Zero-knowledge arguments are similar to zero-knowledge proofs, except that soundness just holds against polynomial-time provers. It often enables to construct much more efficient schemes. Moreover, arguments can replace proofs in most cryptosystems.

Common reference string. In this thesis, we often consider the *common reference string* (CRS) model, in which the prover and the verifier both have access to a common bit string chosen by some trusted party. In practice, such a bit string can be generated by a multi-party computation between users who are believed not to collude. For example, for e-voting, all the candidates and a few random citizens could generate together the CRS.

One major advantage of the CRS model is it enables to construct non-interactive zero-knowledge proofs and arguments, which are impossible without CRS. In a non-interactive proof of argument, the prover just sends one message (called the proof) to the verifier, which can verify it (using the CRS). This proof is similar to a witness of an NP language, except that sending a witness often gives too much knowledge to the verifier.

Honest-verifier zero-knowledge. Honest-verifier zero-knowledge arguments or proofs are similar to zero-knowledge arguments or proofs, except that we only assume that the verifier learns nothing if he behaves honestly and follows the protocol. This relaxation enables to construct even more efficient schemes.

1.2 Projective Hash Functions and Applications

1.2.1 Projective Hash Functions

Smooth (or universal) projective hash functions, also called hash proof systems, were introduced by Cramer and Shoup in their seminal paper [CS02]. They could be considered even more intriguing than zero-knowledge proofs, as they not only resemble honest-verifier zero-knowledge proofs but also are not explicit: the verifier does not necessarily know whether the proof is valid or not.

Similarly to zero-knowledge proofs with efficient provers, smooth projective hash functions are defined for an NP language $\mathscr{L}$. The verifier can generate a (secret) bit string hk called

a *hashing key*. This hashing key defines a function which associates any possible word χ (inside or outside the language $\mathscr{L}$) to some bit string H, called the *hash value* of the word χ. This function can be efficiently evaluated by anyone knowing the hashing key. Furthermore, the verifier can derive from the hashing key, a *projection key* hp, which basically defines the previous function only on $\mathscr{L}$. The verifier can then send the projection key to the prover.

From this projection key hp and a witness w for a word $\chi \in \mathscr{L}$, the prover can efficiently compute a *projected hash value* pH which is equal to the hash value H computed by the verifier. However, if $\chi \notin \mathscr{L}$, the prover cannot guess the hash value H computed by the verifier. This is called the *smoothness* or *universal* property.

Being able to compute the hash value or projected hash value for some word χ given only the projection key hp can therefore be seen as a proof that $\chi \in \mathscr{L}$. Moreover, an honest verifier seeing such a projected hash value computed by a prover learns nothing as he could have computed this value himself using the hashing key.

1.2.2 Applications

Smooth (or universal) projective hash functions have many applications.

Honest-verifier zero-knowledge proofs. A first application is the construction of two-round honest-verifier zero-knowledge proofs: after having received a projection key from the verifier, the prover just sends back the projected hash value pH, and the verifier checks that it corresponds to the computed hash value H. It offers an (often more efficient) alternative to Sigma-protocols [Cra97; CDS94], a classical efficient way of constructing honest-verifier zero-knowledge proofs. This construction has been used for example to construct round-optimal efficient blind signatures in [Cha82; BPV12; BBC+13c]. A blind signature scheme enables Alice to make Bob sign a message without revealing it. Bob still controls the number of messages signed for Alice, and Alice cannot sign a document she has not asked Bob to sign. But Bob does not know which messages he is signing for Alice. Blind signatures are in particular useful to build electronic cash [Cha82], as they allow the bank to blindly sign electronic coins.

Use of the implicitness feature. The implicitness feature of smooth projective hash functions allows for a broader range of applications. For example, the verifier can use the hash value as a key to encrypt a message to the prover, in such a way that the prover can only decrypt the message if he knows a witness for the word. But the verifier does not learn whether the prover has a witness, and whether he was able to read the message. We actually show in Section 2.5.3.3, that a variant of this protocol can be used to securely send a message to a secret agent without the secret agent having to reveal that he is a secret agent. The sender will never know if he was talking to a secret agent, but he will be sure that if he was not, his interlocutor would not be able to read the message.

Password-authenticated key exchange. An important application is password-authenticated key exchange. Such a protocol enables two users, Alice and Bob, to derive a fresh random secret bit string (called a secret key) just from a common password. Contrary to classical key exchanges (used for example by the HTTPS protocol on the web), the authentication means, namely the common password, might be guessed by an adversary. Password-authenticated key exchanges should therefore ensure that the only way for an

adversary to check if a password guess is valid consists in interacting with Alice and Bob. In particular, it should be impossible for the adversary to look once at an execution of the protocol between Alice and Bob, and then to find out the password using his own computer by testing each possible password, without any interaction with Alice nor Bob.

Because of this strong property, classical zero-knowledge proofs often cannot be used to construct password-authenticated key exchange schemes, as an adversary could check whether the proof corresponds to a given password. The implicitness feature of projective hash functions solves this issue and enables to construct efficient password-authenticated key exchange in the CRS model [KOY09; GL06; JG04; GK10].

Oblivious transfer. Another important application of smooth projective hash functions is the construction of oblivious transfer protocols [Rab81]. Intuitively, such protocols enable a client Alice to ask a question to a server Bob and get the answer without Bob learning what the question was. But Bob is still sure that Alice learned at most one answer to a single question.

Encryption secure against chosen-ciphertext attacks. Finally, smooth projective hash functions enable the construction of very efficient (public-key) encryption schemes, secure in a strong model, called indistinguishability under chosen-ciphertext attacks. Intuitively, in this model, the encryption scheme remains secure even if the adversary has access to a key enabling the encryption (called an encryption key or public key), and even if it is allowed to ask the decryption of messages (called plaintext) of its choice. This application was actually the main purpose of the introduction of projective hashing by Cramer and Shoup in [CS02].

1.2.3 Languages

For the original application of smooth projective hash functions to encryption schemes, Cramer and Shoup only needed projective hash functions for languages directly corresponding to the cryptographic assumption on which they were relying, such as *decisional Diffie-Hellman* (DDH), *quadratic residuosity* (QR), or *decisional composite residuosity* (DCR). With the advent of many other applications, more complex languages begin being required, starting with the languages of encrypted versions (formally called ciphertexts) of a given message under some public encryption key.

Limitations. The first natural question is to know whether, like zero-knowledge proofs, smooth projective hash functions can be constructed for any NP language. Unfortunately, as they imply statistically sound witness encryption [GGSW13], this is not possible, unless the polynomial hierarchy (which is a hierarchy of class of languages which can be seen as variants and extensions of NP) collapses. Although it is not known whether the polynomial hierarchy collapses, most researchers believe this cannot happen. We should point out that if $\mathrm{P} = \mathrm{NP}$, then the polynomial hierarchy collapses, but there would also be no more theoretical cryptography as we currently know it.

Diverse groups. Despite these limitations, we now know how to construct smooth projective hash functions for a large variety of languages. Most of these constructions rely on *diverse groups*, a notion introduced by Cramer and Shoup in their original paper [CS02].

From a high level point of view, diverse groups are a way to represent a language as a subgroup of a larger group, together with a large enough group of homomorphisms from this larger group to the set of hash values. Languages represented by subgroups automatically have an associated smooth (or universal) projective hash function.

1.3 Our Results

1.3.1 Diverse Modules and Diverse Vector Spaces

In this thesis, we introduce and study a particular case of diverse groups: *diverse modules*. Diverse modules are a way to represent a language as a submodule of a larger module. We recall that a module is a generalization of a vector space over a ring. In particular, a module is a group and a submodule is a subgroup.

The additional structure of diverse modules compared to diverse groups provides two main advantages. First, it removes the requirement to explicitly construct a large enough set of homomorphisms, as such a set can be canonically constructed (as the set of linear maps for the module to a free module over the base ring). This therefore simplifies the construction and the description of diverse modules compared to diverse groups. Second, it enables to combine generically diverse modules and to algebraically add new useful properties to the associated projective hash function.

An important particular case of diverse modules is when the underlying ring is a finite field of prime order. Such a diverse module is called a *diverse vector space*. Diverse vector spaces were actually already presented by Cramer and Shoup [CS02] as an important case of diverse groups. In this thesis, however, we go further and provide many tools for diverse vector spaces.

1.3.2 Operations on Diverse Modules

While the main application of diverse modules and diverse vector spaces is to construct smooth projective hash functions, we show in this thesis that working with diverse modules provides many benefits over working directly with smooth projective hash functions.

One of them is that we can generically combine and enhance diverse modules. Given two diverse modules for two languages $\mathscr{L}_1$ and $\mathscr{L}_2$, we can generically construct diverse modules for the conjunction and the disjunction of these two languages. The *conjunction* of $\mathscr{L}_1$ and $\mathscr{L}_2$ is the language of pairs (x_1, x_2) such that both $x_1 \in \mathscr{L}_1$ *and* $x_2 \in \mathscr{L}_2$, while the *disjunction* of $\mathscr{L}_1$ and $\mathscr{L}_2$ is the language of pairs (x_1, x_2) such that $x_1 \in \mathscr{L}_1$ *or* $x_2 \in \mathscr{L}_2$.

We point out that Abdalla, Chevalier, and Pointcheval already showed in [ACP09] how to generically construct a smooth projective hash function for the conjunction and the disjunction of two languages for which they know smooth projective hash functions. However, their construction is not algebraic and so cannot be combined with our other tools on diverse modules and cannot be used in some of our direct applications of diverse modules. Furthermore, their disjunction construction only works for a weak version of smooth projective hash functions, where the projection key is allowed to depend on the word of the language.

Another important tool for diverse modules is what we call t-sound extension. While smooth projective hash functions do not provide any security guarantee when the adversary is given access to even a single hash value of a word $x \notin \mathscr{L}$, projective hash functions based

on t-sound extensions of diverse modules are smooth even when given access to $t-1$ hash values (of $t-1$ words chosen by the adversary).

1.3.3 Applications Related to Zero-knowledge

Diverse modules enable to construct smooth projective hash functions, and therefore also all the previous cryptographic protocols described as applications of smooth projective hash functions. But diverse modules enable to go beyond that, thanks to their algebraic structure.

Non-interactive zero-knowledge arguments. One of the most interesting and counter-intuitive applications of diverse modules is the construction of non-interactive zero-knowledge arguments, with constant-size proofs. We have already seen that smooth projective hash functions can be used to construct honest-verifier zero-knowledge arguments, where the verifier sends a projection key hp and the prover sends back to the verifier the projected hash value pH of the word χ to be proven in the language $\mathscr{L}$. The problem of this protocol is that it is not zero-knowledge, as the verifier could maliciously generate the projection key hp in such a way that the projected hash value pH sent by the prover may leak some information about the witness used by the prover. Moreover, only the verifier (who generated the projection key hp from some hashing key hk) can verify the projected hash value pH.

Still, we show how to construct a non-interactive zero-knowledge proof, using a strong version of disjunctions of diverse modules, between the language in which we are interested, and another helper language used to add public verifiability and zero-knowledge. The resulting schemes work in any bilinear group and essentially correspond to the schemes of Jutla and Roy in [JR13; JR14], but with a different proof, that we believe to be more modular and simpler to understand. Furthermore contrary to the schemes of Jutla and Roy, our schemes can be extended to the composite-order setting, using diverse modules, and handle a stronger security notion, called one-time simulation-soundness, using 2-sound extension. One interesting application of our one-time simulation-sound non-interactive zero-knowledge argument is the construction of the most efficient threshold and structure-preserving encryption scheme indistinguishable under chosen-ciphertext attacks.

We point out that Kiltz and Wee later improved our constructions to base them on slightly weaker assumptions in [KW15].

Zero-knowledge variants of projective hash functions. As seen previously, smooth projective hash functions do not provide any guarantee when the projection key is maliciously generated. In many protocols, this means that they only ensure a form of honest-verifier zero-knowledge property.

Using disjunctions of diverse modules between the language we are interested in, and another helper language, we construct *trapdoor smooth projective hash functions* and *implicit zero-knowledge arguments* which can be seen as zero-knowledge variants of smooth projective hash functions. Implicit zero-knowledge arguments in particular are a lightweight alternative to non-interactive zero-knowledge arguments. They can be used to enforce semi-honest behavior in multi-party computation [GMW87], at a relatively low cost.

We point out that the use of disjunctions for these constructions can be seen as dual of their use for non-interactive zero-knowledge arguments. Details are given in Chapter 6.

1.3.4 Associated Personal Publications

As explained in the preface, this thesis is not a collection of papers. However, it is mainly based on three papers:

[BBC+13c] (a merge of [BBC+13b] and [BP13a]) in which we introduce diverse vector spaces (under the name of "generic framework for smooth projective hash functions") and trapdoor smooth projective hash functions;

[ABP15c] in which we construct a strong form of disjunction of diverse vector spaces using tensor products, and derive from it, constant-size non-interactive zero-knowledge arguments;

[BCPW15] in which we introduce and construct implicit zero-knowledge arguments.

These three papers contain other applications that we are only discussing in the next section, but not in the core of this thesis.

The formalization used in this thesis for diverse vector spaces and projective hash functions, while based on the previous papers, has been improved to be coherent amongst the variants of projective hash functions we consider. Moreover, all the work on diverse modules has yet not been published.

1.3.5 Organization

This thesis is organized in seven chapters.

Chapter 1 is the introduction.

Chapter 2 introduces notation and preliminaries to understand the thesis. Besides standard cryptographic preliminaries, we describe our formalization of languages, projective hash functions, and some simple applications.

Chapter 3 presents diverse vector spaces together with generic conjunctions and disjunctions, and an application to non-interactive zero-knowledge arguments. This chapter is both a warm-up for diverse modules and an important particular case thereof. It is therefore didactic and contains many examples to help the reader understand step-by-step the framework of diverse vector spaces.

Chapter 4 defines diverse modules and provides many tools for diverse modules. As such, it is the core of the thesis. It is also a very technical chapter. A very good understanding of Chapter 3 is highly recommended before reading Chapter 4.

Chapter 5 introduces the notion of pseudorandomness and mixed pseudorandomness which can be used to improve the efficiency of many applications based on diverse vector spaces.

Chapter 6 gives various applications of the techniques proposed in the three previous chapters: honest-verifier zero-knowledge arguments, non-interactive zero-knowledge arguments, trapdoor smooth projective hash functions, and implicit zero-knowledge arguments.

Chapter 7 shortly concludes the thesis and raises open questions.

1.4 Our Other Contributions

Besides what is exposed in this thesis, we worked on various other cryptographic subjects.

1.4.1 Other Contributions on Projective Hash Functions

We considered other applications of projective hash functions than the ones related to zero-knowledge and presented in this thesis.

Password-authenticated key exchange and oblivious transfer. We constructed password-authenticated key exchange and oblivious transfer schemes and variants thereof in various models, in different articles with Michel Abdalla, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud:

[BBC+13a] introduces the notion of language-authenticated key exchange, which is an extension of password-authenticated key exchange, where not only the two parties share a password, but they also hold a word in some language defined by this password. In particular, it encompasses the notion of secret-handshake [BDS+03], where two users, who belong to the same group (and who have a certificate proving so), want to establish a key;

[BBC+13c] constructs the first smooth projective hash function for Cramer-Shoup encryption scheme [CS98] in which the projection key does not depend on the word. It then uses this smooth projective hash function to construct the most efficient (at the time) one-round password-authenticated key exchange in the Bellare-Pointcheval-Rogaway model [BPR00] and in the universal composability framework [Can01; CHK+05] with static corruptions. The schemes are based on [KV11];

[ABB+13] introduces the notion of smooth-projective-hash-functions-friendly commitment schemes and uses it to construct oblivious transfer and password-authenticated key exchange in the universal composability framework with adaptive corruptions (which is stronger than static corruptions) assuming that the users can reliably erase data in their memory;

[ABP14] goes even further and introduces the notion of *explainable smooth projective hash functions* to construct oblivious transfer and password-authenticated key exchange schemes in the universal composability framework with adaptive corruptions without assuming the users can reliably erase data in their memory. The resulting schemes are much more efficient than the previously known schemes in the same model. The explainability property ensures that there is a way to exhibit a hashing key corresponding to any hash value of a given word $\chi \notin \mathscr{L}$, even after having published the projection key. This paper also provides the most efficient password-authenticated key exchange schemes in the universal composability framework with adaptive corruption assuming reliable erasures, in cyclic groups under the DDH assumption. Previous schemes were either very inefficient [ACP09] or required pairings or bilinear groups [ABB+13; JR15];

[BP13b] studies password-authenticated key exchange in which the server does not store the password of the user but only a somehow encoded version of it, such that if the database of the server leaks, finding all the passwords takes a longer time;

[ABP15d; ABP16] introduce the notion of encryption scheme indistinguishable under plaintext-checkable attacks. This is a weakening of the notion of indistinguishability under chosen-ciphertext attacks, which is sufficient for the constructions of password-authenticated key exchange schemes based on [KOY09; GL06; JG04; GK10]. This

weakening allows for more efficient constructions of encryption schemes. We believe that this new notion of encryption scheme might be useful in other contexts;

[ABP15c] also constructs the first one-round password-authenticated key exchange for a group of three users.[5]

We should point out that after our papers, in [JR15], Jutla and Roy constructed a password-authenticated key exchange scheme (in bilinear groups), which is more efficient that all our constructions in the universal composability framework, with static and adaptive corruptions.

Other applications. In [BBC+13c], we also constructed blind signatures.

In [BJL16; BJL13], with Benoît Libert and Marc Joye, we constructed a generic aggregator-oblivious encryption scheme with a relatively tight security reduction, which means that the security of the scheme is really close to the hardness of the underlying problem and therefore smaller parameters can be used compared to schemes with non-tight security reduction. An aggregator-oblivious encryption scheme enables a special user, called an aggregator, to compute the sum of the values of n other users $U_1, \ldots, U_n$. Each user U_i encrypts its input value x_i using his secret key. The aggregator can then combine the resulting messages and get the sum $\sum_{i=1}^{n} x_i$ using his own secret key. The aggregator should learn no more than the sum of all the values, even if he colludes with some users.

Our construction is based on key-homomorphic smooth projective hash functions which can be based on diverse modules (although this is not explicitly indicated in [BJL16]).

1.4.2 Pseudorandom Functions

In [ABPP14; ABP15a; ABP15b], with Michel Abdalla, Alain Passelègue, and Kenny Paterson, we studied pseudorandom functions and variants thereof. A pseudorandom function is a family of functions (F_K) indexed by a bit string K called a key such that, when the key K is chosen uniformly at random, F_K looks indistinguishable from a random function. We also consider the security of pseudorandom functions under related-key attacks, in which the adversary can also get access to $F_{\phi(K)}$ for functions ϕ of its choice (in some set Φ). Moreover, we construct multilinear and aggregate pseudorandom functions [CGV15]. The first ones are similar to classical pseudorandom functions, except that F_K looks indistinguishable from a random multilinear function, while the second ones enable to efficiently evaluate an aggregate (like a product) of the outputs of F_K on a potentially exponential number of inputs.

Our main result is the introduction of a framework, called "polynomial linear pseudorandomness security" in [ABP15b]. This framework can be used to very easily derive constructions for all these schemes, under a classical assumption, namely the decisional d-Diffie-Hellman inversion.

To present this framework, let us first recall the bracket notation introduced in [EHK+13]: if $\mathbb{G}$ is a multiplicative cyclic group of prime order p generated by g, for any scalar $x \in \mathbb{Z}_p$, we define $[x]$ to be the group element g^x, where $\mathbb{Z}_p$ is the finite field of order p. Then, we essentially show the following theorem: under the decisional d-Diffie-Hellman inversion assumption, for any positive integer q, for any multivariate polynomials $P_1, \ldots, P_q$ over $\mathbb{Z}_p$

[5] Actually, the construction works for any constant number of players, but requires multilinear maps with properties, for which we do not know any instantiation due to recent attacks [GGH13; CHL+15; CGH+15; HJ15; MF15; CLR15; Mar16].

with indeterminates $T_1, \ldots, T_n$ (and degree at most d in each of these indeterminates), the following q group elements:

$$[P_1(a_1, \ldots, a_n)], \ \ldots, \ [P_q(a_1, \ldots, a_n)] \,,$$

with $a_1, \ldots, a_n$ being n uniform secret scalars in $\mathbb{Z}_p$, are indistinguishable from the following q group elements

$$[U(P_1)], \ \ldots, [U(P_q)] \,,$$

where U is a random linear map from the set of multivariate polynomials to $\mathbb{Z}_p$.

This theorem is actually straightforward in the generic group model. The difficulty is to prove the theorem under a standard assumption, namely the decisional d-Diffie-Hellman assumption.

We also extend the theorem to weaker assumptions which are shown to hold in generic groups with symmetric multilinear maps.

1.4.3 Randomness Complexity of Private Circuits

In [BBP+16], with Sonia Belaïd, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud, we study the construction of multiplication circuits secure in the d-probing model, which has been introduced by Ishai, Sahai, and Wagner in [ISW03] to model the security of real implementations of circuits against side-channel attacks. As discovered by Kocher in [Koc96], in real life, an adversary has not only access to the inputs and the outputs of cryptosystems, but has also often access to other information, called side-channel information, such as the power consumption or the electromagnetic radiation of the device that performs the protocol (for example, a credit card that authenticates itself to the bank using some secret). In devices not protected against such side-channel attacks exploiting this additional information, an adversary can often efficiently retrieve important secrets and completely break everything.

The d-probing model is a theoretical model aiming at formalizing what information an attacker can get from side-channel attacks. More precisely, in the d-probing model, we suppose that the adversary can choose d wires (or probes) of the circuit implementing the cryptosystem, and can get the value of these wires.

In [ISW03], Ishai, Sahai, Wagner showed how to generically transform any circuit into one that is secure in the d-probing model. One important tool is the construction of a circuit able to securely perform the and (or multiplication) of two bits. In [BBP+16], we revisit this construction and try to optimize the number of random bits used by the transformed circuit that is secure in the d-probing model.

On the theoretical side, we show a linear lower bound (in d) on the number of random bits, together with an almost matching (non-constructive) upper bound of $O(d \cdot \log d)$. These results are obtained using an algebraic characterization of the d-probing model, which intensively uses linear algebra.

On the practical side, we show a generic construction using half as much random bits as the construction in [ISW03], together with optimal constructions for $d = 2, 3, 4$, which match our lower bound. Furthermore, using our algebraic characterization, we develop a tool which can find attacks on candidate schemes and which is orders of magnitude faster than the previously known tool [BBD+15]. Although our new tool cannot prove the security of a

scheme with 100% accuracy, contrary to the previous tool, it enables to quickly rule out bad candidates and speed up the research of new secure schemes.

The constructions for small values of d are highly likely to be used in real life very soon, as [ISW03] is actually already used in practice, and randomness generation has a very high cost on embedded devices, such as credit cards.

1.4.4 Cryptosystems Based on Residue Symbols

In [BHJL16; JBL14], with Javier Herranz, Marc Joye, and Benoît Libert, we introduce a new public-key encryption scheme based on quadratic residuosity (for some special moduli), similar to the Goldwasser-Micali scheme [GM84]. Contrary to the latter scheme, our scheme can encrypt a large number of bits in one single ciphertext.

1.4.5 Lattice-Based Zero-Knowledge Arguments

In [BCK+14; BKLP15], with Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, Gregory Neven, and Krzysztof Pietrzak, we study lattice-based zero-knowledge arguments, based on Sigma-protocols. Contrary to cyclic and factorization-based groups we are considering in this thesis, lattice-based cryptosystems are noisy, which usually makes everything more complex. In particular, while Sigma-protocols over cyclic groups are generally relatively easy to prove secure, lattice-based ones are much more subtle. We need to carefully take care of the leakage due to noise and of the issues with invertibility of the elements we are considering, as we are working over rings of the form $\mathbb{Z}_q[X]/(X^n + 1)$.

1.4.6 Forward-Secure Signature Schemes

In [ABP13], with Michel Abdalla and David Pointcheval, we studied forward-secure signature schemes, and more precisely the tightness of the reductions of such schemes. In forward-secure signature scheme, signatures and secret keys (used to sign messages) are associated to a time period, for example the current year. A secret key for a given time period T can only be used to produce signatures for this time period T or a subsequent one $T' \geq T$. Furthermore, a secret key for the period T can be updated to period $T + 1$.

The advantage of forward-secure signatures compared to classical digital signatures is the following: if an adversary manages to steal a secret key at some time period T, it still cannot sign messages for previous time periods. Only signatures for the time period T or a subsequent time period might be fake signatures created by the adversary.

1.4.7 Security Proof of J-PAKE

In [ABM15], with Michel Abdalla and Philip MacKenzie, we proved the security of the J-PAKE protocol. The J-PAKE protocol is a password-authenticated key exchange scheme proposed by Hao and Ryan in [HR10]. It is included as an optional protocol in the OpenSSL library [OpenSSL] (enabled using a configuration parameter during install, see directory crypto/jpake), and has been used in various products, such as Firefox Sync [Firefox Sync] and Nest products [Nest] (as part of the Thread protocol [Thread]). Its popularity is mainly due to the fact that it is based on a different paradigm than other schemes, which seems to avoid existing patents, since the scheme is even less efficient than the standard-model schemes we propose in [ABP15d] and requires much stronger security assumptions.

To do the security reduction, we prove an intermediate result which is interesting in its own right: Schnorr proofs [Sch91], one of the most efficient non-interactive zero-knowledge arguments, satisfy a very strong security notion, namely simulation-extractability with non-rewinding extractors, in the random oracle model, assuming algebraic adversaries.

1.4.8 Coppersmith Methods and Analytic Combinatorics

In [BCTV16], with Céline Chevalier, Adrian Thillard, and Damien Vergnaud, we revisit attacks based on Coppersmith methods [Cop96b; Cop96a] and propose a framework based on analytic combinatorics to significantly simplify their analyses.

Personal Publications

Journal Papers

[ABP16] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks". In: *IET Information Security* (2016). To appear. Full version of [ABP15d].

[BHJL16] Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. "Efficient Cryptosystems From 2^k-th Power Residue Symbols". In: *Journal of Cryptology* (2016). ISSN: 1432-1378. DOI: 10.1007/s00145-016-9229-5. URL: http://dx.doi.org/10.1007/s00145-016-9229-5.

[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.

Conference Papers

[ABP13] Michel Abdalla, Fabrice Ben Hamouda, and David Pointcheval. "Tighter Reductions for Forward-Secure Signature Schemes". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 292–311. DOI: 10.1007/978-3-642-36362-7_19.

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. "SPHF-Friendly Non-interactive Commitments". In: *ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013, pp. 214–234. DOI: 10.1007/978-3-642-42033-7_12.

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. "Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 272–291. DOI: 10.1007/978-3-642-36362-7_18.

[BBC+13c] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. "New Techniques for SPHFs and Efficient One-Round PAKE Protocols". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 449–475. DOI: 10.1007/978-3-642-40041-4_25.

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. "Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 77–94. DOI: 10.1007/978-3-662-44371-2_5.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. "Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 551–572. DOI: 10.1007/978-3-662-45611-8_29.

[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. "Security of the J-PAKE Password-Authenticated Key Exchange Protocol". In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 571–587. DOI: 10.1109/SP.2015.41.

[ABP15a] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 388–409. DOI: 10.1007/978-3-662-47989-6_19.

[ABP15b] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved Security". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 103–120. DOI: 10.1007/978-3-662-48797-6_5.

[ABP15c] 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.

[ABP15d] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 332–352. DOI: 10.1007/978-3-662-46447-2_15.

[BCPW15] Fabrice Benhamouda, Geoffroy Couteau, David Pointcheval, and Hoeteck Wee. "Implicit Zero-Knowledge Arguments and Applications to the Malicious Setting". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 107–129. DOI: 10.1007/978-3-662-48000-7_6.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. "Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings". In: *ESORICS 2015, Part I*. Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. LNCS. Springer, Heidelberg, Sept. 2015, pp. 305–325. DOI: 10.1007/978-3-319-24174-6_16.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. "Randomness Complexity of Private Circuits for Multiplication". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 616–648. DOI: 10.1007/978-3-662-49896-5_22.

[BCTV16] Fabrice Benhamouda, Céline Chevalier, Adrian Thillard, and Damien Vergnaud. "Easing Coppersmith Methods Using Analytic Combinatorics: Applications to Public-Key Cryptography with Weak Pseudorandomness". 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. 36–66. DOI: 10.1007/978-3-662-49387-8_3.

Manuscripts

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. *New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange*. Cryptology ePrint Archive, Report 2013/034. http://eprint.iacr.org/2013/034. 2013.

[BP13a] Fabrice Benhamouda and David Pointcheval. *Trapdoor Smooth Projective Hash Functions*. Cryptology ePrint Archive, Report 2013/341. http://eprint.iacr.org/2013/341. 2013.

[BP13b] Fabrice Benhamouda and David Pointcheval. *Verifier-Based Password-Authenticated Key Exchange: New Models and Constructions*. Cryptology ePrint Archive, Report 2013/833. http://eprint.iacr.org/2013/833. 2013.

[ABP14] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. *Removing Erasures with Explainable Hash Proof Systems*. Cryptology ePrint Archive, Report 2014/125. http://eprint.iacr.org/2014/125. 2014.

Patent Applications

[BJL13] Fabrice Benhamouda, Marc Joye, and Benoît Libert. "Method for determining a statistic value on data based on encrypted data". Corresponds to [BJL16]. 2013.

[JBL14] Marc Joye, Fabrice Benhamouda, and Benoît Libert. "Method and device for cryptographic key generation". Corresponds to [BHJL16]. 2014.

Chapter 2

Preliminaries

In this chapter, we introduce the notation used throughout this manuscript and give some cryptographic definitions and notions. The first two sections are classical in cryptography and thus can be quickly skimmed. However, it is highly recommended to read Sections 2.3 and 2.5 very carefully, as they are important for the sequel. A list of notation and abbreviations is provided at the end of the manuscript on pages 181 and 183.

Contents

2.1 Notation and Preliminaries **18**
- 2.1.1 General Notation 18
- 2.1.2 Preliminaries on Provable Security 20
- 2.1.3 Statistical and Computational Indistinguishability 22
- 2.1.4 Proof by Games or Hybrid Arguments 26
- 2.1.5 Cyclic Groups, Bilinear Groups, and Multilinear Groups 26

2.2 Cryptographic Primitives **28**
- 2.2.1 Collision-Resistant Hash Function Families 28
- 2.2.2 Encryption 29
- 2.2.3 Randomness Extractors and Min Entropy 34

2.3 Languages **35**
- 2.3.1 Languages for Projective Hash Functions 36
- 2.3.2 Hard-Subset-Membership Languages 37
- 2.3.3 Language for Zero-Knowledge Arguments 39

2.4 Zero-Knowledge Arguments **39**
- 2.4.1 Overview 39
- 2.4.2 Formal Definitions of Zero-Knowledge Arguments 42
- 2.4.3 Non-Interactive Zero-Knowledge Arguments 47

2.5 Projective Hash Functions **48**
- 2.5.1 Projective Hash Functions (PHFs) 48
- 2.5.2 Smooth Projective Hash Functions (SPHFs) 49
- 2.5.3 Simple Applications of SPHFs 50

2.1 Notation and Preliminaries

2.1.1 General Notation

Sets, integers, moduli, and associated rings and groups. We denote by $\mathbb{Z}$ the set of integers and by $\mathbb{N}$ the set of non-negative integers. If a and b are two integers such that $a < b$, we denote by $\{a, \ldots, b\}$ the set of integers between a and b (a and b included). If $\mathcal{S}$ is a finite set, $|\mathcal{S}|$ denotes its size.

If M is a positive integer, $(\mathbb{Z}_M, +, \cdot)$ or $\mathbb{Z}_M$ denotes the ring of integers modulo M. Sometimes, we only see this ring $\mathbb{Z}_M$ as just an additive group $(\mathbb{Z}_M, +)$. We also consider the multiplicative subgroup $(\mathbb{Z}_M^*, \cdot)$ of $(\mathbb{Z}_M, +, \cdot)$: $\mathbb{Z}_M^*$ contains the invertible elements of the ring $\mathbb{Z}_M$. This group has order $\phi(M)$, where ϕ is Euler's totient function.

In most of our constructions, we work in groups $\mathbb{Z}_M$ with known order M. In this case, elements of $\mathbb{Z}_M$ are represented by integers of the set $\{0, \ldots, M-1\}$, and operations $+$ and $\cdot$ over elements of $\mathbb{Z}_M$ can be performed efficiently in polynomial time (in the size of M). Furthermore, if $x \in \mathbb{Z}$ is an integer, $x \bmod M$ is the remainder of the Euclidean division of x by M. It can be seen both as an integer in $\{0, \ldots, M-1\}$ and as an element of $\mathbb{Z}_M$.

In cases where the order M is not known by our algorithms, elements might have multiple representations whose sizes grow with each ring operation. We deal with this issue separately when it happens.

When $M = p$ is prime, the ring $\mathbb{Z}_M = \mathbb{Z}_p$ is a finite field, and we always suppose that $M = p$ is public in this case.

If R is a ring, then $R[X]$ denotes the ring of polynomials with coefficients in R.

Vector spaces, modules, and matrices. We intensively use vector spaces and modules in this thesis. A module can basically be seen as a vector space over a ring, instead of over a finite field. In this thesis, we focus on $\mathbb{Z}_M$-modules of the form $\mathbb{Z}_M^n$ (with their canonical basis) and their submodules. When $M = p$ is prime, these are $\mathbb{Z}_p$-vector spaces.

Vectors are usually in bold (e.g., $\boldsymbol{u}$, $\boldsymbol{\theta}$), while matrices are usually denoted by capital letters (e.g., A, Γ) but are sometimes also just in bold (e.g., $\boldsymbol{\alpha}$ in Chapter 4). By default vectors are *column* vectors.

If $\boldsymbol{u} \in \mathbb{Z}_M^n$ is a vector, its coordinates or entries are $u_1, \ldots, u_n$ (not in bold). We can also write: $\boldsymbol{u} = (u_i)_{i=1,\ldots,n}$. Sometimes, we consider families of vectors $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ (in bold). The j-th coordinate of $\boldsymbol{u}_i$ (with $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, k\}$) is denoted by $u_{i,j}$. In particular the canonical basis of $\mathbb{Z}_M^n$ is the family $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$, such that $e_{i,j} = 1$ if $i = j$ and $e_{i,j} = 0$ otherwise.

Similarly, if $A \in \mathbb{Z}_M^{n \times k}$ is a matrix with n rows and k columns, its entries are denoted by $A_{i,j}$ with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$. We can also write: $A = (A_{i,j})_{\substack{i=1,\ldots,n \\ j=1,\ldots,k}}$. Sometimes, we consider families of matrices $(A_1, \ldots, A_\ell)$. In that case, the coordinates of the matrix A_l are denoted by $A_{l,i,j}$.

If $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)$ is a family of k vectors in $\mathbb{Z}_M^n$, the submodule of $\mathbb{Z}_M^n$ *generated* (or *spanned*) by these vectors is denoted by:

$$\mathrm{Span}\,(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k) \quad .$$

If $A \in \mathbb{Z}_M^{n \times k}$ is a matrix, then $\mathrm{ColSpan}\,(A)$ is the submodule generated by the columns of the matrix A. It can also be seen as the *image* of the linear function ϕ from $\mathbb{Z}_M^n$ to $\mathbb{Z}_N^k$, which

associates a vector $\boldsymbol{u} \in \mathbb{Z}_M^n$ to $A \cdot \boldsymbol{u}$. The *kernel* of this function is denoted $\ker \phi$ or $\ker A$ and is the set of all vectors $\boldsymbol{u} \in \mathbb{Z}_M^n$ such that $A \cdot \boldsymbol{u} = \mathbf{0}$.

If A is a matrix, $A^\intercal$ is its *transpose*. The identity matrix for the module $\mathbb{Z}_M^n$ is denoted by Id_n. The zero matrix with n rows and k columns is denoted by $\mathbf{0}_{n\times k}$ or just by $\mathbf{0}$ when the dimensions are clear from the context. The zero column (resp. row) vector of size n if denoted by $\mathbf{0}_n$ (resp. $\mathbf{0}_n^\intercal$) or just by $\mathbf{0}$ (resp. $\mathbf{0}^\intercal$) when the dimensions are clear from context.

Prime decomposition. Let $M \geq 2$ be a positive integer. We say that $M = p_1^{e_1} \cdots p_r^{e_r}$ is the *prime decomposition* of M if $p_1, \ldots, p_r$ are distinct prime numbers and $e_1, \ldots, e_r$ are positive integers. We recall that the prime decomposition of a positive integer is unique up to the order of its prime factors $p_1, \ldots, p_r$.

Legendre symbol, Jacobi symbol, and quadratic residue. Let p be a prime number and x be an element of $\mathbb{Z}_p$. The Legendre symbol of x modulo p is:

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x = 0 \in \mathbb{Z}_p\,, \\ 1 & \text{if } \exists y \in \mathbb{Z}_p^*,\ x = y^2\,, \\ -1 & \text{otherwise.} \end{cases}$$

Let $M \geq 2$ be a positive integer and let $M = p_1^{e_1} \cdots p_r^{e_r}$ be its prime decomposition. We recall that there exists a canonical surjection from $\mathbb{Z}_M$ to $\mathbb{Z}_{M'}$ for any factor $M' \geq 1$ of M. Let x be an element of $\mathbb{Z}_M$. The image of the element x of $\mathbb{Z}_M$ by this surjection is denoted $x \bmod M'$. The Jacobi symbol of x modulo M is:

$$\left(\frac{x}{M}\right) = \prod_{i=1}^{r} \left(\frac{x \bmod p_i}{p_i}\right)^{e_i}\,.$$

Furthermore, we say that x is a *quadratic residue modulo* M if and only if there exists an element $y \in \mathbb{Z}_M^*$ such that $x = y^2$. The *Chinese remainder theorem* (CRT)[1] directly implies that if x is a quadratic residue modulo M, its Jacobi symbol modulo M is equal to 1. The converse is only true when M is a prime number.

Bit strings. The set of all bit strings is denoted by $\{0,1\}^*$, while the set of bit strings of length n is $\{0,1\}^n$. A bit string is often denoted by a lowercase letter, such as x or y. If $x \in \{0,1\}^*$, its length is $|x|$ and the i-bit is x is x_i (for $i \in \{1, \ldots, |x|\}$). The *exclusive or* between two bit string x and y of the same length is denoted by $x \mathsf{xor}\, y$.

Miscellaneous. The value "true" is represented by 1, while the value "false" is represented by 0. Logarithm log is always in basis 2: $\log = \log_2$. The binary operator ":=" is used to define a symbol. To indicate that some variable x is assigned the result of some computation, e.g., $y + z$, we write $x \leftarrow y + z$. In many cases, ":=" and "$\leftarrow$" are used interchangeably.

If x is a real number, $|x|$ is its absolute value.

When f and g are two functions from $\mathbb{N}$ to the set of real numbers, we write $f = O(g)$ or $g = \Omega(f)$ to indicate that there exists a constant c and an integer $n_0 \in \mathbb{N}$ such that for any integer $n \geq n_0$, $|f(n)| \leq c \cdot |g(n)|$.

[1] We recall the CRT in Section 4.2.2.3 when using it.

Algorithms, Turing machines, and interactive Turing machines. Algorithms are programs for Turing machines. By default, algorithms are probabilistic, i.e., they can use an additional tape of the Turing machine containing random bits (a.k.a., *random coins*). If A is an algorithm, we write $y \xleftarrow{\$} A(x)$ to say that we execute the algorithm A on input x with fresh random coins and that we store the result in y. If A is deterministic, we also write $y \leftarrow A(x)$. We write $A(x; r)$ to explicitly indicate that A used the random coins r (seen as a bit string). When we write "for any $y \xleftarrow{\$} A(x)$", we mean that we consider all the possible outputs $A(x; r)$ of the algorithm A on input x, for any possible random coins r.

An *interactive Turing machine* (ITM) is special kind of Turing machine with additional tapes to communicate with other Turing machines: one communication input tape to receive messages from other machines and one communication output tape to send messages to other machines. It is used in the modelization of interactive protocols. In practice however, we never formally describe an ITM but often use figures as Figure 2.8 on page 51.

By default algorithms and ITM are not supposed to run in polynomial time. If we want them to be polynomial time, we say it explicitly. By polynomial time, we mean strong polynomial time and not expected polynomial time. In most cases, this is without loss of generality, as our algorithms are probabilistic anyway (and we often allow for errors).

We only consider *uniform complexity*: algorithms and Turing machines do not have access to an advice string depending on the length of their input.

Sets and distributions. If $\mathcal{S}$ is a set, $x \xleftarrow{\$} \mathcal{S}$ indicates that x is taken uniformly at random from the set $\mathcal{S}$ (independently of everything else). Similarly, if $\mathcal{D}$ is a probability distribution, $x \xleftarrow{\$} \mathcal{D}$ indicates that x is drawn randomly according to $\mathcal{D}$. We often use the term "random" to mean "uniformly at random".

Finally, we also write $x, y \xleftarrow{\$} \mathcal{S}$ to indicate that x and y are drawn independently uniformly at random from $\mathcal{S}$.

2.1.2 Preliminaries on Provable Security

2.1.2.1 Security Parameters and Negligibility

Almost any cryptosystem can be broken with a powerful enough computer.[2] The goal is just to construct fast enough cryptosystems which cannot be broken by reasonable computers. In practice, we often consider today that if 2^{128} elementary operations[3] are required to break a cryptosystem with high probability (e.g., $1/2$), then this cryptosystem is considered secure. Such a cryptosystem is said to provide $\mathfrak{K} = 128$ bits of security.

To properly formalize this idea of security, theoretical papers like this thesis use the notion of a security parameter $\mathfrak{K} \in \mathbb{N}$. We do as if all the algorithms of a cryptosystem take as input a unary representation $1^{\mathfrak{K}}$ of this security parameter and that all these algorithms run in polynomial time in this security parameter. We then generally consider only attacks against the cryptosystem which can be performed by polynomial time algorithms or *adversaries*. Details for this are given in the next section.

[2] There exists purely information theoretic cryptography though, but even in that case, we may need to use a security parameter.

[3] Elementary operations depend on the exact computer architecture but, in practice, there are not that many differences between the computer architectures.

But before going to the next section, we need the following notions. Let ε be a function from $\mathbb{N}$ to reals between 0 and 1. We say that ε is *negligible* or $1-\varepsilon$ is *overwhelming*, if for any $k \in \mathbb{N}$, $\varepsilon = O(1/\mathfrak{K}^k)$.

2.1.2.2 Adversaries and Experiments

Adversaries. Adversaries are probabilistic algorithms or Turing machines. They are often denoted by calligraphic letters, such as $\mathcal{A}$ or $\mathcal{B}$ and may or may not run in polynomial time. If they do not run in polynomial time, we say that they are *unbounded*. Adversaries may need to be called multiple times. In that case, they may or may not carry a state st.

In any case, adversaries are always supposed to implicitly take as input a unary representation of the security parameter. As inputs to adversaries are always polynomial in the security parameter, a polynomial-time adversary runs in time polynomial in the security parameter.

Experiments, games, and oracles. Security notions and security assumptions are often described as *experiments* where an adversary is called one or several times with various inputs. An adversary may also be given access to oracles, which are also Turing machines. The running time of the oracles are not taken into account in the running time of the adversary: a query to an oracle always only counts for one clock tick. We write $\mathcal{A}^{\mathsf{O}_1,\mathsf{O}_2}(x)$ to say that the adversary $\mathcal{A}$ is called with input x and has access to the oracles O_1 and O_2.

An experiment can also be seen as a *game* between an adversary $\mathcal{A}$ and an implicit challenger which gives its input to the adversary and allows some oracle calls. In this thesis, we use the terms *experiment* and *game* interchangeably.

As usual, experiments are parameterized by the security parameter $\mathfrak{K}$. Examples of experiments can be found in Figure 2.4 on page 45.

Success probability. The *success probability* of an adversary $\mathcal{A}$ in an experiment $\mathsf{Exp}^{\mathtt{exp}}$ is the probability that this adversary outputs 1 in this experiment:

$$\mathsf{Succ}^{\mathtt{exp}}(\mathcal{A},\mathfrak{K}) := \Pr\left[\,\mathsf{Exp}^{\mathtt{exp}}(\mathcal{A},\mathfrak{K}) = 1\,\right] \quad .$$

It depends on the security parameter $\mathfrak{K}$. When the experiment $\mathsf{Exp}^{\mathtt{exp}}$ corresponds to a cryptographic assumption or to a security notion, we say that the assumption or the security notion *statistically holds* when this success probability is negligible for any unbounded adversary $\mathcal{A}$. It *perfectly holds* when this success probability is 0 for any unbounded adversary $\mathcal{A}$. It *(computationally) holds* when this success probability is negligible for any polynomial-time adversary $\mathcal{A}$.

Advantage. Sometimes, a security notion or assumption consists in distinguishing two experiments $\mathsf{Exp}^{\mathtt{exp-0}}$ and $\mathsf{Exp}^{\mathtt{exp-1}}$. In this case, we define the *advantage* of an adversary $\mathcal{A}$ in distinguishing the experiments $\mathsf{Exp}^{\mathtt{exp}-b}$ (where b is always supposed to be in $\{0,1\}$ in this context) as:

$$\mathsf{Adv}^{\mathtt{exp}}(\mathcal{A},\mathfrak{K}) = \left|\Pr\left[\,\mathsf{Exp}^{\mathtt{exp-1}}(\mathcal{A},\mathfrak{K}) = 1\,\right] - \Pr\left[\,\mathsf{Exp}^{\mathtt{exp-0}}(\mathcal{A},\mathfrak{K}) = 1\,\right]\right| \quad .$$

When the experiments $\mathsf{Exp}^{\mathtt{exp}-b}$ correspond to a cryptographic assumption or to a security notion, we say that the assumption or the security notion *statistically holds* when this

advantage is negligible for any unbounded adversary $\mathcal{A}$. It *perfectly holds* when this advantage is 0 for any unbounded adversary $\mathcal{A}$. It *(computationally) holds* when this advantage is negligible for any polynomial-time adversary $\mathcal{A}$.

Hardness and efficiency. An *efficient* algorithm is a algorithm running in polynomial time (in the security parameter). A problem is said to be *easy* if it can be solved in polynomial time. A problem is said to be *hard* if it cannot be solved in polynomial time, with non-negligible probability.

2.1.3 Statistical and Computational Indistinguishability

We have already seen that the notion of advantage captures the general idea of indistinguishability of two experiments $\mathsf{Exp}^{\mathtt{exp-0}}$ and $\mathsf{Exp}^{\mathtt{exp-1}}$ (both statistical, when the adversary is unbounded, and computational, when it runs in polynomial time). In many cases, these experiments just consist in drawing an element from some distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively and feeding it as input to the adversary. The output of the experiment is just the output of the adversary.

In this section, we further study these particular cases and give some additional vocabulary and tools.

2.1.3.1 Statistical Indistinguishability

Definition. To define statistical indistinguishability, we first need to define statistical distance. We suppose that the reader is familiar with basic probability and the notion of distributions and random variables. There are many ways to define statistical distance in mathematics and computer science. We use (one of) the most common one in cryptography, which corresponds to the total variation distance.

Definition 2.1.1. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distributions over some finite set $\mathcal{S}$. Let X_1 and X_2 be two random variables sampled according to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. Their* ranges *are therefore the same finite set $\mathcal{S}$. The* statistical distance *between $\mathcal{D}_1$ and $\mathcal{D}_2$ (or X_1 and X_2) is:*

$$\mathsf{Dist}(\mathcal{D}_1, \mathcal{D}_2) := \mathsf{Dist}(X_1, X_2) := \frac{1}{2} \cdot \sum_{x \in \mathcal{S}} |\Pr[\, X_1 = x \,] - \Pr[\, X_2 = x \,]| \quad .$$

We use distributions and random variables interchangeably, depending on what is the most convenient.

The following theorem gives a very useful and equivalent definition of statistical distance.

Theorem 2.1.2. *Let X_1 and X_2 be two random variables with some finite range $\mathcal{S}$. We have:*

$$\mathsf{Dist}(X_1, X_2) = \max_{\mathcal{A}} |\Pr[\, \mathcal{A}(X_1) = 1 \,] - \Pr[\, \mathcal{A}(X_2) = 1 \,]| \ ,$$

where the maximum is taken over all the possible adversaries taking as input an element of $\mathcal{S}$ outputting a bit. The definition is equivalent whether the adversaries are probabilistic or deterministic.

Proof. First, we remark that for any adversary $\mathcal{A}$:

$$|\Pr[\, \mathcal{A}(X_1) = 1 \,] - \Pr[\, \mathcal{A}(X_2) = 1 \,]| = \left| \sum_{x \in \mathcal{S}} \Pr[\, \mathcal{A}(x) = 1 \,] \cdot (\Pr[\, X_1 = x \,] - \Pr[\, X_2 = x \,]) \right| \ ,$$

because for any $x \in \mathcal{S}$, the events $\mathcal{A}(x) = 1$ and $X_1 = 1$ are independent (and $\mathcal{A}(x) = 1$ and $X_2 = 1$ are independent too). If for some $x \in \mathcal{S}$, $0 < \Pr[\mathcal{A}(x) = 1] < 1$, then the above expression can be increased by increasing or decreasing this probability. That shows that we can restrict ourselves to deterministic adversaries.

Second, let $\mathcal{A}$ be a deterministic adversary and let B be the subset of $\mathcal{S}$ defined by:

$$B := \{x \in \mathcal{S} \mid \mathcal{A}(x) = 1\} \ .$$

We have:

$$\begin{aligned}
&|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]| \\
&= |\Pr[X_1 \in B] - \Pr[X_2 \in B]| = |\Pr[X_1 \in \mathcal{S} \setminus B] - \Pr[X_2 \in \mathcal{S} \setminus B]| \\
&= \frac{1}{2} \cdot (|\Pr[X_1 \in B] - \Pr[X_2 \in B]| + |\Pr[X_1 \in \mathcal{S} \setminus B] - \Pr[X_2 \in \mathcal{S} \setminus B]|) \ .
\end{aligned}$$

Furthermore, for any subset $C \subseteq \mathcal{S}$:

$$|\Pr[X_1 \in C] - \Pr[X_2 \in C]| \leq \sum_{x \in C} |\Pr[X_1 = x] - \Pr[X_2 = x]| \ .$$

Therefore, we get:

$$\begin{aligned}
&|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]| \\
&\leq \frac{1}{2} \cdot \left(\sum_{x \in B} |\Pr[X_1 = x] - \Pr[X_2 = x]| + \sum_{x \in \mathcal{S} \setminus B} |\Pr[X_1 = x] - \Pr[X_2 = x]| \right) \\
&= \mathsf{Dist}(X_1, X_2) \ .
\end{aligned}$$

Thus, we have:

$$\sup_{\mathcal{A}} |\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]| \leq \mathsf{Dist}(X_1, X_2) \ .$$

To conclude, let us now construct an adversary $\mathcal{A}$ such that:

$$|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]| = \mathsf{Dist}(X_1, X_2) \ .$$

Let B be the following set:

$$B := \{x \in \mathcal{S} \mid \Pr[X_1 = x] \geq \Pr[X_2 = x]\} ,$$

and $\mathcal{A}$ be the adversary which on input $x \in \mathcal{S}$ outputs 1 if and only if $x \in B$. Using the previous analysis, we then have:

$$\begin{aligned}
&|\Pr[\mathcal{A}(X_1) = 1] - \Pr[\mathcal{A}(X_2) = 1]| \\
&= \frac{1}{2} \cdot (|\Pr[X_1 \in B] - \Pr[X_2 \in B]| + |\Pr[X_1 \in \mathcal{S} \setminus B] - \Pr[X_2 \in \mathcal{S} \setminus B]|) \\
&= \frac{1}{2} \cdot \left(\left| \sum_{x \in B} (\Pr[X_1 = x] - \Pr[X_2 = x]) \right| + \left| \sum_{x \in \mathcal{S} \setminus B} (\Pr[X_1 = x] - \Pr[X_2 = x]) \right| \right) \\
&= \frac{1}{2} \cdot \left(\sum_{x \in B} |\Pr[X_1 = x] - \Pr[X_2 = x]| + \sum_{x \in \mathcal{S} \setminus B} |\Pr[X_1 = x] - \Pr[X_2 = x]| \right) \\
&= \mathsf{Dist}(X_1, X_2) ,
\end{aligned}$$

where the last-but-one inequality comes from the fact that $\Pr[X_1 = x] - \Pr[X_2 = x]$ has the same sign for all $x \in B$ on one hand, and for all $x \in \mathcal{S} \setminus B$ on the other hand. This concludes the proof. □

Definition 2.1.3. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ two distributions over some finite set $\mathcal{S}$. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-*statistically indistinguishable *or ε-*close *if their statistical distance $\mathsf{Dist}(\mathcal{D}_1, \mathcal{D}_2)$ is at most ε.*

Furthermore, if $(\mathcal{D}_{1,\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$ and $(\mathcal{D}_{2,\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$ are distribution ensembles *(i.e., families of distributions) over finite sets, we say that they are* statistically indistinguishable *if their statistical distance $\mathsf{Dist}(\mathcal{D}_{1,\mathfrak{K}}, \mathcal{D}_{2,\mathfrak{K}})$ is negligible in $\mathfrak{K}$.*

For simplicity, we often say "distribution" instead of "distribution ensemble" and the security parameter is often implicit.

Properties. Let us state some useful properties, which we use throughout the thesis sometimes without explicitly citing them, as they are classical.

Proposition 2.1.4. *Let X_1, X_2, and X_3 be three random variables with some finite range $\mathcal{S}$. Then, we have:*

$$\mathsf{Dist}(X_1, X_3) \leq \mathsf{Dist}(X_1, X_2) + \mathsf{Dist}(X_2, X_3) \ .$$

This proposition is essentially what underlies the idea of proofs by games or hybrid arguments (in the statistical case) sketched later in Section 2.1.4.

Proof. The proof follows directly from the triangular inequality. □

Proposition 2.1.5 (data processing inequality)**.** *Let X_1 and X_2 be two random variables with some finite range $\mathcal{S}$. Let f be any function (not necessarily efficiently computable) from $\mathcal{S}$ to some other finite set $\mathcal{S}'$. Then we have:*

$$\mathsf{Dist}(f(X_1), f(X_2)) \leq \mathsf{Dist}(X_1, X_2) \ .$$

This proposition essentially says that processing the output of two distributions using any function f does not increase their statistical distance.

Proof. The proof follows directly from the triangular inequality. □

Proposition 2.1.6. *Let X_1 and X_2 be two random variables with some finite range $\mathcal{S}$. Let Y be a Bernoulli random variable independent from X_1 and of parameter $\varepsilon < 1$: $Y = 0$ with probability $1 - \varepsilon$ and $Y = 1$ with probability ε. Let X_1' be another random variable such that:*

$$\Pr[X_1' = X_1 \mid Y = 0] = 1 \,,$$

or in other words, such that when $Y = 0$, X_1' coincides with X_1. Then, we have:

$$\mathsf{Dist}(X_1', X_2) \leq \varepsilon + \mathsf{Dist}(X_1, X_2) \ .$$

This proposition is an extension of Lemma 1 of [Sho01] to statistical distance.

Proof. Let us just prove it when $X_2 = X_1$, i.e., let us prove that:

$$\mathsf{Dist}(X_1', X_1) \leq \varepsilon \ .$$

The general version follows from Proposition 2.1.4.

We have:

$$\begin{aligned}
2 \cdot \mathsf{Dist}(X_1', X_1) &= \sum_{x \in \mathcal{S}} \left| \Pr\left[\, X_1' = x \,\right] - \Pr\left[\, X_1 = x \,\right] \right| \\
&= \sum_{x \in \mathcal{S}} \Big| \Pr\left[\, Y = 0 \,\right] \cdot \Pr\left[\, X_1' = x \mid Y = 0 \,\right] \\
&\qquad + \Pr\left[\, Y = 1 \,\right] \cdot \Pr\left[\, X_1' = x \mid Y = 1 \,\right] - \Pr\left[\, X_1 = x \,\right] \Big| \\
&= \sum_{x \in \mathcal{S}} \Big| (1 - \varepsilon) \cdot \Pr\left[\, X_1' = x \mid Y = 0 \,\right] \\
&\qquad + \varepsilon \cdot \Pr\left[\, X_1' = x \mid Y = 1 \,\right] - \Pr\left[\, X_1 = x \,\right] \Big| \ .
\end{aligned}$$

As X_1' and X_1 coincides when $Y = 0$, $\Pr\left[\, X_1' = x \mid Y = 0 \,\right] = \Pr\left[\, X_1 = x \,\right]$, and:

$$\begin{aligned}
2 \cdot \mathsf{Dist}(X_1', X_1) &= \sum_{x \in \mathcal{S}} \left| -\varepsilon \cdot \Pr\left[\, X_1 = x \,\right] + \varepsilon \cdot \Pr\left[\, X_1' = x \mid Y = 1 \,\right] \right| \\
&\leq \varepsilon \cdot \sum_{x \in \mathcal{S}} \left(\Pr\left[\, X_1 = x \,\right] + \Pr\left[\, X_1' = x \mid Y = 1 \,\right] \right) \\
&\leq 2 \cdot \varepsilon \ .
\end{aligned}$$

This concludes the proof. □

Proposition 2.1.7. *Let M and N be two positive integers, such that $N \geq M$. Let U_M and U_N be a uniform random variable over $\{0, \ldots, M-1\}$ and $\{0, \ldots, N-1\}$. Then the distance between U_M and U_N is:*

$$\mathit{Dist}(U_N, U_M) = \frac{N - M}{N} \ .$$

Proof. We have:

$$\begin{aligned}
\mathit{Dist}(U_N, U_M) &= \frac{1}{2} \cdot \left(\sum_{x=0}^{M-1} \left| \frac{1}{N} - \frac{1}{M} \right| + \sum_{x=M}^{N-1} \left| \frac{1}{N} - 0 \right| \right) \\
&= \frac{1}{2} \cdot \left(M \cdot \frac{N - M}{MN} + (N - M) \cdot \frac{1}{N} \right) \\
&= \frac{N - M}{N} \ .
\end{aligned}$$

□

Corollary 2.1.8. *Let M and N be two positive integers, such that $N \geq M$. Let U_M and U_N be a uniform random variable over $\{0, \ldots, M-1\}$ and $\{0, \ldots, N-1\}$. Then the distance between U_M and $U_N \bmod M$ is:*

$$\mathit{Dist}(U_N \bmod M, U_M) \leq \frac{N \bmod M}{N} \ .$$

Proof. Let $L := N - (N \bmod M)$ and let U_L be a uniform random variable over $\{0, \ldots, L-1\}$. We remark that the distribution $U_L \bmod M$ is the same as the distribution of U_M, as M divides L. Therefore, we have:

$$\begin{aligned}\mathsf{Dist}(U_N \bmod M,\, U_M) &= \mathsf{Dist}(U_N \bmod N,\, U_L \bmod M)\\ &\leq \mathsf{Dist}(U_N, U_L) = \frac{N-L}{N} = \frac{N \bmod M}{N}\,,\end{aligned}$$

where the inequality comes from Proposition 2.1.5 and the second to last equality comes from Proposition 2.1.7. □

2.1.3.2 Computational Indistinguishability

Let us now recall the definition of computational indistinguishability.

Definition 2.1.9. *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distribution ensembles which can be sampled in polynomial time in $\mathfrak{K}$. Let $\mathcal{A}$ be a polynomial-time adversary outputting a bit. For any security parameter $\mathfrak{K}$, the* advantage *of $\mathcal{A}$ in distinguishing $\mathcal{D}_1$ and $\mathcal{D}_2$ is:*

$$\mathsf{Adv}^{\mathcal{D}_1,\mathcal{D}_2}(\mathcal{A}, \mathfrak{K}) := \left|\Pr\left[\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{D}_1\right] - \Pr\left[\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{D}_0\right]\right| \quad .$$

Then, $\mathcal{D}_1$ and $\mathcal{D}_2$ are said to be computationally indistinguishable *if for any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{D}_1,\mathcal{D}_2}(\mathcal{A}, \mathfrak{K})$ is negligible in $\mathfrak{K}$.*

If we remove the requirement that the adversary has to run in polynomial time, we get exactly the notion of statistical indistinguishability, due to Theorem 2.1.2.

2.1.4 Proof by Games or Hybrid Arguments

Most of our security proofs are proofs by games (also called hybrid arguments) as defined by Shoup in [Sho01; KR01; BR06]: to bound a success probability in some game $\mathbf{G}_0$ corresponding to some security notion, we construct of sequence of games. The first game is $\mathbf{G}_0$, while the last game corresponds to some security notion or is such that the adversary just cannot win. Furthermore, we prove that two consecutive games are indistinguishable either perfectly, statistically, or computationally. In other words, we bound the difference of success probabilities by a negligible quantity.

Similarly, to bound an advantage of an adversary in distinguishing two games $\mathbf{G}_0$ and $\mathbf{G}_1$, we construct a sequence of indistinguishable games starting with $\mathbf{G}_0$ and ending with $\mathbf{G}_1$.

2.1.5 Cyclic Groups, Bilinear Groups, and Multilinear Groups

In this thesis, we use a lot cyclic groups and bilinear groups.

Cyclic groups. Cyclic groups are groups generated by a single element (and thus are commutative or Abelian). A cyclic group of order M is isomorphic to the additive group $\mathbb{Z}_M$. However, a given function or operation on a group of order M may have very different complexities depending the exact group (representation) we are considering.

In this thesis, we denote a cyclic group of order M generated by g, by a tuple $(M, \mathbb{G}, g)$ or just $(M, \mathbb{G})$. The set of generators of the group $\mathbb{G}$ is denoted $\mathbb{G}^*$. The group law is

multiplicative and we suppose that it can be efficiently computed. In particular, this means that given $x \in \mathbb{Z}_M$, the element g^x can be efficiently computed too, which implies that exponentiations can be efficiently computed.

However, for cyclic groups to be useful in cryptography, we often require that that the reverse operation of exponentiation, which is called the *discrete logarithm* operation, is hard: given two element $g, g^x \in \mathbb{G}$, it should be hard to compute the scalar x, which is called the *discrete logarithm* of g^x in basis g. The basis is often implicit. We often consider cyclic groups of prime order $M = p$.

There are two main classes of cyclic groups in cryptography: multiplicative subgroups of finite fields and (subgroups of) elliptic curves [Mil86; Kob87]. The latter is still used in practice but is significantly less efficient for current security parameters. Current best practices consist in using elliptic curves, such as Curve25519 [Ber06]. For security parameter $\mathfrak{K}$, we can generally use elliptic curves of prime order of only $2\mathfrak{K}$ bits, as it is believed that only generic attacks (in time square root of the order p, like Shank's baby-step giant-step attack [Sha71]) apply to reasonable elliptic curves. Group elements (i.e., points of the curve) can then also be represented by about $2\mathfrak{K} + 1$ bits, using point compression. For simplicity, we term "+1" is often omitted.

As usual, when we are referring to one cyclic group $(p, \mathbb{G}, g)$, we are actually referring to a family of cyclic groups indexed by the security parameter $\mathfrak{K}$.

Bilinear groups. Some constructions require additional structures such as pairings or bilinear groups.

When $(M, \mathbb{G}_1, g_1)$, $(M, \mathbb{G}_2, g_2)$, and $(M, \mathbb{G}_T, g_T)$ are three cyclic groups of the same order M, $(M, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ or $(M, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is called a *bilinear group* if e is a map (called a *pairing*) from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ satisfying the following properties:

- *Bilinearity.* For all $(a, b) \in \mathbb{Z}_M^2$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- *Non-degeneracy.* The element $e(g_1, g_2) = g_T$ generates $\mathbb{G}_T$;
- *Efficiency.* The function e is efficiently computable.

It is called a *symmetric* (or *type 1*) bilinear group if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. In this case, we denote it $(p, \mathbb{G}, \mathbb{G}_T, e)$ and we suppose $g = g_1 = g_2$. Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an *asymmetric* (or *type 3*)[4] bilinear group.

Bilinear groups can be constructed from elliptic curves. Currently, asymmetric ones are more efficient (both in term of time to compute the group law or the pairing e and in term of the size of the representation of group elements).

For $\mathfrak{K} = 128$ bits of security, we can use *asymmetric bilinear groups* with prime order p of about 256 bits, elements in $\mathbb{G}_1$ represented by about 256 bits, elements of $\mathbb{G}_2$ represented by 512 bits, and elements of $\mathbb{G}_T$ represented by 3072 bits [BN06] (for the reader familiar with elliptic curves, these elliptic curves have embedding degree 12).

Unfortunately, for prime order *symmetric bilinear groups*, due to recent attacks [Jou14; GGMZ13; BGJT14; AMOR14], the best choice seem to be supersingular elliptic curves over a prime order finite field, and therefore of embedding degree 2. These make parameters much worse. Concretely, for $\mathfrak{K} = 128$ bits of security, elements of $\mathbb{G}$ are represented by about 1536

[4] Type 2 bilinear groups correspond to the case where there is an efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. We never use this case in this thesis.

Table 2.1: Relative time of operations on two curves (one with pairing and one without)

Curve \ Efficiency	Pairing	Exponentiation in $\mathbb{G}_1$ or $\mathbb{G}$	Exponentiation in $\mathbb{G}_2$
Curve25519 [Ber06]	no pairing	1	✗
[BGM+10]	≈ 8	≈ 3	≈ 6

bits, while elements of $\mathbb{G}_T$ are represented by about 3072 bits, for a prime order p of about 256 bits.

Efficiency of cyclic and bilinear groups. Both cyclic and bilinear groups are considered practical nowadays, at least for prime orders. For composite-order groups, we usually need the order to be hard to factor which makes it much larger, as there exist sub-exponential factorization algorithms [Gui13].

However, we should point out that operations over cyclic groups without efficient pairing can be much faster, when using specific curves like Curve25519 [Ber06]. In Table 2.1, we show the efficiency difference between two reasonable choices of curves with and without efficient pairings respectively. Furthermore, curves without efficient pairings might be less susceptible to recent breakthroughs in discrete log attacks [Jou14; GGMZ13; BGJT14; AMOR14; GKZ14]. For these reasons, it is usually preferable to construct protocols without pairings.

Additional notation. The element 1 might be either the element $1 \in \mathbb{Z}_M$ or the element $g^0 \in \mathbb{G}$, $g_1^0 \in \mathbb{G}_1$, ... When this is not clear from context, we explicitly add an index and write $1_{\mathbb{Z}_M}$, $1_{\mathbb{G}}$, ...

2.2 Cryptographic Primitives

Let us now introduce some of the classical cryptographic primitives used throughout this thesis.

2.2.1 Collision-Resistant Hash Function Families

Definition 2.2.1. *A* collision-resistant hash function family *is defined by a tuple of two polynomial-time algorithms* $\mathcal{HF} = (\mathsf{H.Setup}, \mathsf{H.Eval})$*:*

- $\mathsf{H.Setup}(1^{\mathfrak{K}})$ *outputs a parameter* par*;*
- $\mathsf{H.Eval}(\mathsf{par}, \mathsf{m})$ *deterministically outputs some bit string* $y \in \{0,1\}^*$*, called the* hash value *of the input* $\mathsf{m} \in \{0,1\}^*$*.*

It should satisfy the following property:

- Collision resistance. *A polynomial-time adversary cannot find two inputs* m_0 *and* m_1 *with the same hash value, with non-negligible probability. The success probability of an adversary* $\mathcal{A}$ *against collision resistance is defined by the experiment* Exp^{coll} *depicted in Figure 2.1.* $\mathcal{HF}$ *is collision-resistant if this success probability is negligible for any polynomial-time adversary* $\mathcal{A}$*.*

Collision Resistance

$\mathsf{Exp}^{\mathtt{coll}}(\mathcal{A}, \mathfrak{K})$
  $\mathsf{par} \xleftarrow{\$} \mathsf{H.Setup}(1^{\mathfrak{K}})$
  $(\mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}(\mathsf{par})$
  if $\mathsf{H.Eval}(\mathsf{par}, \mathsf{m}_0) = \mathsf{H.Eval}(\mathsf{par}, \mathsf{m}_1)$ **then**
    return 1
  else
    return 0

Figure 2.1: Experiment for Definition 2.2.1 (collision resistance)

Since the notation used in this definition is cumbersome, we often write: $\mathcal{H} \xleftarrow{\$} \mathcal{HF}_{\mathfrak{K}}$ or even just $\mathcal{H} \xleftarrow{\$} \mathcal{HF}$, instead of $\mathsf{par} \xleftarrow{\$} \mathsf{H.Setup}(1^{\mathfrak{K}})$, where $\mathcal{H}$ represents the function $\mathsf{m} \mapsto \mathsf{H.Eval}(\mathsf{par}, \mathsf{m})$. Furthermore, for the sake of simplicity, we suppose that $\mathcal{H}$ can take several parameters and first converts them (efficiently) into a single bit string which is then hashed. For example, we write $\mathcal{H}(u, v)$ (e.g., with u and v being two group elements), to say that the bit string representation of the tuple (u, v) is hashed.

We use a family of functions depending on some parameter par instead of a single function for theoretical reasons. Without it, a non-uniform adversary could just have a collision hard-wired. This would not be such an issue with a uniform adversary, but the notion of hash function family is easier to handle in theory. However, in practice, we can just use SHA-256 [NIS12] for $\mathfrak{K} = 128$ bits of security, for example.

2.2.2 Encryption

Classical (a.k.a, *symmetric* or *secret-key*) encryption is one of the most fundamental primitives in cryptography. It enables two users sharing a secret key K to communicate confidentially. Each user *encrypts* the message he wants to send to the other user, using the key K; and the other user *decrypts* the received encrypted message (a.k.a., *ciphertext*) to get the original message back, using the same key K. Anybody who intercepts the ciphertext should not be able to learn anything about the original message, without knowledge of the secret key K.

Public-key encryption schemes are more powerful: they enable to encrypt a message to a user if we know his *public key* or *encryption key* ek. Decryption can only be realized using the associated *private key* or *decryption key* dk which is kept private by the user. In this thesis, we consider labeled public-key encryption scheme [Sho06], where "labeled" means that a ciphertext can be associated to a *label* which can be seen as some public metadata.

2.2.2.1 Definition

Definition 2.2.2. *An* (labeled) encryption scheme *is defined by a tuple of four polynomial-time algorithms* $(\mathsf{Setup.gpar}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ *outputs some global parameters* gpar*;*
- $\mathsf{KeyGen}(\mathsf{gpar})$ *generates a pair* $(\mathsf{ek}, \mathsf{dk})$*, where* ek *is an* encryption key *(a.k.a.,* public key*) and* dk *is a* decryption key *(a.k.a.,* secret key*); these two keys are supposed to implicitly contain the global parameters* gpar*;*

- $\mathsf{Enc}(\mathsf{ek}, \ell, \mathsf{m})$ *generates a bit string* c*, called a* ciphertext*, encrypting the message (a.k.a.,* plaintext*)* m *with label* ℓ *under the encryption key* ek*; the plaintext* m *is supposed to be in some set* $\mathscr{M}$ *which might depend in the global parameters but which is efficiently recognizable (i.e., testing whether* $\mathsf{m} \in \mathscr{M}$ *or not can be done in polynomial time in the security parameter* $\mathfrak{K}$*); similarly the label* ℓ *is in some set* Labels *which is also efficiently recognizable;*

- $\mathsf{Dec}(\mathsf{dk}, \ell, \mathsf{c})$ *decrypts the ciphertext* c *with label* ℓ *and outputs the corresponding plaintext* m *or* $\perp$ *if the decryption failed.*

It should satisfy the following property:

- Perfect correctness. *For any global parameters* $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$*, for any key pair* $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{gpar})$*, for any label* ℓ*, for any message* $\mathsf{m} \in \mathscr{M}$*, for any ciphertext* $\mathsf{c} \xleftarrow{\$} \mathsf{Enc}(\ell, \mathsf{ek}, \mathsf{m})$*, we have* $\mathsf{Dec}(\mathsf{dk}, \ell, \mathsf{c}) = \mathsf{m}$ *(with probability* 1*, if* Dec *is probabilistic).*

If no label is used, the scheme is called a *non-labeled encryption scheme*, the set Labels is $\{\perp\}$, and the input ℓ is omitted in Enc and Dec.

The previous definition of encryption scheme does not provide any security guarantee: the encryption procedure could just return $\mathsf{c} := \mathsf{m}$ and it would still satisfy the above definition. There are many possible security notions for encryption schemes. In this thesis, we only consider two of them: *indistinguishability under chosen plaintext attacks* (IND-CPA) and *indistinguishability under chosen ciphertext attacks* (IND-CCA).[5]

IND-CPA basically ensures that a ciphertext does not reveal any information about its plaintext if we only know the encryption key ek but not the decryption key dk. Formally, we have the following definition.

Definition 2.2.3. *Let* $(\mathsf{Setup.gpar}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme. The advantage of an adversary* $\mathcal{A}$ *against IND-CPA is defined by the experiments* $\mathsf{Exp}^{\mathtt{ind\text{-}cpa\text{-}b}}$ *depicted in Figure 2.2. The encryption scheme is* IND-CPA *if this advantage is negligible in* $\mathfrak{K}$*, for any polynomial-time adversary* $\mathcal{A}$*.*

We recall that the advantage $\mathsf{Adv}^{\mathtt{ind\text{-}cpa\text{-}b}}(\mathcal{A}, \mathfrak{K})$ of the adversary $\mathcal{A}$ against IND-CPA is defined by

$$\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}(\mathcal{A}, \mathfrak{K}) = \left| \Pr\left[\mathsf{Exp}^{\mathtt{ind\text{-}cpa\text{-}1}}(\mathcal{A}, \mathfrak{K}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathtt{ind\text{-}cpa\text{-}0}}(\mathcal{A}, \mathfrak{K}) = 1\right] \right| ,$$

as explained in Section 2.1.2.2.

The ciphertext c^* in Figure 2.2 is called the *challenge ciphertext*, while m_0 and m_1 are called the *challenge plaintexts*. We notice that labels do not play an important role in an IND-CPA encryption scheme: a non-labeled IND-CPA encryption scheme is also IND-CPA for any set of labels Labels (if Enc and Dec simply ignores the label ℓ).

This notion is often too weak, as the adversary may be able to get the decryption of some ciphertexts, when an encryption scheme is used in a larger protocol. IND-CCA ensures that a (challenge) ciphertext does not reveal any information about its plaintext, even if one knows the encryption key ek and even if one has access to an oracle which decrypts ciphertexts of its choice. Obviously, the adversary is not allowed to ask for the decryption of the challenge ciphertext itself, as this would yield to a trivial unavoidable attack. Formally, we have the following definition.

[5] In this thesis, IND-CCA stands for IND-CCA-2. We never consider IND-CCA-1.

IND-CPA	IND-CCA
$\mathsf{Exp}^{\mathtt{ind\text{-}cpa}\text{-}b}(\mathcal{A}, \mathfrak{K})$ $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$ $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{gpar})$ $(\mathsf{st}, \ell^*, \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{ek})$ $\mathsf{c}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \ell^*, \mathsf{m}_b)$ $b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{c}^*)$ **return** b'	$\mathsf{Exp}^{\mathtt{ind\text{-}cca}\text{-}b}(\mathcal{A}, \mathfrak{K})$ $L \leftarrow$ empty list $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$ $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{gpar})$ $(\mathsf{st}, \ell^* \mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}^{\mathsf{ODec}}(\mathsf{ek})$ $\mathsf{c}^* \leftarrow \mathsf{Enc}(\mathsf{ek}, \ell^*, \mathsf{m}_b)$ $b' \leftarrow \mathcal{A}^{\mathsf{ODec}}(\mathsf{st}, \mathsf{c}^*)$ **if** $(\ell^*, \mathsf{c}^*) \in L$ **then return** 0 **else return** b' $\mathsf{ODec}(\ell, \mathsf{c})$ add (ℓ, c) to the list L **return** $\mathsf{Dec}(\mathsf{dk}, \ell, \mathsf{c})$

Figure 2.2: Experiments for Definition 2.2.3 (IND-CPA) and Definition 2.2.4 (IND-CCA)

Definition 2.2.4 (IND-CCA)**.** *Let* (Setup.gpar, KeyGen, Enc, Dec) *be an encryption scheme. The advantage of an adversary $\mathcal{A}$ against IND-CCA is defined by the experiments* $\mathsf{Exp}^{\mathtt{ind\text{-}cca}\text{-}b}$ *depicted in Figure 2.2. The encryption scheme is* IND-CCA *if this advantage is negligible in $\mathfrak{K}$, for any polynomial-time adversary $\mathcal{A}$.*

Contrary to IND-CPA, for IND-CCA, labels play an important role: a non-labeled IND-CCA encryption scheme is not IND-CCA for any set of labels Labels containing at least two distinct labels ℓ_1, ℓ_2: an adversary can just ask for the encryption of two distinct messages $\mathsf{m}_0, \mathsf{m}_1 \in \mathscr{M}$ with label ℓ_1, get the challenge ciphertext c^*, and decrypt it by querying ODec on (ℓ_2, c^*).

In this thesis, by default, IND-CPA encryption schemes are supposed to be non-labeled, while IND-CCA encryption schemes are supposed to be labeled.

2.2.2.2 ElGamal and the Decisional Diffie-Hellman Assumption (DDH)

The ElGamal encryption scheme [ElG85] is one of the most famous IND-CPA encryption scheme. It is defined as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ generates a cyclic group $(p, \mathbb{G}, g)$ of prime order p and outputs the global parameters $\mathsf{gpar} = (p, \mathbb{G}, g)$;[6] we suppose that there exists an efficiently computable and efficiently reversible injective map $\mathcal{G}$ from the public set of messages $\mathscr{M}$ to the group $\mathbb{G}$;

- $\mathsf{KeyGen}(\mathsf{gpar})$ picks a random scalar $z \in \mathbb{Z}_p$, computes $h := g^z \in \mathbb{G}$, and outputs $(\mathsf{ek}, \mathsf{dk}) := (h, z)$;

- $\mathsf{Enc}(\mathsf{ek}, \mathsf{m})$ computes $\mathsf{M} := \mathcal{G}(\mathsf{m})$, picks a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and outputs the ciphertext $\mathsf{c} := (u, v) := (g^r, h^r \cdot \mathsf{M}) \in \mathbb{G}^2$;

[6] Formally, gpar only contains a description of the group. But for the sake of simplicity, we write $\mathsf{gpar} = (p, \mathbb{G}, g)$, throughout this thesis.

- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ computes $\mathsf{m} \leftarrow \mathcal{G}^{-1}(v/u^z)$ and outputs m.

Sometimes we want to directly encrypt messages $\mathsf{m} \in \mathbb{G}$, in that case $\mathcal{G}$ is just the identity and $\mathsf{m} = \mathsf{M}$.

Correctness is straightforward. Proving the security of the ElGamal encryption scheme requires to use some computational assumption. Let us therefore first recall the *decisional Diffie-Hellman* (DDH) assumption.

Assumption 2.2.5 (DDH)**.** *Let $(p, \mathbb{G}, g)$ be a cyclic group of prime order p. The advantage of an adversary $\mathcal{A}$ against the* decisional Diffie-Hellman *(DDH) assumption is defined by:*

$$\mathsf{Adv}^{ddh}(\mathcal{A}, \mathfrak{K}) := \Big| \Pr\left[\mathcal{A}(g, g^z, g^r, g^{zr}) = 1 \mid z, r \xleftarrow{\$} \mathbb{Z}_p\right] - \Pr\left[\mathcal{A}(g, g^z, g^r, g^s) = 1 \mid z, r, s \xleftarrow{\$} \mathbb{Z}_p\right] \Big| \ .$$

The DDH assumption holds when the advantage of any polynomial-time adversary against it is negligible.

Let us immediately introduce a useful definition for later.

Definition 2.2.6. *Let $(p, \mathbb{G}, g)$ be a cyclic group of prime order p. We say that a tuple $(g, g^z, g^r, g^{zs}) \in \mathbb{G}^4$ (or (g^z, g^r, g^{zs}) when g is implicit) is a* DH tuple*, if $r = s$. We also say that a tuple (g^r, g^{zs}) is a* DH tuple in basis (g, g^z)*, if $r = s$.*

We have the following well-known theorem.

Theorem 2.2.7 (IND-CPA security of ElGamal)**.** *The ElGamal encryption scheme is IND-CPA under the DDH assumption. More formally, for any adversary $\mathcal{A}$ against IND-CPA for ElGamal, there exists an adversary $\mathcal{B}$ against DDH with similar running time*[7] *such that:*

$$\mathsf{Adv}^{ind\text{-}cpa}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{ddh}(\mathcal{B}, \mathfrak{K}) \ .$$

2.2.2.3 Cramer-Shoup IND-CCA Encryption Scheme

Unfortunately, ElGamal is not IND-CCA as soon as the set of messages $\mathscr{M}$ contains two distinct messages. Let us show this on the simpler case where $\mathscr{M} = \mathbb{G}$ and $\mathcal{G}$ is the identity function: the adversary just outputs the challenge plaintexts $\mathsf{m}_0 = \mathsf{M}_0 := 1 \in \mathbb{G}$ and $\mathsf{m}_1 = \mathsf{M}_1 := g^2 \in \mathbb{G}$ (for example), then it gets the challenge ciphertext $\mathsf{c}^* = (u^*, v^*)$, and asks to decrypt $(u^*, v^* \cdot g)$ to ODec. If ODec outputs g, the adversary outputs 0, otherwise, it outputs 1. This attack works because (u^*, v^*) is a ciphertext of $\mathsf{m}_0 = 1$ if and only if $(u^*, v^* \cdot g)$ is a ciphertext of g.

In [CS98], Cramer and Shoup proposed the first efficient IND-CCA encryption scheme under DDH. The scheme is defined as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ generates a cyclic group $(p, \mathbb{G}, g)$ of prime order p and picks a collision-resistant hash function $\mathcal{H}$ from a hash family $\mathcal{HF}$.[8] It then outputs the global parameters $\mathsf{gpar} = (p, \mathbb{G}, g, \mathcal{H})$; we suppose that there exists an efficiently computable and efficiently reversible injective map $\mathcal{G}$ from the public set of messages $\mathscr{M}$ to the group $\mathbb{G}$;

[7] This notion is informal and means that the difference of running time between the two adversaries is reasonable: here it basically consists in a constant number of group operations. We could just have said that $\mathcal{B}$ is polynomial-time if $\mathcal{A}$ is, but this is slightly less precise, in particular if we are interested in the tightness of the security proof.

[8] A universal one-way hash function is sufficient when there is no label. But we do not try to optimize this.

- $\mathsf{KeyGen}(\mathsf{gpar})$ picks two generators $g_1, g_2 \xleftarrow{\$} \mathbb{G}^*$, a tuple of five random scalars $\mathsf{dk} := (x_1, x_2, y_1, y_2, z) \in \mathbb{Z}_p^5$, computes $c \leftarrow g_1^{x_1} g_2^{x_2}$, $d \leftarrow g_1^{y_1} g_2^{y_2}$, and $h \leftarrow g_1^z$. It sets the encryption key to $\mathsf{ek} := (g_1, g_2, c, d, h)$. It finally outputs $(\mathsf{ek}, \mathsf{dk})$.

- $\mathsf{Enc}(\ell, \mathsf{ek}, \mathsf{m})$ computes $\mathsf{M} := \mathcal{G}(\mathsf{m})$, picks a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and outputs the ciphertext
$$\mathsf{c} := (\ell,\ u_1 := g_1^r,\ u_2 := g_2^r,\ v := h^r \cdot \mathsf{M},\ w := (cd^\xi)^r)\,,$$
where w is computed after $\xi := \mathcal{H}(\ell, u_1, u_2, v)$;

- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ first computes $\xi = \mathcal{H}(\ell, u_1, u_2, v)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} w$. If the equality holds, it computes $\mathsf{m} \leftarrow \mathcal{G}^{-1}(v/u_1^z)$ and outputs m. Otherwise, it outputs $\perp$.

This scheme is IND-CCA secure under DDH [CS98; CS02].

2.2.2.4 ElGamal and Cramer-Shoup Encryption for Vectors

ElGamal and Cramer-Shoup encryption schemes enable to encrypt one group element. In our constructions, we often need to encrypt several group elements, or in other words, a vector of group elements.

Vector encryption without randomness reuse. The first naive solution consists in encrypting each element of the vector independently and to concatenate the ciphertexts. A ciphertext for a vector of n group elements consists of n ciphertexts. This perfectly works for ElGamal: the resulting scheme is clearly IND-CPA. More generally, this would work for any IND-CPA scheme.

Unfortunately, for Cramer-Shoup, the resulting scheme is no more IND-CCA: the adversary can indeed mix and match the group elements inside the vectors and use the decryption oracle to break IND-CCA. Fortunately, there is a very efficient way to solve this issue: use the same value ξ for all the ciphertexts, as a hash over all the parts which do not depend on ξ. Concretely, to encrypt a vector $(\mathsf{M}_1, \ldots, \mathsf{M}_n) \in \mathbb{G}^n$ with label ℓ, we pick random scalars $r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$ and output the ciphertext:

$$\mathsf{c} := (\ell,\ (u_{i,1} := g_1^{r_i},\ u_{i,2} := g_2^{r_i},\ v_i := h^{r_i} \cdot \mathsf{M}_i,\ w_i := (cd^\xi)^{r_i})_{i=1,\ldots,n})\,,$$

where the group elements w_i are computed after:

$$\xi := \mathcal{H}(\ell, (u_{i,1}, u_{i,2}, v_i)_{i=1,\ldots,n})\ .$$

This scheme is IND-CCA under DDH. This is a folklore result. A formal proof (of a slightly stronger result) can be found in [BBC+13a].

Vector encryption with randomness reuse. The previous vector encryption schemes did not increase the size of the encryption key: only the ciphertext size was increased. If we are willing to increase the size of the encryption key, we can reduce the size of the ciphertext using randomness reuse [Kur02; BBS03].

Let us first state the ElGamal encryption scheme with randomness reuse for vectors of n group elements:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ is the same as for classical ElGamal;
- $\mathsf{KeyGen}(\mathsf{gpar})$ picks a tuple of n independent random scalars $\mathsf{dk} := (z_1, \ldots, z_n) \xleftarrow{\$} \mathbb{Z}_p^n$, computes the encryption key $\mathsf{ek} := (h_1 := g^{z_1}, \ldots, h_n := g^{z_n}) \in \mathbb{G}^n$, and outputs $(\mathsf{ek}, \mathsf{dk})$;
- $\mathsf{Enc}(\mathsf{ek}, (\mathsf{M}_1, \ldots, \mathsf{M}_n))$ picks a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the ciphertext:

$$\mathsf{c} := (u = g^r,\ v_1 := h_1^r \cdot \mathsf{M}_1,\ \ldots,\ v_n := h_n^r \cdot \mathsf{M}_n) \in \mathbb{G}^{n+1}\,;$$

- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ computes $\mathsf{M}_i = v_i / u_1^{z_i}$ for $i \in \{1, \ldots, n\}$ and outputs $(\mathsf{M}_1, \ldots, \mathsf{M}_n)$.

This scheme is IND-CPA under DDH [Kur02; BBS03].

Let us now show the Cramer-Shoup encryption scheme with randomness reuse for vectors of n group elements:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ is the same as for classical Cramer-Shoup;
- $\mathsf{KeyGen}(\mathsf{gpar})$ picks two generators $g_1, g_2 \xleftarrow{\$} \mathbb{G}^*$, $n+4$ random scalars $\mathsf{dk} := (x_1, x_2, y_1, y_2, z_1, \ldots, z_n) \in \mathbb{Z}_p^5$, computes $c := g_1^{x_1} g_2^{x_2}$, $d := g_1^{y_1} g_2^{y_2}$, and $h_1 := g_1^{z_1}, \ldots, h_n := g_1^{z_n}$. It sets the encryption key to $\mathsf{ek} := (g_1, g_2, c, d, h_1, \ldots, h_n)$. It finally outputs $(\mathsf{ek}, \mathsf{dk})$.
- $\mathsf{Enc}(\ell, \mathsf{ek}, \mathsf{m})$ picks a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and outputs the ciphertext

$$\mathsf{c} := (\ell,\ u_1 := g_1^r,\ u_2 := g_2^r,\ v_1 := h_1^r \cdot \mathsf{M}_1,\ \ldots,\ v_n := h_n^r \cdot \mathsf{M}_n,\ w := (cd^{\xi})^r)\,,$$

  where the group element w is computed after $\xi := \mathcal{H}(\ell, u_1, u_2, v_1, \ldots, v_n)$;
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ first computes $\xi = \mathcal{H}(\ell, u_1, u_2, v_1, \ldots, v_n)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} w$. If the equality holds, it computes $\mathsf{M}_i = v_i / u_1^{z_i}$ for $i \in \{1, \ldots, n\}$ and outputs $(\mathsf{M}_1, \ldots, \mathsf{M}_n)$. Otherwise, it outputs $\perp$.

This scheme is IND-CCA under DDH [Kur02; BBS03].

2.2.2.5 Variants of ElGamal and Cramer-Shoup

ElGamal and Cramer-Shoup encryption schemes can be slightly changed to be based on variants of the DDH assumptions, including the *decisional linear* (DLin) assumption [BBS04] or the *matrix decisional Diffie-Hellman* (MDDH) family of assumptions introduced by Escala et al. in [EHK+13]. All our constructions can also be easily extended to these cases. We even have done it explicitly in some of our papers, e.g. [BBC+13a; ABP15c]. However, for the sake of simplicity, we focus on classical ElGamal and Cramer-Shoup in this thesis.

2.2.3 Randomness Extractors and Min Entropy

A randomness extractor enables to extract from some "sufficiently random" variable, a uniform (or almost uniform) bit string of some given length. By "sufficiently random", we mean that it is hard to guess the value of this random variable. More precisely, let us first define the notion of min entropy which formally characterizes this notion.

Definition 2.2.8. *Let X be a random variable with some finite range $\mathcal{S}$. The* min entropy *of X is defined as:*

$$-\log \max_{x \in \mathcal{S}} \Pr\left[\, X = x \,\right] \quad .$$

Then we can define a randomness extractor.

Definition 2.2.9. *A* randomness extractor *for a distribution ensemble $(\mathcal{D}_{\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$ over some some finite sets $(\mathcal{S}_{\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$ is a family of functions $(\mathsf{Ext}_{\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$, where $\mathsf{Ext}_{\mathfrak{K}}$ is a function from $\{0,1\}^{n(\mathfrak{K})} \times \mathcal{S}_{\mathfrak{K}}$ to $\{0,1\}^{m(\mathfrak{K})}$, for some integers $n(\mathfrak{K})$ and $m(\mathfrak{K})$, such that the following two distributions are statistically indistinguishable:*

$$\left\{ (\mathsf{seed}, \mathsf{Ext}(\mathsf{seed}, x)) \mid \mathsf{seed} \xleftarrow{\$} \{0,1\}^{n(\mathfrak{K})}; x \xleftarrow{\$} \mathcal{D}_{\mathfrak{K}} \right\}$$
$$\left\{ (\mathsf{seed}, u) \mid \mathsf{seed} \xleftarrow{\$} \{0,1\}^{n(\mathfrak{K})}; u \xleftarrow{\$} \{0,1\}^{m(\mathfrak{K})} \right\} \ .$$

We insist on the fact that seed *and x are chosen independently of each other, in the first distribution.*

In the sequel, as usual, the security parameter $\mathfrak{K}$ is often omitted.

The value seed is called a seed. Extractors without seed (i.e., for which $n = 0$), are called *deterministic* extractors. For some distributions, there exist deterministic extractors, for example for uniform points of some elliptic curves [CFPZ09].

In most cases, we do not know of any deterministic extractor. Fortunately, the leftover hash lemma enables the construction of an extractor for any distribution with min entropy about at least $3\mathfrak{K}$. More precisely, we have the following theorem [HILL99]:

Theorem 2.2.10. *Let $(\mathcal{D}_{\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$ be a family of distributions over some some finite sets $(\mathcal{S}_{\mathfrak{K}})_{\mathfrak{K}\in\mathbb{N}}$. We suppose that elements of $\mathcal{S}_{\mathfrak{K}}$ can be represented by strings of size at most $k(\mathfrak{K})$, such that $k(\mathfrak{K})$ is polynomial in $\mathfrak{K}$. We then suppose without loss of generality that $\mathcal{S}_{\mathfrak{K}} = \{0,1\}^{k(\mathfrak{K})}$. For any polynomial function m in $\mathfrak{K}$, we define a randomness extractor $\mathsf{Ext}_{\mathfrak{K}}$, by $\mathsf{Ext}_{\mathfrak{K}}(\mathsf{seed}, x) = y \in \{0,1\}^{m(\mathfrak{K})}$ where:*

$$y_i = \sum_{j=1}^{k(\mathfrak{K})} \mathsf{seed}_{(i-1)\cdot k(\mathfrak{K})+j} \cdot x_i \qquad \text{for } i \in \{1, \ldots, m(\mathfrak{K})\} \,,$$

where $\mathsf{seed} \in \{0,1\}^{n(\mathfrak{K})}$, $x \in \{0,1\}^{k(\mathfrak{K})}$, and $n(\mathfrak{K}) = k(\mathfrak{K}) \cdot m(\mathfrak{K})$. If the min entropy of $\mathcal{D}_{\mathfrak{K}}$ is $\beta(\mathfrak{K})$, then the two distributions in Definition 2.2.9 are $2^{(m(\mathfrak{K})-\beta(\mathfrak{K}))/2}$-close. This randomness extractor is secure as soon as $m(\mathfrak{K}) - \beta(\mathfrak{K}) = \Omega(\mathfrak{K})$.

Concretely, this means that we get an extractor with output size $m(\mathfrak{K}) = \mathfrak{K}$ (and the two distributions in Definition 2.2.9 are $2^{-\mathfrak{K}}$-close), if $\mathcal{D}_{\mathfrak{K}}$ has min entropy at least $3\mathfrak{K} + 2$.

This theorem works as soon as $(\mathsf{Ext}_{\mathfrak{K}}(\mathsf{seed}, \cdot))_{\mathsf{seed}}$ is a universal hash function family. But we just need the existence of an extractor for this thesis and do not need to formally define universal hash function families.

2.3 Languages

The notion of language is used throughout this thesis: not only for the central cryptographic primitive of this thesis *projective hash functions* (PHFs) and the central notions of *diverse*

modules (DMs) and *diverse vector spaces* (DVSs); but also for many applications to zero-knowledge arguments.

We start by defining (NP) languages for *projective hash functions* (PHFs), and some specific languages for PHF called hard-subset-membership languages. We then show how languages for zero-knowledge arguments differ from languages for PHF. Differences are purely syntactical.

2.3.1 Languages for Projective Hash Functions

We consider a family of NP languages $(\mathscr{L}_{\mathsf{lpar}})_{\mathsf{lpar}}$, indexed by some parameter lpar, with witness relation $\mathscr{R}_{\mathsf{lpar}}$, namely

$$\mathscr{L}_{\mathsf{lpar}} = \{\chi \in \mathcal{X}_{\mathsf{lpar}} \mid \exists w,\ \mathscr{R}_{\mathsf{lpar}}(\chi, w) = 1\}\,,$$

where $(\mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}$ is a family of sets. The *(language) parameters* lpar are generated by a polynomial-time algorithm $\mathsf{Setup.lpar}$ which takes as input some *global parameters* gpar. These global parameters are themselves generated by a polynomial-time algorithm $\mathsf{Setup.gpar}$ which takes as input a unary representation of the security parameter $\mathfrak{K}$. They are always supposed to be (implicitly) included in lpar. We suppose that membership in $\mathcal{X}_{\mathsf{lpar}}$ and $\mathscr{R}_{\mathsf{lpar}}$ can be checked in polynomial-time in $\mathfrak{K}$. More precisely, there exist a polynomial-time algorithm taking as input (lpar, χ) and outputting 1 if $\chi \in \mathcal{X}_{\mathsf{lpar}}$, and 0 otherwise; and another polynomial-time algorithm taking as input (lpar, χ, w) and outputting $\mathscr{R}_{\mathsf{lpar}}(\chi, w)$. In particular, words χ in $\mathcal{X}_{\mathsf{lpar}}$ and witnesses w have polynomial length in $\mathfrak{K}$.

Language trapdoor ltrap for lpar. We suppose that $\mathsf{Setup.lpar}$ also outputs a (language) trapdoor ltrap associated to lpar. This trapdoor is empty $\perp$ in most cases, but for some applications in Sections 5.2, 6.2 and 6.3, we require that ltrap contains enough information to decide whether a word $\chi \in \mathcal{X}$ is in $\mathscr{L}$ or not (or slightly more information). We notice that for most languages (we are interested in), it is easy to make $\mathsf{Setup.lpar}$ output such a trapdoor, without changing the distribution of lpar.

When the trapdoor ltrap is used, we write $(\mathsf{lpar}, \mathsf{ltrap}) \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$. When it is not used, we just write $\mathsf{lpar} \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$.

Example 2.3.1 (DDH language)**.** *The algorithm* $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ *generates global parameters* $\mathsf{gpar} = (p, \mathbb{G}, g)$ *where* $\mathbb{G}$ *is a cyclic group* $\mathbb{G}$ *of prime order* p *and generated by* g*. The algorithm* $\mathsf{Setup.lpar}(\mathsf{gpar})$ *picks a random integer* z *in* $\mathbb{Z}_p^*$ *and outputs* $\mathsf{lpar} = (\mathsf{gpar}, g, h)$ *with* $h = g^z$*, and* $\mathsf{ltrap} = z$*. To simplify notation, we write* $\mathsf{gpar} = (p, \mathbb{G}, g)$ *and* $\mathsf{lpar} = (g, h)$*, where* $\mathbb{G}$ *stands for a description of* $\mathbb{G}$ *and where we do not explicitly write that* lpar *contains* gpar*.*

Then the DDH language in basis (g, h) *is defined by* $\mathcal{X} = \mathbb{G}^2$ *and:*

$$\mathscr{R}(\chi, w) = 1 \quad \Longleftrightarrow \quad (u, v) = (g^r, h^r) \quad \text{with } \chi = (u, v) \in \mathcal{X} \text{ and } w = r \in \mathbb{Z}_p\ .$$

This implies that:

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (g^r, h^r)\}\ . \tag{2.1}$$

We remark that using ltrap*, we efficiently check whether a word* (u, v) *is in the language or not: for all* $(u, v) \in \mathbb{G}^2$*,*

$$(u, v) \in \mathscr{L} \quad \Longleftrightarrow \quad v = u^z\ .$$

If we do not require that check to be possible in polynomial time, we can just set $\mathsf{ltrap} = \perp$*.*

Global parameters gpar **and language parameters** lpar**.** As shown in the example, global parameters gpar usually correspond to the description of the groups involved in the construction, while lpar is what actually defines the language. Both are always *public*. The reason why we separate gpar and lpar is that in the sequel, we will consider combinations (conjunctions and disjunctions for example) of languages over the same groups or related groups, and therefore we need to consider different parameters lpar corresponding to the same global parameters gpar.

More generally, in this whole thesis, we suppose that global parameters are common to all the primitives we consider.

Notation simplification. In the sequel, in most cases, the global parameters gpar and the language parameters lpar are often omitted to simplify notation. Furthermore, when we talk about a family of languages $(\mathscr{L}_{\mathsf{lpar}})$, we implicitly also consider all the algorithms described above. We even often just call the family of languages $(\mathscr{L}_{\mathsf{lpar}})$ or $(\mathscr{L}_{\mathsf{lpar}} \subseteq \mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}$, "the language $\mathscr{L} \subset \mathcal{X}$". The fact that we are dealing with a family of languages instead of a simple language, and the parameters lpar are implicit.

Finally, we say "for any lpar" to mean: for any security parameter $\mathfrak{K}$, any global parameters $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(\mathfrak{K})$, and any parameters $\mathsf{lpar} \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$. In addition, it is often sufficient that everything holds with overwhelming probability instead of for any lpar. Finally, in the sequel, we often define a language just by giving an equation like Equation (2.1), from which the set $\mathcal{X}$ and the witness relation $\mathscr{R}$ can be easily deduced.

Historical note 2.3.2. *There has been a large variety of formalizations of (family of) languages for hash proof systems. The formalization we adopt in this thesis is inspired from the one in [BBC+13c; ABP15c]. We tried to make it as expressive and formal as possible, while keeping it relatively simple. Compared to [BBC+13c], for example, we have added the notion of global parameters* gpar*, used to enable combination of languages (we only enable to combine language with the same global parameters) but we removed the notion of private parameter* aux*. We basically only need this private parameter for the constructions of* password authenticated key exchange *(PAKE), and in this case, we can just put it in the word χ itself, and add a specific requirement on the PHF. Furthermore, as we do not focus on PAKE in this thesis, avoiding this extra private parameter improves the readability.*

2.3.2 Hard-Subset-Membership Languages

In our applications in Chapter 6, we often need to use specific languages which are called hard-subset-membership languages. Informally, a hard subset membership language is a language for which it hard to say whether an element is inside the language or not. We should point out that, except for hard-subset-membership languages, we do not require that $\mathscr{L}$ and $\mathcal{X}$ are efficiently samplable.

Formally, we have the following definition:

Definition 2.3.3. *A* hard-subset-membership *(family of) languages $(\mathscr{L}_{\mathsf{lpar}} \subseteq \mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}$ is a family of languages as defined in Section 2.3 with the following additional properties*

- $\mathscr{R}$-samplability. *There exists a polynomial-time algorithm which takes as input a parameter* lpar *and randomly sample words χ from $\mathscr{L}_{\mathsf{lpar}}$ together with a valid witness w, according to some distribution (which might not be uniform). We write $(\chi, w) \xleftarrow{\$} \mathscr{R}_{\mathsf{lpar}}$*

Hard Subset Membership

$\mathsf{Exp}^{\texttt{sub-memb-}b}(\mathcal{A}, \mathfrak{K})$
  $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
  $(\mathsf{lpar}, \mathsf{ltrap}) \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$
  if $b = 0$ **then**
    $(\chi, w) \xleftarrow{\$} \mathscr{R}_{\mathsf{lpar}}$
  else
    $\chi \xleftarrow{\$} \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$
  return $\mathcal{A}(\mathsf{lpar}, \chi)$

Figure 2.3: Experiments for Definition 2.3.3 (hard subset membership)

to say that χ and w are sampled by this algorithm. We also write $\chi \xleftarrow{\$} \mathscr{L}_{\mathsf{lpar}}$ when we do not care about w. We suppose that for any lpar *and any pair (χ, w) sampled by this algorithm, we have $\mathscr{R}_{\mathsf{lpar}}(\chi, w) = 1$.*

- $(\mathcal{X} \setminus \mathscr{L})$-samplability. *There exists a polynomial-time algorithm which takes as input a parameter* lpar *and randomly sample words χ form $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$, according to some distribution (which might not be uniform). We write $\chi \xleftarrow{\$} \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ to say that χ is sampled via this algorithm. We suppose that for any* lpar *and any χ sampled by this algorithm, we have $\chi \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$.*

- hard-subset-membership. *Randomly sampled words χ from $\mathscr{L}_{\mathsf{lpar}}$ and from $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ (by the two previous algorithms respectively) are computationally indistinguishable, without knowing* ltrap. *Formally, the advantage of an adversary $\mathcal{A}$ against subset-membership is defined by the experiments $\mathsf{Exp}^{\texttt{sub-memb-}b}$ depicted in Figure 2.3. The language is* hard-subset-membership, *if this advantage is negligible, for any polynomial-time adversary $\mathcal{A}$.*

Remark 2.3.4. *There are many ways to define hard-subset-membership languages. Here, as we do not enforce samplability of general languages, we need to incorporate it in the definition of hard-subset-membership languages. Furthermore, there are two ways to define the hard-subset-membership property itself: distinguishing random words from $\mathscr{L}$ from random words from $\mathcal{X}$ or from $\mathcal{X} \setminus \mathscr{L}$. In the first case, we often require that the probability for a random word from $\mathcal{X}$ to be in $\mathscr{L}$ is negligible. In all our applications, it is simpler to use the second definition. If we want tighter reductions however, we might want to prefer the first definition, for which a lot of languages (such as DDH, defined below) yield a random self-reducible hard-subset-membership property.*

Example 2.3.5 (DDH assumption)**.** *The hard-subset-membership property of the DDH language as defined in Example 2.3.1 almost corresponds to the DDH assumption in Assumption 2.2.5, when:*

- *$(\chi, w) \xleftarrow{\$} \mathscr{R}_{\mathsf{lpar}}$ is sampled as follows: $w := r \xleftarrow{\$} \mathbb{Z}_p$ and $\chi := (u, v) := (g^r, h^r)$;*

- *$\chi \xleftarrow{\$} \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ is sampled as follows: $r \xleftarrow{\$} \mathbb{Z}_p$, $s \xleftarrow{\$} \mathbb{Z}_p \setminus \{r\}$, and $(u, v) := (g^r, h^s)$.*

The only difference is that, in the DDH assumption, in the second case, s is completely independent of r instead of being forced to be distinct from r. This does not matter as when $r, s \xleftarrow{\$} \mathbb{Z}_p$, the probability that $r = s$ is only $1/p$ which is negligible.

2.3.3 Language for Zero-Knowledge Arguments

Languages for zero-knowledge arguments are defined similarly to languages for PHF. To easily distinguish them in applications from languages for PHF, we add two dots: $\ddot{\chi}$, $\ddot{w}$, $\ddot{\mathsf{lpar}}$, $\ddot{\mathscr{L}}$, $\ddot{\mathscr{R}}$, $\ddot{\mathcal{X}}$, $\ddot{\mathsf{Setup.lpar}}$ instead of χ, w, lpar, $\mathscr{L}$, $\mathscr{R}$, $\mathcal{X}$, $\mathsf{Setup.lpar}$. This notation is also used for the language of other primitives such as *implicit zero-knowledge argument* (iZK) in Section 6.3. Global parameters gpar are the same in all our applications.

In zero-knowledge arguments, we sometimes want to be able to extract part of the witness. For that purpose, we often (but not always) split the witness for language for zero-knowledge arguments in two parts: a part $\ddot{w}_{\text{Ꙗ}}$ which we want to prove knowledge of and so which can be extracted, and a part $\ddot{w}_{\exists}$ for which we only want to prove existence: $\ddot{w} = (\ddot{w}_{\text{Ꙗ}}, \ddot{w}_{\exists})$. Concretely, languages are often written as follows:

$$\ddot{\mathscr{L}} = \{\ddot{\chi} \mid \text{Ꙗ}\ddot{w}_{\text{Ꙗ}}, \exists\ddot{w}_{\exists}, \ddot{\mathscr{R}}(\ddot{\chi}, (\ddot{w}_{\text{Ꙗ}}, \ddot{w}_{\exists})) = 1\} \subseteq \ddot{\mathcal{X}}\,,$$

where the symbol Ꙗ is used instead of $\exists$ to indicate the extractable part of the witness. When there is no extractable part, we either write $\ddot{w}_{\text{Ꙗ}} = \perp$ or simply use $w = \ddot{w}_{\exists}$.

2.4 Zero-Knowledge Arguments

Let us now introduce zero-knowledge arguments. We start by an informal overview of them, before giving more formal definitions.

For the reader familiar with zero-knowledge arguments, we only consider non-rewinding simulators and extractors in the *common reference string* (CRS) model. This is useful when such proofs are used in concurrent settings or as building blocks for protocols in the *universal composability* (UC) framework [Can01]. Furthermore, our arguments are in the quasi-adaptive setting [JR13], in which the CRS depends on the language.

The same CRS can still be used a polynomial number of times to run the same protocols between the same parties or different parties. In other words, the CRS is supposed to be *re-usable*.

Reading note 2.4.1. *Formal definitions of zero-knowledge arguments are quite complex due to interactivity. Most of the paper should be understandable without a careful reading of Section 2.4. Section 2.4.3 is much easier to follow, as it focuses on the non-interactive case.*

2.4.1 Overview

2.4.1.1 Zero-Knowledge Arguments and Proofs

A *zero-knowledge argument* or *zero-knowledge proof* is a protocol enabling some prover to prove that a *word* or *statement* $\ddot{\chi}$ is in a language $\ddot{\mathscr{L}}$. Informally, such a protocol has to satisfy three properties:

- *Completeness.* An honest verifier always accepts a proof made by an honest prover for a valid word and using a valid witness.

- *Soundness.* No adversary can make an honest verifier accept a proof of a word $\ddot{\chi} \notin \ddot{\mathscr{L}}$, either statistically (for *zero-knowledge proofs*) or computationally (for *zero-knowledge arguments*).
- *Zero-knowledge.* It is possible to simulate (in polynomial-time) the interaction between a (potentially malicious) verifier and an honest prover for any word $\ddot{\chi} \in \ddot{\mathscr{L}}$ without knowing a witness $\ddot{w}$.

Common reference string and rewinding. In this thesis, we are always in the *common reference string* (CRS) model, where a trusted setup ZK.Setup generates some CRS crs. For the simulation in the zero-knowledge property, the simulator generates himself the CRS crs and can therefore add a trapdoor trap to it which enables him the simulation.

The implicit global parameters gpar (which often contain a description of the group used, in case of use of cyclic groups of bilinear groups) are common to the argument and the language.

Quasi-adaptivity. We are in the *quasi-adaptive* model introduced by Jutla and Roy [JR13], which allows the CRS crs to depend on the parameters $\ddot{\mathsf{lpar}}$ of the language. We suppose that crs always implicitly contains $\ddot{\mathsf{lpar}}$ and gpar. The soundness property still holds adaptively.

This quasi-adaptive model is sufficient for many applications as explained in [JR13].

Witness samplability. Our most efficient constructions assume that the setup $\ddot{\mathsf{Setup}}$.lpar outputs a trapdoor $\ddot{\mathsf{ltrap}}$ which satisfies some additional properties detailed when needed. This slightly restricts the languages which can be considered.

We note however that $\ddot{\mathsf{ltrap}}$ is never used in the protocols, but only in the proof of the soundness and zero-knowledge property.

Argument versus proof. The only difference between argument and proof is the fact the latter requires statistical soundness.

Tag. We consider arguments with tags, which are similar to labels from encryption scheme. As labels are not useful for an IND-CPA encryption scheme, tags are not useful for a classical zero-knowledge argument. However, they can be quite handy, for stronger variants such as simulation-sound zero-knowledge arguments (described below). Later, in Remark 2.4.2, we explain why we use the term "tag" instead of "label" for zero-knowledge arguments.

2.4.1.2 Variants

(Partial) extractability. As already mentioned in Section 2.3.3, we sometimes consider another property: *(partial) extractability* which states that there exists an extractor able to simulate a verifier and output a valid *partial witness* $\ddot{w}_{\text{Я}}$ from any successful interaction with a polynomial-time adversary playing the role of a prover, on some word $\ddot{\chi}$. By valid partial witness, we mean that there exists $\ddot{w}_{\exists}$ such that $\mathscr{R}(\ddot{\chi}, (\ddot{w}_{\text{Я}}, \ddot{w}_{\exists})) = 1$.

We consider non-rewinding extractors that use a trapdoor in the CRS crs similarly to the zero-knowledge simulators we consider.

In case $\ddot{w}_{\text{Я}}$ is not used ($\ddot{w}_{\text{Я}} = \perp$), partial extractability is equivalent to soundness.

Honest-verifier zero-knowledge. For some applications, it suffices to consider a weaker version of the zero-knowledge property, called *honest-verifier zero-knowledge*, which only needs to hold with respect to honest verifiers.

Simulation-extractability/soundness. *Simulation-extractibility* is a stronger property than extractibility. It states that extractability has to hold even if the adversary can ask for simulated proofs of words of his choice (not necessarily in $\ddot{\mathscr{L}}$). When $\ddot{w}_{\mathcal{H}}$ is not used, this property is also called *simulation-soundness*.

Obviously, we have to add some restrictions on the proofs made by the adversary (from which the extractor finds a partial witness). In this thesis, we use a tag-based approached: proofs made by the adversary have to be done with a tag different from all the simulated proofs. Except for this restriction, the adversary can adaptively choose the tags of the simulated proofs. This is similar to the definition used by Kiltz and Wee in [KW15] for simulation-sound non-interactive zero-knowledge arguments.

If the set of tags is exponential in the security parameter, we can transform this notion of simulation-extractability into a more classical (stronger) one where the only restriction is that the transcripts of the proofs made by the adversary is different (or do not match with) any transcript of a simulated proof, using a one-time signature: the tag is a fresh public key for the one-time signature scheme and the prover signs the whole transcript at the end and sends the signature to the verifier. If we only want the pair tag-word of the proof made by the adversary to be different from the pairs tag-word (tag, χ) of the simulated proofs, we just need to replace the tag by a hash value of the original tag and the word χ, using a collision-resistant hash function.

Remark 2.4.2. *We use the term "tag" and not "label" as for IND-CCA labeled encryption schemes, because the proof made by the adversary has to be for a different tag than the simulated proofs, even if the transcripts are different. Instead, for IND-CCA, only queries with the same label-ciphertext pair (ℓ, c) as the challenge label-ciphertext pair (ℓ^*, c^*) are forbideen.*

Witness indistinguishability. *Witness indistinguishability* states that an adversary cannot distinguish a proof made using one witness w_1 or another witness w_2 for the same word $\ddot{\chi}$ (and the same tag tag). It is clearly weaker than zero-knowledge. We do not define this property formally, as we just use it to show that some schemes are not zero-knowledge by showing that they are not witness indistinguishable.

2.4.1.3 Non-Interactive Zero-Knowledge Arguments (NIZK)

An important particular case of zero-knowledge arguments are *non-interactive zero-knowledge arguments* (NIZK). In a NIZK, the prover just sends one flow to the verifier, called the *proof* and denoted π. The verifier can then directly check it.

Contrary to zero-knowledge arguments, which can be constructed without using a CRS, NIZK (for non-trivial languages) cannot. NIZK can be used in many more applications than interactive zero-knowledge arguments, but are usually more complex to construct and less efficient.

2.4.2 Formal Definitions of Zero-Knowledge Arguments

In this section, we give formal definitions for partially-extractable zero-knowledge arguments, using the formalism of Garay-MacKenzie-Yang (GMY) in [GMY06]. We use uniform adversaries and non-rewinding extractors and simulators, without auxiliary information. This not only simplifies definitions but also make them more useful when such proofs are used in concurrent settings or as building blocks in protocols in the UC framework [Can01].

2.4.2.1 GMY Formalism

For two ITMs A and B, $\langle A(\mathsf{priv}_A), B(\mathsf{priv}_B)\rangle_{\mathsf{crs}}(\mathsf{in})$ is the local output of B after an interactive execution with A using CRS crs, common input in, private inputs priv_A and priv_B for A and B respectively.[9] The transcript tr of a machine is a tuple composed of its common input in, the messages received on its communication input tape and the messages sent through its communication output tape.

For any ITM A, we also denote by $\boxed{A}$ its multi-session extension or protocol wrapper. $\boxed{A}$ works as follows:

- on input message $(\mathsf{START}, \mathsf{sid}, \mathsf{in}, \mathsf{priv})$, $\boxed{A}$ starts a new interactive machine A with session id sid (distinct from all the other session ids), common input in, private input priv and a fresh random tape;
- on input message $(\mathsf{MSG}, \mathsf{sid}, m)$, $\boxed{A}$ sends the message m to the interactive machine with session id sid (if it exists), and returns the output message of this machine.

All machines A started by $\boxed{A}$ use the same CRS crs.

Let $\boxed{A}_1$ be the single-session extension of A, which works as $\boxed{A}$, except it only accepts one START query. The output of $\boxed{A}_1$ is the tuple $(\mathsf{in}, \mathsf{tr}, v)$ where in is the common input, tr is the transcript of the machine A started by $\boxed{A}_1$ and v is the output of A. The output of $\boxed{A}$ is a tuple $(\mathbf{in}, \mathbf{tr}, \boldsymbol{v})$ of three vectors, such that $(\mathsf{in}_i, \mathsf{tr}_i, v_i)$ is the tuple that $\boxed{A}_1$ would have output for the i-th machine started by $\boxed{A}$.

Two ITMs B and C are said to be *coordinated* if they have a single control (and, in particular, a common state), but two distinct sets of input/output communication tapes. For four interactive Turing machines A, B, C and D, with B and C coordinated, $(\langle A, B\rangle, \langle C, D\rangle)_{\mathsf{crs}}$ is the local output of D after an interactive execution with C and an interactive execution between A and B, all using the CRS crs.

2.4.2.2 (Partially) Extractable Zero-Knowledge Arguments

Definition 2.4.3. *A* (labeled) (partially) extractable zero-knowledge argument *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is a tuple* $\mathsf{ZK} = (\mathsf{ZK.Setup}, \mathsf{ZK.Prove}, \mathsf{ZK.Ver}, \mathsf{ZK.Sim} = (\mathsf{ZK.Sim}_1, \mathsf{ZK.Sim}_2), \mathsf{ZK.Ext} = (\mathsf{ZK.Ext}_1, \mathsf{ZK.Ext}_2))$*, where:*

- $\mathsf{ZK.Setup}$ *is a polynomial-time algorithm which takes the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$ *as input and outputs a CRS* crs *which is implicitly given as common input of all the other algorithms (except for* $\mathsf{ZK.Sim}_1$*);* crs *is supposed to implicitly contain* gpar *and* $\ddot{\mathsf{lpar}}$*;*

[9] The input tape is therefore separated in three parts: CRS, common input, and private input. This separation is convenient to simplify notation.

- $\mathsf{ZK.Prove}$ *is a polynomial-time ITM which takes a tag* tag *(in some set* Tags*) and a word* $\ddot{\chi} \in \ddot{\mathscr{L}}$ *as common input and a valid witness* $(\ddot{w}_{\aleph}, \ddot{w}_{\exists})$ *as private input (i.e.,* $\mathscr{R}(\ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) = 1$*) and is able to run a protocol (with a verifier* $\mathsf{ZK.Ver}$*) to prove that* $\ddot{\chi} \in \ddot{\mathscr{L}}$*;*

- $\mathsf{ZK.Ver}$ *is a polynomial-time ITM which takes a tag* tag *and a word* $\ddot{\chi}$ *as common input, is able to run a protocol (with a prover* $\mathsf{ZK.Prove}$*) and outputs* 1 *if it accepts the proof of the prover and* 0 *otherwise;*

- $\mathsf{ZK.Sim}_1$ *is a polynomial-time algorithm which takes the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$ *as input and outputs a* simulated *(a.k.a.,* fake*) CRS* crs *together with a trapdoor* trap*;*

- $\mathsf{ZK.Sim}_2$ *is a polynomial-time ITM which takes the trapdoor* trap *as private input and a tag* tag *with a word* $\ddot{\chi}$ *as common input, and is able to simulate a run of* $\mathsf{ZK.Prove}$ *(without knowing* $\ddot{w}_{\aleph}$ *nor* $\ddot{w}_{\exists}$*);*

- $\mathsf{ZK.Ext}_1$ *is a polynomial-time algorithm which takes the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$ *as input and outputs an* extraction *(a.k.a.,* fake*) CRS* crs *together with a trapdoor* trap*;*

- $\mathsf{ZK.Ext}_2$ *is a polynomial-time ITM which takes as private input the trapdoor* trap *and as common input a tag* tag *and a word* $\ddot{\chi}$*, and is able to simulate a run of* $\mathsf{ZK.Ver}$ *in such a way that, if* $\mathsf{ZK.Ver}$ *accepts, it is able to extract a valid partial witness* $\ddot{w}_{\aleph}$ *for* $\ddot{\chi}$*.* $\mathsf{ZK.Ext}$ *outputs a pair* $(b, \ddot{w}_{\aleph})$ *where* b *indicates if* $\mathsf{ZK.Ver}$ *accepts and* $\ddot{w}_{\aleph}$ *is a partial witness;*

such that the following properties are verified:

- Completeness. ZK *is* ε*-complete, if for all global parameters* $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$*, for all language parameters* $\mathsf{lpar} \xleftarrow{\$} \mathsf{Setup.lpar}(1^{\mathfrak{K}})$*, for all tags* tag*, for all* $\ddot{\chi} \in \ddot{\mathscr{L}}$*, and for all valid witnesses* $(\ddot{w}_{\aleph}, \ddot{w}_{\exists})$ *of* $\ddot{\chi}$*:*

$$\Pr\left[\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \mathsf{lpar}); \langle \mathsf{ZK.Prove}((\ddot{w}_{\aleph}, \ddot{w}_{\exists})), \mathsf{ZK.Ver}\rangle_{\mathsf{crs}}(\mathsf{tag}, \ddot{\chi}) = 1\right] \geq 1 - \varepsilon;$$

- Soundness. *A polynomial-time adversary cannot make a verifier accept a proof for a word* $\ddot{\chi} \notin \ddot{\mathscr{L}}$*. Formally, the success probability of an ITM adversary* $\mathcal{A}$ *against soundness is defined by the experiment* Exp^{sound} *depicted in Figure 2.4.* ZK *is* sound *if this success probability is negligible for any polynomial-time ITM adversary* $\mathcal{A}$*;*

- (Unbounded) zero-knowledge. *A polynomial-time adversary cannot distinguish normal proofs made by* $\mathsf{ZK.Prove}$ *from simulated proofs made by* $\mathsf{ZK.Sim}$*. Formally, the advantage of an ITM adversary* $\mathcal{A}$ *against zero-knowledge is defined by the experiment* $\mathsf{Exp}^{zk\text{-}b}$ *depicted in Figure 2.4, where* $\mathsf{ZK.Sim}'(\mathsf{trap})$ *takes as common input a tag* tag *and a word* $\ddot{\chi}$ *and as private input a witness* $(\ddot{w}_{\aleph}, \ddot{w}_{\exists})$*, runs* $\mathsf{ZK.Sim}_2(\mathsf{trap})$ *with common input* tag *and* $\ddot{\chi}$ *if* $\mathscr{R}(\ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) = 1$ *and aborts otherwise.* ZK *is* zero-knowledge *if this advantage is negligible for any polynomial-time ITM adversary* $\mathcal{A}$*;*

- Extraction reference string indistinguishability. *A polynomial-time adversary cannot distinguish a normal CRS* crs *generated by* $\mathsf{ZK.Setup}$ *from one generated by* $\mathsf{ZK.Ext}_1$*.*

Formally, the advantage of an adversary $\mathcal{A}$ against reference string indistinguishability is defined by the experiment $\mathsf{Exp}^{\mathit{ext\text{-}crs\text{-}ind\text{-}b}}$ *depicted in Figure 2.4.* ZK *is* extraction-reference-string-indistinguishable *if this advantage is negligible for any polynomial-time adversary $\mathcal{A}$.*

- (Perfect) extractor indistinguishability. *The extractor* ZK.Ext *behaves exactly the same way as an honest verifier* ZK.Ver*, when the CRS is generated by* $\mathsf{ZK.Ext}_1$*. Formally,* ZK *is* extractor-indistinguishable *if, for any word $\ddot{\chi} \in \mathcal{X}$, for any* $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{ZK.Sim}_1(\mathsf{gpar})$*, for any (unbounded) adversary $\mathcal{A}$, the distribution of $\langle \mathcal{A}, \boxed{\mathsf{ZK.Ver}}_1 \rangle$ is identical to the distribution of $\langle \mathcal{A}, \boxed{\mathsf{ZK.Ext}_1(\mathsf{trap})}_1 \rangle$ when we restrict the output of* $\mathsf{ZK.Ext}_1$ *to the first element b of the ordered pair $(b, \ddot{w}_{\text{Ꙗ}})$;*

- (Partial) extractability. *When playing against the extractor* ZK.Ext*, a polynomial-time adversary cannot make the extractor accept, while making it unable to extract a valid partial witness $\ddot{w}_{\text{Ꙗ}}$. Formally, the success probability of an ITM adversary $\mathcal{A}$ against partial extractability is defined by the experiment* $\mathsf{Exp}^{\mathit{ext}}$ *depicted in Figure 2.4.* ZK *is* (partially)-extractable *if this success probability is negligible for any polynomial-time ITM adversary $\mathcal{A}$.*

We remark that soundness is implied by the extraction reference string indistinguishability, extractor indistinguishability, and partial extractability properties. Conversely, these three properties are implied by soundness, when $\ddot{w}_{\text{Ꙗ}}$ is not used, and in that case, we get the classical notion of zero-knowledge arguments (by using ZK.Setup and ZK.Ver as $\mathsf{ZK.Ext}_1$ and $\mathsf{ZK.Ext}_2$, respectively, and setting $\mathsf{trap} = \perp$ and $\ddot{w}_{\text{Ꙗ}} = \perp$ in their respective outputs). However, we keep soundness as a security requirement for simplicity.

When we talk about partial extractability, we sometimes mean partial extractability together with extractor indistinguishability and extraction reference string indistinguishability. For example, when we say that some property is implied by partial extractability, it means it is implied by partial extractability, extractor indistinguishability, and reference string indistinguishability.

When soundness holds statistically, ZK is also called a *(partially extractable) zero-knowledge proof*. When it does not hold statistically, we remark that we need to be very careful as the condition $\ddot{\chi} \in \ddot{\mathcal{X}} \setminus \ddot{\mathscr{L}}$ in the experiment $\mathsf{Exp}^{\mathsf{sound}}$ might not be testable in polynomial time. The same issue arises with partial extractability.

As labels for IND-CPA encryption schemes, tags tag are not really useful and are omitted if they are not used.

2.4.2.3 Extractable Honest-Verifier Zero-Knowledge Arguments

Definition 2.4.4. *An* extractable honest-verifier zero-knowledge argument *for a language $\mathscr{L} \subseteq \mathcal{X}$ is a tuple* $\mathsf{ZK} = (\mathsf{ZK.Setup}, \mathsf{ZK.Prove}, \mathsf{ZK.Ver}, \mathsf{ZK.Sim} = (\mathsf{ZK.Sim}_1, \mathsf{ZK.Sim}_2), \mathsf{ZK.Ext} = (\mathsf{ZK.Ext}_1, \mathsf{ZK.Ext}_2))$*, which satisfies the same properties as a zero-knowledge extractable argument, except that* $\mathsf{ZK.Sim}_2$ *is now just a polynomial-time algorithm and not an ITM, and the zero-knowledge property is replaced by the following weaker property:*

- Honest-verifier zero-knowledge. *A polynomial-time adversary cannot distinguish normal proofs made by* ZK.Prove *from simulated proofs made by* ZK.Sim*, with regards to an honest verifier. Formally, the advantage of an adversary $\mathcal{A}$ against honest-verifier*

Soundness

$\mathsf{Exp}^{\mathtt{sound}}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $((\mathsf{tag}, \ddot{\mathfrak{x}}), \mathsf{tr}, b) \xleftarrow{\$} \langle \mathcal{A}(\ddot{\mathsf{ltrap}}), \boxed{\mathsf{ZK.Ver}}_1 \rangle_{\mathsf{crs}}$
- **if** $b = 1$ and $\ddot{\mathfrak{x}} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}} \setminus \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ **then**
  - **return** 1
- **else**
  - **return** 0

Extraction Reference String Indistinguishability

$\mathsf{Exp}^{\mathtt{ext\text{-}crs\text{-}ind\text{-}0}}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$

$\mathsf{Exp}^{\text{ext-crs-ind-1}}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{ZK.Ext}_1(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$

Zero-Knowledge

$\mathsf{Exp}^{\mathtt{zk}-0}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\langle \boxed{\mathsf{ZK.Prove}}, \mathcal{A}(\ddot{\mathsf{ltrap}}) \rangle_{\mathsf{crs}}$

$\mathsf{Exp}^{\mathrm{zk}-1}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{ZK.Sim}_1(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\langle \boxed{\mathsf{ZK.Sim}'(\mathsf{trap})}, \mathcal{A}(\ddot{\mathsf{ltrap}}) \rangle_{\mathsf{crs}}$

Partial Extractability

$\mathsf{Exp}^{\mathrm{ext}}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $((\mathsf{tag}, \ddot{\mathfrak{x}}), \mathsf{tr}, (b, \ddot{w}_{\daleth})) \xleftarrow{\$} \langle \mathcal{A}(\ddot{\mathsf{ltrap}}), \boxed{\mathsf{ZK.Ext}(\mathsf{trap})}_1 \rangle_{\mathsf{crs}}$
- **if** $b = 1$ and $\forall \ddot{w}_{\exists}, \ddot{\mathscr{R}}(\ddot{\mathfrak{x}}, (\ddot{w}_{\daleth}, \ddot{w}_{\exists})) = 0$ **then**
  - **return** 1
- **else**
  - **return** 0

Figure 2.4: Experiments for Definition 2.4.3 (zero-knowledge argument)

Honest-Verifier Zero-Knowledge

$\mathsf{Exp}^{\mathtt{hvzk\text{-}0}}(\mathcal{A}, \mathfrak{K})$
  $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
  $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup.lpar}}(\mathsf{gpar})$
  $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
  $(\mathsf{st}, (\mathsf{tag}, \ddot{\chi}), (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$
  if $\mathscr{R}(\ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) = 1$ **then**
    $b \xleftarrow{\$} \langle \mathsf{ZK.Prove}((\ddot{w}_{\aleph}, \ddot{w}_{\exists})), \mathsf{ZK.Ver} \rangle_{\mathsf{crs}}(\mathsf{tag}, \ddot{\chi})$
    $\mathsf{tr} \leftarrow$ the previous transcript
    $r \leftarrow$ the random tape of $\mathsf{ZK.Ver}$
    return $\mathcal{A}(\mathsf{st}, \mathsf{tr}, r)$
  else
    return 0

$\mathsf{Exp}^{\mathtt{hvzk\text{-}1}}(\mathcal{A}, \mathfrak{K})$
  $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
  $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup.lpar}}(\mathsf{gpar})$
  $\mathsf{crs} \xleftarrow{\$} \mathsf{ZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
  $(\mathsf{st}, (\mathsf{tag}, \ddot{\chi}), (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$
  if $\mathscr{R}(\ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) = 1$ **then**
    $(\mathsf{tr}, r) \xleftarrow{\$} \mathsf{ZK.Sim}_2(\mathsf{crs}, \mathsf{trap}, (\mathsf{tag}, \ddot{\chi}))$
    return $\mathcal{A}(\mathsf{st}, \mathsf{tr}, r)$
  else
    return 0

Figure 2.5: Experiments for Definition 2.4.4 (honest-verifier zero-knowledge)

zero-knowledge is defined by the experiments $\mathsf{Exp}^{\mathtt{hvzk\text{-}b}}$ *depicted in Figure 2.5.* ZK *is* honest-verifier zero-knowledge *if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$.

2.4.2.4 Simulation-(Partially)-Extractable Zero-Knowledge Arguments

Definition 2.4.5. *A* simulation-(partially)-extractable zero-knowledge argument *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is a tuple* $\mathsf{ZK} = (\mathsf{ZK.Setup}, \mathsf{ZK.Prove}, \mathsf{ZK.Ver}, \mathsf{ZK.Sim} = (\mathsf{ZK.Sim}_1, \mathsf{ZK.Sim}_2), \mathsf{ZK.Ext} = (\mathsf{ZK.Ext}_1, \mathsf{ZK.Ext}_2))$*, which verifies the same properties as a zero-knowledge extractable argument with* $\mathsf{ZK.Sim}_1 = \mathsf{ZK.Ext}_1$ *and an additional property:*

- Simulation partial extractability. *When playing against the extractor* ZK.Ext*, a polynomial-time adversary cannot make the extractor accept, while making it unable to extract a valid partial witness* $\ddot{w}_{\aleph}$*, even if it has access to simulated proofs for any words* $\ddot{\chi}$ *of its choice. Formally, the success probability of two coordinated ITM adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *against simulation partial extractability is defined by the experiment* $\mathsf{Exp}^{\mathtt{sim\text{-}ext}}$ *depicted in Figure 2.6.* ZK *is* simulation-(partially)-extractable *if this success probability is negligible for any coordinated polynomial-time ITM adversaries* $\mathcal{A}_1$ *and* $\mathcal{A}_2$.

Simulation-Extractability

$\mathsf{Exp}^{\text{sim-ext}}(\mathcal{A}_1, \mathcal{A}_2, \mathfrak{K})$
  $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
  $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup.lpar}}(\mathsf{gpar})$
  $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{ZK.Sim}_1(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
  $(\mathsf{st}, (\mathsf{tag}, \ddot{\chi}), (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$
  $(\ddot{\chi}, \mathsf{tr}, (b, \ddot{w}_{\aleph})) \xleftarrow{\$} (\langle \boxed{\mathsf{ZK.Sim}_2(\mathsf{trap})}, \mathcal{A}_1(\ddot{\mathsf{ltrap}})\rangle, \langle \mathcal{A}_2(\ddot{\mathsf{ltrap}}), \boxed{\mathsf{ZK.Ext}}_1\rangle)_{\mathsf{crs}}$
  let T be the set of tags used by $\boxed{\mathsf{ZK.Sim}_2}$
  if $b = 1$ and $\forall \ddot{w}_{\exists}, \mathscr{R}(\ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists})) = 0$ and $\mathsf{tag} \notin T$ **then**
    return 1
  else
    return 0

Figure 2.6: Experiment for Definition 2.4.5 (simulation-extractability)

Simulation partial extractability implies extractability. Furthermore, when $\ddot{w}_{\aleph}$ is not used, simulation partial extractability (together with extraction reference string indistinguishability and extractor indistinguishability) corresponds to *simulation-soundness*.

We also consider *t-time simulation-extractability* and *t-time simulation-soundness* that are similar except that the simulator can only be called at most t times.

2.4.3 Non-Interactive Zero-Knowledge Arguments

Non-interactive zero-knowledge arguments are particular cases of zero-knowledge arguments, except that ZK.Prove and $\mathsf{ZK.Sim}_2$ just output the proof π for the word $\ddot{\chi}$ on their communication output tape and do not use their communication input tapes, while ZK.Ver and $\mathsf{ZK.Ext}_2$ just read this proof π from their communication input tapes and do not use their communication output tapes. Security properties are defined exactly the same way.

As dealing with ITM is more cumbersome than with classical algorithms, we re-define NIZK as a tuple of probabilistic polynomial-time algorithms $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver}, \mathsf{NIZK.Sim} = (\mathsf{NIZK.Sim}_1, \mathsf{NIZK.Sim}_2), \mathsf{NIZK.Ext} = (\mathsf{NIZK.Ext}_1, \mathsf{NIZK.Ext}_2))$, where:

- $\mathsf{NIZK.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ outputs a CRS crs;
- $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, \ddot{\chi}, (\ddot{w}_{\aleph}, \ddot{w}_{\exists}))$ outputs a proof π with tag tag for the word $\ddot{\chi}$ using the witness $(\ddot{w}_{\aleph}, \ddot{w}_{\exists})$;
- $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, \ddot{\chi}, \pi)$ outputs 1 if π is a valid proof with tag tag for $\ddot{\chi}$, and outputs 0 otherwise;
- $\mathsf{NIZK.Sim}_1(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ outputs a CRS crs and a trapdoor trap;
- $\mathsf{NIZK.Sim}_2(\mathsf{crs}, \mathsf{trap}, \mathsf{tag}, \ddot{\chi})$ outputs a simulated proof π with tag tag for the word $\ddot{\chi}$;
- $\mathsf{NIZK.Ext}_1(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ outputs a CRS crs and a trapdoor trap;
- $\mathsf{NIZK.Ext}_2(\mathsf{crs}, \mathsf{trap}, \mathsf{tag}, \ddot{\chi}, \pi)$ outputs a pair $(b, \ddot{w}_{\aleph})$ where $b = 1$ if the proof is valid and 0 otherwise, and $\ddot{w}_{\aleph}$ is a partial witness (if $b = 1$);

2.5 Projective Hash Functions

In this section, we define the central cryptographic primitive of this thesis: *projective hash functions* (PHFs). As already explained in the introduction, a PHF is defined over an NP language. We therefore first define the kind of languages we are considering before defining PHF and some of its associated security notions: smoothness and universality.

PHFs have been introduced by Cramer and Shoup in their seminal paper [CS02]. We follow Abdalla, Chevalier, and Pointcheval in [ACP09] for the formalization of PHFs and their associated languages. In particular, we use long algorithm names for each function or algorithm in a PHF, instead of short one letter names as in [CS02]. We also consider two variants of smoothness which slightly differ from the original smoothness definition. These differences are explained in more details in this section.

2.5.1 Projective Hash Functions (PHFs)

Definition 2.5.1. *A projective hash function (PHF) for a language* $(\mathscr{L}_{\mathsf{lpar}})$ *is defined by a tuple of four polynomial-time algorithms* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*, where:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *generates a hashing key* hk *for the language parameters* lpar*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *deterministically derives a projection key* hp *from the hashing key* hk*, the language parameters* lpar*, and possibly the word* $\chi \in \mathscr{X}_{\mathsf{lpar}}$*;*
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *deterministically outputs a hash value* H *from the hashing key* hk*, for the word* $\chi \in \mathscr{X}_{\mathsf{lpar}}$ *and the language parameters* lpar*;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *deterministically outputs a projected hash value* pH *from the projection key* hp*, and the witness* w*, for the word* $\chi \in \mathscr{L}_{\mathsf{lpar}}$ *(i.e.,* $\mathscr{R}_{\mathsf{lpar}}(\chi, w) = 1$*) and the language parameters* lpar*;*

The set of hash values is called the range *and is denoted by* Π*. It is often a cyclic group. We suppose that it is possible to pick a uniform element of* Π *in polynomial time. We always suppose that its size is* exponential *in the security parameter* $\mathfrak{K}$ *so that* the probability to guess correctly a uniform hash value is negligible.

A PHF has to satisfy the following property:

- Perfect correctness. *For any* lpar *and any word* $\chi \in \mathscr{L}_{\mathsf{lpar}}$ *with witness* w *(i.e., such that* $\mathscr{R}_{\mathsf{lpar}}(\chi, w) = 1$*), for any* $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$ *and for* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$*,*

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w) \ .$$

Dependence of hp on χ. Originally, in [CS02], the projection key hp was supposed to be computed independently of χ. That is, ProjKG did not take χ as an input. In some applications, the independence of hp and χ is required. But, in other applications, such as PAKE, as remarked by Gennaro and Lindell [KOY09; GL06], this is not required and allowing hp to depend on χ enables to get more efficient PHF for more languages. In this manuscript, we consider both cases, and indicate in which case we are, whenever necessary.

Figure summary. Figure 2.7 summarizes the definition of PHF. The parameter lpar is implicit and the dash arrow from χ to ProjKG indicates that hp may or may not depend on χ.

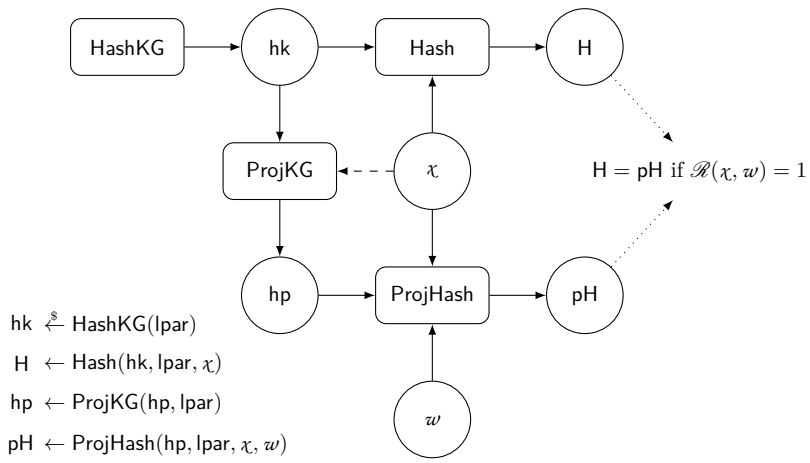

Figure 2.7: Summary of the definition of PHF

2.5.2 Smooth Projective Hash Functions (SPHFs)

To satisfy the correctness property of a PHF, we could just set $\mathsf{hk} = \mathsf{hp}$ and not use any witness at all. In addition to correctness, many security notions exist for PHFs. We start by introducing one of the simplest ones: smoothness. Later in this manuscript, we will introduce other security notions for PHF, both stronger (to construct more complex applications) and weaker (to handle more languages). A smooth PHF is called a *smooth projective hash function* (SPHF).

We consider two variants of smoothness. Both these variants are different from the original smoothness definition of Cramer and Shoup, which supposed that words were randomly chosen and so which cannot easily handle words generated by malicious adversaries. It should however be remarked that all the constructions of SPHF in [CS02] satisfy both our smoothness definitions.

Essentially, smoothness says that knowing only hp (but not hk), the hash value H of a word $\chi \notin \mathscr{L}_{\mathsf{lpar}}$ looks uniformly random in the set of possible hash values Π.

2.5.2.1 Smoothness for GL-SPHF and CS-SPHF

Definition 2.5.2. *A PHF* (HashKG, ProjKG, Hash, ProjHash) *for a language* $\mathscr{L} \subseteq \mathscr{X}$ *is* ε-GL/CS-smooth *if for any* lpar *and any word* $\chi \in \mathscr{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$*, the following distributions are* ε*-close:*

$$\left\{(\mathsf{hp}, \mathsf{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi); \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)\right\}$$
$$\left\{(\mathsf{hp}, \mathsf{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi); \mathsf{H} \xleftarrow{\$} \Pi\right\} \quad .$$

A PHF is GL/CS-smooth *if it is* ε*-GL/CS-smooth with* ε *negligible in* $\mathfrak{K}$*.*

An SPHF satisfying this smoothness definition is called

- a *CS-smooth projective hash function* (CS-SPHF) if hp does not depend on χ (i.e., ProjKG does not use its input χ); the name CS-SPHF comes from the fact this kind of

SPHF is the closest one to the original notion of strong smoothness[10] of Cramer and Shoup, the only difference being that our smoothness holds for any word $\chi \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathcal{L}_{\mathsf{lpar}}$, while the original definition only holds on average on such words;

- a *GL-smooth projective hash function* (GL-SPHF) if hp may depend on χ; the name GL-SPHF comes from the fact that this variant has been introduced by Gennaro and Lindell in [GL06].

2.5.2.2 Smoothness for KV-SPHF

In [KV11], Katz and Vaikuntanathan remarked that in their one-round PAKE protocol, χ can be maliciously generated by the adversary after seeing hp. So the classical smoothness notion is not sufficient here, since it only ensures that H looks uniformly random when hk and hp are chosen independently of χ. We therefore consider a stronger smoothness definition. SPHFs satisfying this stronger security notion are called *KV-smooth projective hash function* (KV-SPHF). Obviously the projection key hp of a KV-SPHF cannot depend on the word χ.

Definition 2.5.3. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathcal{L} \subseteq \mathcal{X}$ *is* ε-KV-smooth *if for any* lpar *and any function*[11] f *from the set of possible projection keys* hp *to* $\mathcal{X}_{\mathsf{lpar}} \setminus \mathcal{L}_{\mathsf{lpar}}$*, the following distributions are* ε*-close:*

$$\left\{(\mathsf{hp}, \mathsf{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f(\mathsf{hp}))\right\}$$
$$\left\{(\mathsf{hp}, \mathsf{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \mathsf{H} \xleftarrow{\$} \Pi\right\} \quad .$$

A PHF is KV-smooth *if it is* ε*-KV-smooth with* ε *negligible in* $\mathfrak{K}$*.*

Remark 2.5.4. *In general, an* ε*-smooth CS-SPHF is not necessarily an* ε*-smooth KV-SPHF. For example, the projection key* hp *of a CS-SPHF might contain a random word in* $\mathcal{X} \setminus \mathcal{L}$ *and its corresponding hash value, while the projection key of a KV-SPHF cannot. However, a* 0*-smooth (or perfectly smooth) CS-SPHF is also a* 0*-smooth (or perfectly smooth) KV-SPHF, as in that case both distributions in the CS smoothness definition are actually equal and so stay equal when conditioned on the value of* hp*.*

2.5.3 Simple Applications of SPHFs

We now show some simple applications of SPHFs, with a dual goal: to help to better understand what SPHFs and in the same time, and to give some intuition of their restrictions (in particular regarding zero-knowledge) and how these restrictions have been lifted in our works. More precisely, the main message we want to convey is the following:

> An SPHF does not provide any security when the projection key is maliciously generated. That often means that naive use of SPHF does not provide any kind of zero-knowledge property but generally only an honest-verifier zero-knowledge property.

We keep the description of the applications in this section slightly informal for readability. Formal applications of the tools introduced in this thesis are later provided in particular in Chapter 6.

[10] In [CS02], smoothness does not require ε to be negligible in $\mathfrak{K}$, but only strong smoothness requires that.
[11] Not necessarily computable in polynomial time: everything we do here is statistical.

Prover		Verifier
Input: (χ, w)	CRS: lpar	Input: χ
		$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$
	$\xleftarrow{\quad \mathsf{hp} \quad}$	$\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$
$\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$	$\xrightarrow{\quad \mathsf{pH} \quad}$	
		$\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$
		if $\mathsf{pH} = \mathsf{H}$ **then accept**
		else reject

Figure 2.8: Honest-verifier zero-knowledge proof from an SPHF

2.5.3.1 Honest-Verifier Zero-Knowledge Proofs

The simplest application of SPHF for a language $\mathscr{L} \subseteq \mathcal{X}$ is certainly the construction of an honest-verifier zero-knowledge proof for the same language $\ddot{\mathscr{L}} := \mathscr{L} \subseteq \mathcal{X} =: \ddot{\mathcal{X}}$. The construction is depicted in Figure 2.8: the verifier generates a hashing key hk and an associated projection key hp, sends the latter, and asks the prover to give him the hash value of the word χ the prover wants to prove to be in $\mathscr{L}$. The verifier accepts if and only if the received hash value is equal to the one he can compute using the hashing key hk.

On the one hand, if the prover knows a witness w for χ, he can just compute this hash value using the projection key hp and this witness w, and the verifier will accept the proof. Thus, we get *correctness*.

On the other hand, if the word χ is not in the language, the smoothness property ensures that the hash value expected by the adversary is statistically indistinguishable from a random value. As the size of the set Π is exponential in the security parameter $\mathfrak{K}$, the probability that the prover can guess the hash value is negligible in $\mathfrak{K}$. Thus we get *statistical soundness*.

It remains to prove *perfectly honest-verifier zero-knowledge*. This is straightforward, as when the verifier is honest, he already knows what the prover will send to him! So he definitely can learn nothing and simulation is straightforward

We remark that the construction works for any variant of SPHF: GL-SPHF, CS-SPHF, and KV-SPHF. When a KV-SPHF is used, the first flow can be sent before the word χ is known, and the protocol remains secure. This would not necessarily be the case with a GL-SPHF (actually, the modified protocol cannot even be constructed if hp needs to depend on the word χ) or a CS-SPHF.

We might wonder if this construction is zero-knowledge and not only honest-verifier zero-knowledge. Unfortunately, this is not the case. We can see two intuitive reasons for that. First, given a projection key hp, an SPHF does not provide anyway to compute a corresponding hash value H for a word $\chi \in \mathscr{L}$ if we do not know the witness. Actually, this is even worse than that: if the language is hard-subset-membership, then it is computationally hard to tell if $\chi \in \mathscr{L}$ or not, and if it is not, there is statistically no way to compute H. Second, even if we manage to solve this issue, it is still possible that the verifier can construct the projection key hp in some way it reveals information about the witness w used by the prover.

An important question in this thesis is to find ways to add zero-knowledge to SPHFs. The main tool for that is the classical "or" trick.

2.5.3.2 Witness Encryption and Limitations of SPHF

In [GGSW13], Garg et al. introduced a new primitive called *witness encryption*, which can be seen as a computational version of GL-SPHF. Intuitively, a witness encryption scheme enables a user to encrypt a message m to any user who knows a witness for some word χ. It can be seen as a generalization of a classical encryption scheme which enables a user to encrypt a message m to any user knowing the decryption key dk associated to some encryption key $\chi = \mathsf{ek}$.

Concretely, it is possible to construct witness encryption for any language handled by a GL-SPHF. To encrypt some message m to some word χ, we just pick a random hashing key hk, derive a projection key hp from hk (and possibly χ), and output the ciphertext $\mathsf{c} = (\mathsf{hp}, \mathsf{c}')$ with $\mathsf{c}' := \mathsf{H} \,\mathsf{xor}\, \mathsf{m}$ and H is the hash value of the word χ.[12] Anyone knowing a witness w for the word χ, can compute H as pH using the projection key hp and this witness w, and then recover the message m by computing $\mathsf{c}' \,\mathsf{xor}\, \mathsf{pH}$.

In [GGSW13], Garg et al. showed that statistical witness encryption (and therefore SPHF) cannot exist for any NP language[13] unless the polynomial hierarchy collapses. This comes from the fact that, as shown in Section 2.5.3.1, an SPHF for some language $\mathscr{L}$ directly yields a statistical honest-verifier zero-knowledge proof for the same language, which implies that $\mathscr{L}$ is in the complexity class SZK [GSV98]. But we know that SZK $\subseteq$ AM $\cap$ co-AM [For87; AH91], and therefore the existence of an SPHF for any NP language implies that NP $\subseteq$ AM $\cap$ co-AM, which in turn implies that the polynomial hierarchy collapses [BHZ87].

2.5.3.3 Sending a Message to a Secret Agent

A nice application of SPHF (and actually just of witness encryption too) is the following. Let us suppose that Alice is a secret agent and she wants to receive some information (a message m) from some informer, Bob. Alice has a certificate signed by the CIA that she is a secret agent but does not want to reveal it to any informer, for obvious reasons. On the other hand, the informer does not want to give m to a bad guy and really wants to be sure that only an official secret agent will be able to read the message.

This seems completely infeasible, but SPHF enables to do it. Actually, it is not just theoretical but efficient and practical, at least for the purely cryptographic part.

Suppose that there is some CRS containing an encryption key $\mathsf{lpar} = \mathsf{ek}$ for an IND-CPA encryption scheme. Also suppose that the secret agent has some certificate σ (under some public key pk)[14] and that the language $\mathscr{L}$ of valid encryption of valid certificates has an associated SPHF:

$$\mathscr{L} := \{\chi \mid \exists r,\, \exists \sigma,\, \chi = \mathsf{Enc}(\mathsf{ek}, \sigma; r) \text{ and } \sigma \text{ is a valid certificate for } \mathsf{pk}\}$$

If the certificate is a Waters signature[15] [Wat05] and the encryption scheme is ElGamal over the same cyclic group, this is efficiently implementable, as checking a Waters signature can

[12] We assume that Π is just the set $\{0,1\}^k$ for some large enough k and that the message m is in the same set. We can use a randomness extractor to make Π of this form. Furthermore, if Π is a group, we can also encode the message in Π and use the group operation on Π instead of xor.

[13] In our formalization of languages, we can see an NP language as a family of languages, with $\mathsf{gpar} = \mathsf{lpar} = \mathfrak{K}$, and $\mathcal{X}_{\mathfrak{K}} = \{0,1\}^{\mathfrak{K}}$.

[14] A certificate needs to be associated to some public key.

[15] of some fixed message or even of some message satisfying some simple properties.

Informer		Secret agent
Input: message m	CRS: $\mathsf{lpar} = (\mathsf{ek}, \mathsf{pk})$	Input: certificate σ
		$\mathsf{c} \xleftarrow{\$} \mathsf{Enc}(\mathsf{ek}, \sigma; r)$ with fresh random coins r
	$\xleftarrow{\quad \mathsf{c} \quad}$	
$\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$ $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$ $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathsf{c})$ $\mathsf{c}' \leftarrow \mathsf{H} \,\mathsf{xor}\, \mathsf{m}$		
	$\xrightarrow{\quad \mathsf{hp},\ \mathsf{c}' \quad}$	
		$\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \mathsf{c}, (r, \sigma))$ $\mathsf{m} \leftarrow \mathsf{c}' \,\mathsf{xor}\, \mathsf{pH}$ Output the message m

Figure 2.9: Protocol enabling an informer to send a message m to a secret agent

be done using a simple pairing equation and SPHF can handle verification of pairing equation of encrypted group elements (see Section 3.4.3).

The protocol is described in Figure 2.9. The secret agent encrypts its certificate under the encryption key ek in the CRS and sents the resulting ciphertext c to the informer. The informer generates a hashing key hk, derives a projection key hp, and uses the hash value H of c under hk as a one-time pad (or mask) for the message m: $\mathsf{c}' := \mathsf{H} \,\mathsf{xor}\, \mathsf{m}$.[16] He then sends the projection key hp together with c'. If the secret agent encrypted a valid certificate σ, using the random coins r used by the encryption, the certificate, and hp, he can compute the projected hash value pH (which is equal to H by correctness of the SPHF) and recover the message from c', as $\mathsf{m} = \mathsf{c}' \,\mathsf{xor}\, \mathsf{pH}$.

If the secret agent is not a real secret agent, he does not know any certificate σ and therefore has no way to produce a ciphertext c of a valid certificate. In other words, the ciphertext c sent by a fake secret agent will not be in $\mathscr{L}$. Thus, the smoothness of the SPHF ensures that from the secret agent point of view (who only sees hp and c' but not hk), H is uniformly random and completely masks the message m.

We remark that this protocol actually ensures a very strong property of forward secrecy: if the fake secret agent later becomes a real secret agent, it is already too late. As $\mathsf{c} \notin \mathscr{L}_{\mathsf{lpar}}$, there is no way for him to recover the message m sent to him while he was not a secret agent yet. Furthermore this property holds statistically. Concretely, this means that the message m remains completely hidden, even if later the encryption scheme used is broken, e.g., by quantum computers.

When the informer is honest, he clearly learns nothing, as he knows exactly what a real secret agent will get at the end. However, similarly to the honest-verifier zero-knowledge protocol in Section 2.5.3.1, this protocol might not be secure in case of a malicious informer (which corresponds to the verifier in Section 2.5.3.1): the informer might generate an invalid projection key hp such that depending on the certificate of the secret agent, the secret agent will read two different messages and later act differently. If everything ends when the secret agent receives the message and nothing happens later, this does not matter; but this is a rather restrictive model. In general, a stronger notion is required.

[16] See Footnote 12 on page 52 for the use of xor.

2.5.3.4 IND-CPA Encryption Scheme

A CS-SPHF for any hard-subset-membership language $\mathscr{L}$ can be used to construct an IND-CPA encryption scheme in a straightforward manner. The construction is as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ generates the global parameters $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$ and the language parameters $\mathsf{lpar} \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar}')$ for the language $\mathscr{L}$ and outputs the (encryption) global parameters $\mathsf{gpar}' := \mathsf{lpar}$;
- $\mathsf{KeyGen}(\mathsf{gpar}')$ generates a random hashing key $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$, derives the projection key $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$, and outputs $(\mathsf{ek}, \mathsf{dk}) := (\mathsf{hp}, \mathsf{hk})$;
- $\mathsf{Enc}(\mathsf{ek}, \mathsf{m})$ generates a random pair $(\chi, w) \xleftarrow{\$} \mathscr{R}_{\mathsf{lpar}}$ and outputs the ciphertext $\mathsf{c} := (\chi, \mathsf{c}')$ with $\mathsf{c}' := \mathsf{pH} \,\mathsf{xor}\, \mathsf{m}$ where $\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$;[17]
- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ outputs $\mathsf{c}' \,\mathsf{xor}\, \mathsf{H}$, where $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$.

Correctness follows from the correctness of the SPHF.

To prove the IND-CPA security, note that the simulator can also generate the part c' of the ciphertext as $\mathsf{c}' := \mathsf{H} \,\mathsf{xor}\, \mathsf{m}$ where $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$. Since, in that case it does not need to use w, this is indistinguishable to the case where it generates χ as $\chi \xleftarrow{\$} \mathscr{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$, thanks to hard subset membership. Finally, smoothness ensures that in this setting, H is uniformly random in Π and therefore completely masks the message c'.

[17] See Footnote 12 on page 52 for the use of xor.

Chapter 3

Diverse Vector Spaces

As a warm-up for *diverse modules* (DMs) which constitute the core of this thesis and as an important particular case, we introduce *diverse vector spaces* (DVSs). Intuitively, a DVS is a way to describe languages which can be seen as subspaces of some vector space over some finite field. Cramer and Shoup already showed in their seminal paper [CS02] that such languages automatically admit SPHFs. This class of languages might seem very restrictive, but actually encompasses many languages from Diffie-Hellman-like languages to tuples of ElGamal-like ciphertexts satisfying a system of multi-exponentiation equations or even quadratic bilinear pairing equations (when over a bilinear group). Furthermore, as far as we know, all the existing constructions of SPHFs over cyclic groups follow from this construction (or are minor variants thereof).

In this chapter, we first define DVSs and give an overview of the expressive power of DVSs on some examples. We then show that DVSs can be combined together: given two DVSs, we design a new DVS corresponding to their conjunction and disjunction (under some simple conditions in some cases). Next, we show how to use disjunctions of DVSs to construct constant-size non-interactive zero-knowledge proofs. We conclude with some more advanced examples of DVSs.

Contents

3.1 First Examples, Definition, and Link with SPHFs **56**
3.1.1 Step-by-Step Overview 56
3.1.2 Definition 62
3.2 Conjunctions and Disjunctions **68**
3.2.1 Conjunctions 68
3.2.2 Disjunctions 69
3.3 Application to Non-Interactive Zero-Knowledge Arguments **76**
3.3.1 Overview 76
3.3.2 Construction 78
3.3.3 Completeness and Security 79
3.4 More Examples **80**
3.4.1 Matrix Decisional Diffie-Hellman Assumptions (MDDH) 80
3.4.2 Cramer-Shoup Encryption 82
3.4.3 Encryption of Plaintexts Satisfying a System of Quadratic Equations . 83

3.1 First Examples, Definition, and Link with SPHFs

3.1.1 Step-by-Step Overview

In this overview, we introduce the various properties and elements of DVSs step by step. We start with a very simple DVS for a very simple language: the DDH language, which can directly be seen as a subspace of some vector space. We then extend DVS to more complex languages which are not directly subspaces of some vector space, but which can be transformed into; or languages for which the subspace might depend on the word χ of the language or even on some random coins to allow efficient batching.

3.1.1.1 The DDH Language as a Subspace of a Vector Space

Let $(p, \mathbb{G}, g)$ be a cyclic group and let us look back at the DDH language $\mathscr{L}$ defined in Example 2.3.1:

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (g^r,\, h^r)\} \quad \subsetneq \quad \mathbb{G}^2 = \mathcal{X} \ .$$

If the group law of $\mathbb{G}$ is denoted additively, $(\mathbb{G}, \boldsymbol{+})$, instead of multiplicatively, $(\mathbb{G}, \cdot)$, and if we use the symbol $\bullet$ to denote exponentiation by a scalar, we could have written this language as:

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (r \bullet g,\, r \bullet h) \ .$$

This notation makes us think about matrix multiplication, and we would like to write:

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = r \bullet (g, h)\} \ .$$

Looking at $(\mathbb{G}, \cdot)$ or $(\mathbb{G}, \boldsymbol{+})$ as $(\mathbb{Z}_p, +)$, we could see $\mathscr{L}$ as the subspace of the vector space $\mathcal{X} = \mathbb{G}^2$ generated by the row vector (g, h). If we transpose everything to use more classical notation, we get:

$$\mathscr{L} = \left\{ \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{G}^2 \,\middle|\, \exists r \in \mathbb{Z}_p,\ \begin{pmatrix} u \\ v \end{pmatrix} = r \bullet \begin{pmatrix} g \\ h \end{pmatrix} \right\} \ ,$$

which can also be rewritten as:

$$\mathscr{L} = \mathrm{Span}\left(\begin{pmatrix} g \\ h \end{pmatrix} \right) \ .$$

3.1.1.2 Constructing an SPHF for the DDH Language

Looking back at the definition of an SPHF, from a high level point of view, we can see a (perfectly smooth) SPHF as a random function $\alpha : \mathcal{X} \to \Pi$ defined by some random hashing key hk, for which it is possible to derive a projection key hp which only defines this function α on $\mathscr{L}$. In our case, as the language is a subspace, a (uniformly) random linear function works, since the values of such a function on a subspace only defines this function on the subspace and not outside the subspace.

Concretely, this function α can just be a random function from $\mathcal{X}^*$, the dual of $\mathcal{X}$, i.e., the set of linear functions from $\mathcal{X}$ to $\mathbb{G}$. Such a function α can be defined by row vector $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times 2}$, as follows: $\alpha(\boldsymbol{\theta}) = \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$, for any vector $\boldsymbol{\theta} \in \mathbb{G}^2$. The hash value of a word χ is then

$$H \coloneqq \alpha(\chi) \coloneqq \boldsymbol{\alpha}^\intercal \bullet \chi = u^{\alpha_1} \cdot v^{\alpha_2} \ ,$$

when $\chi = \begin{pmatrix} u \\ v \end{pmatrix}$ is seen as a column vector in $\mathbb{G}^2$. Then, the projection key hp and the projected hash value pH for c with witness $r = \boldsymbol{w}$ are:

$$\mathsf{hp} := \gamma := \alpha(\begin{pmatrix} g \\ h \end{pmatrix}) = \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} g \\ h \end{pmatrix} = g^{\alpha_1} \cdot h^{\alpha_2} \in \mathbb{G} \qquad \mathsf{pH} := \gamma \bullet r = \gamma^r \in \mathbb{G} \ .$$

Correctness follows from the fact that if $(u, v) = (g^r, h^r)$:

$$\mathsf{H} = \boldsymbol{\alpha}^\intercal \bullet \chi = \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} g \\ h \end{pmatrix} \bullet r = \boldsymbol{\gamma}^\intercal \bullet r = \mathsf{pH} \ .$$

Smoothness directly comes from the previous discussion: even knowing the function α on every word in $\mathscr{L}$ does not give any information about its value outside $\mathscr{L}$, as $\mathscr{L}$ is a vector space. A formal proof is provided later in Section 3.1.2.3.

Historical note 3.1.1. *This SPHF for the DDH language was explicitly proposed by Cramer and Shoup in [CS02], but the ideas were already implicit in [CS98].*

3.1.1.3 ElGamal: Introducing θ

Now that we have an SPHF for the DDH language, we can design an SPHF for the language of pairs $\chi = (\mathsf{m}, \mathsf{c})$, where c is an ElGamal ciphertext of a message m in some message set $\mathscr{M}$, under the public key $\mathsf{lpar} = \mathsf{ek} = (g, h) \in \mathbb{G}^2$ (see Section 2.2.2.2 for the definition of the ElGamal encryption scheme):

$$\mathscr{L} = \{(\mathsf{m},\, \mathsf{c} = (u, v)) \in \mathscr{M} \times \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (g^r,\, h^r \cdot \mathcal{G}(\mathsf{m}))\} \,,$$

with $\mathcal{G}$ a reversible map from the message set $\mathscr{M}$ to $\mathbb{G}$. We write $\mathsf{M} = \mathcal{G}(\mathsf{m})$. We just need to use the SPHF for the DDH language on the word $(u, v/\mathsf{M})$:

$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times 2} \qquad \mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \cdot \begin{pmatrix} g \\ h \end{pmatrix} \in \mathbb{G}$$

$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} u \\ v/\mathsf{M} \end{pmatrix} = u^{\alpha_1} \cdot (v/\mathsf{M})^{\alpha_2} \in \mathbb{G} \qquad \mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet r \in \mathbb{G} \ .$$

We remark that, when the map $\mathcal{G}$ is not linear, the language $\mathscr{L}$ cannot be seen directly as a subspace of some vector space over $\mathbb{Z}_p$ or $\mathbb{G}$. But still, we can construct an SPHF, because we can map words $\chi = (\mathsf{m}, \mathsf{c} = (u, v)) \in \mathcal{X}$ to vectors $(u, v/\mathsf{M})^\intercal \in \mathbb{G}^2$, which live in the subspace of $\mathbb{G}^2$ generated by $(g, h)^\intercal$ if and only if c is an ElGamal ciphertext of m.

That is why, in the definition of a DVS, we do not require that the language is really a subspace, but only that there exists an (efficiently computable) function $\boldsymbol{\theta}$ from $\mathcal{X}$ to some vector space $\hat{\mathcal{X}}$ such that: $\boldsymbol{\theta}(\chi)$ is in some subspace $\hat{\mathscr{L}}$ of $\hat{\mathcal{X}}$ if and only if $\chi \in \mathscr{L}$. In the previous example:

$$\boldsymbol{\theta}(\chi) := (u,\, v/\mathsf{M})^\intercal \in \mathbb{G}^2 =: \hat{\mathcal{X}} \qquad \text{with } \chi = (\mathsf{m},\, \mathsf{c} = (u, v)) \in \mathcal{X} \,,$$

and

$$\hat{\mathscr{L}} := \mathrm{Span}\left(\begin{pmatrix} g \\ h \end{pmatrix}\right) \quad \subsetneq \quad \mathbb{G}^2 =: \hat{\mathcal{X}} \ .$$

We should point out that there is no restriction at all on $\boldsymbol{\theta}$, except that it should be computable in polynomial time.

The DDH language can also be represented this way by setting $\hat{\mathcal{X}} := \mathcal{X}$, $\hat{\mathscr{L}} := \mathscr{L}$, and $\boldsymbol{\theta}$ the identity function.

3.1.1.4 Introducing more Dimensions

In the previous examples, the subspace $\hat{\mathscr{L}}$ is of dimension 1 in a vector space $\mathcal{X}$ of dimension 2. But there are no reasons to restrict ourselves to such small dimensions. Let us now show an example of a language of higher dimension.

Example 3.1.2 (encryption of a DH tuple)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}, g)$*, with* $\mathbb{G}$ *being a cyclic group of prime order* p *generated by* g*, while the language parameters are* $\mathsf{lpar} := (g, h, g', h') \in \mathbb{G}^4$ *where* $\mathsf{ek} := (g, h)$ *is a random ElGamal encryption key, and where* g' *and* h' *are two random generators of* $\mathbb{G}$*. We consider the language of ElGamal encryption of DH tuples as follows:*

$$\mathscr{L} = \{(u_1, v_1, u_2, v_2)^\intercal \in \mathbb{G}^4 \mid \exists r_1, r_2, r' \in \mathbb{Z}_p, \\ (u_1, v_1, u_2, v_2) = (g^{r_1},\, h^{r_1} \cdot g'^{r'},\, g^{r_2},\, h^{r_2} \cdot h'^{r'})\} \ .$$

In other words, a word (u_1, v_1, u_2, v_2) *is in the language if and only if the ciphertexts* (u_1, v_1) *and* (u_2, v_2) *encrypt two elements* $u' \in \mathbb{G}$ *and* $v' \in \mathbb{G}$ *under* ek*, such that* (g', h', u', v') *is a DH tuple.*

This language can directly be seen as a DVS with $\boldsymbol{\theta}$ *the identity function:*

$$\hat{\mathscr{L}} := \mathscr{L} = \mathrm{Span}\left(\begin{pmatrix} g \\ h \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \end{pmatrix}, \begin{pmatrix} 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ g \\ h \end{pmatrix}, \begin{pmatrix} 1_{\mathbb{G}} \\ g' \\ 1_{\mathbb{G}} \\ h' \end{pmatrix}\right) = \mathrm{ColSpan}\,(\Gamma) \quad \subsetneq \quad \mathbb{G}^4 = \mathcal{X} =: \hat{\mathcal{X}}\,,$$

where Γ *is the matrix:*

$$\Gamma := \begin{pmatrix} g & 1_{\mathbb{G}} & 1_{\mathbb{G}} \\ h & 1_{\mathbb{G}} & g' \\ 1_{\mathbb{G}} & g & 1_{\mathbb{G}} \\ 1_{\mathbb{G}} & h & h' \end{pmatrix} \in \mathbb{G}^{4\times 3} \ .$$

We can now define an SPHF as follows:

$$\begin{aligned}
\mathsf{hk} &:= \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times 4} \\
\mathsf{hp} &:= \boldsymbol{\gamma}^\intercal := \left(\alpha\left(\begin{pmatrix} g \\ h \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \end{pmatrix}\right), \alpha\left(\begin{pmatrix} 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ g \\ h \end{pmatrix}\right), \alpha\left(\begin{pmatrix} 1_{\mathbb{G}} \\ g' \\ 1_{\mathbb{G}} \\ h' \end{pmatrix}\right)\right) = \boldsymbol{\alpha}^\intercal \cdot \Gamma \in \mathbb{G}^{1\times 3} \\
\mathsf{H} &:= \boldsymbol{\alpha}^\intercal \bullet \chi = u_1^{\alpha_1} \cdot v_1^{\alpha_2} \cdot u_2^{\alpha_3} \cdot v_2^{\alpha_4} \in \mathbb{G} \\
\mathsf{pH} &:= \boldsymbol{\gamma}^\intercal \bullet \begin{pmatrix} r_1 \\ r_2 \\ r' \end{pmatrix} \in \mathbb{G} \ .
\end{aligned}$$

The projection key hp *is now a row vector of 3 elements, as it has to define the linear map* α *over a subspace of dimension* 3*, namely* $\hat{\mathscr{L}}$*.*

3.1.1.5 Introducing Dependence on Word (GL-SPHF)

In all the previous examples, the subspace $\hat{\mathscr{L}}$ of $\hat{\mathscr{X}}$ is independent of the word χ. We can also consider languages where $\hat{\mathscr{L}}$ depends on χ, which we sometimes write $\hat{\mathscr{L}}_\chi$ instead of $\hat{\mathscr{L}}$. In this case however, we only get a GL-SPHF (compared to KV-SPHF in all the previous examples), as the projection key hp depends on $\hat{\mathscr{L}}$, or more precisely on the chosen basis of $\hat{\mathscr{L}}$.

Example 3.1.3 (encryption of a bit)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}, g)$*, with* $\mathbb{G}$ *being a cyclic group of prime order* p *generated by* g*, while the language parameters are* $\mathsf{lpar} := (g, h) \in \mathbb{G}^2$ *where* $\mathsf{ek} := (g, h)$ *is a random ElGamal encryption key. We consider the language of ElGamal encryption of a bit:*

$$\mathscr{L} = \{(u, v)^\intercal \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ \exists b \in \{0, 1\},\ (u, v) = (g^r,\ h^r \cdot g^b)\}\ .$$

This language cannot be directly be seen as a DVS, as b *is a bit and not a scalar in* $\mathbb{Z}_p$*. However, we can transform* $\mathscr{L}$ *into a subspace of the vector space* $\hat{\mathscr{X}} := \mathbb{G}^4$ *using the following function* $\boldsymbol{\theta}$ *and the following space* $\hat{\mathscr{X}}$ *and subspace* $\hat{\mathscr{L}}_\chi$*:*

$$\boldsymbol{\theta}(\chi) := (u,\ v,\ 1_\mathbb{G},\ 1_\mathbb{G})^\intercal \in \mathbb{G}^4 =: \hat{\mathscr{X}} \qquad \text{with } \chi = (u, v)^\intercal \in \mathbb{G}^2$$

$$\hat{\mathscr{L}}_\chi := \mathrm{ColSpan}\,(\Gamma(\chi)) \quad \subsetneq \quad \mathbb{G}^4 =: \hat{\mathscr{X}} \qquad \text{with } \Gamma(\chi) := \begin{pmatrix} g & 1_\mathbb{G} & 1_\mathbb{G} \\ h & g & 1_\mathbb{G} \\ 1_\mathbb{G} & u & g \\ 1_\mathbb{G} & v/g & h \end{pmatrix}\ .$$

Indeed, if $\chi = (u, v)^\intercal$ *is a word in* $\mathscr{L}$ *with witness* $w = (r, b)$*, then*

$$\boldsymbol{\theta}(\chi) = \Gamma(\chi) \bullet \boldsymbol{\lambda}(\chi, w) \qquad \text{with } \boldsymbol{\lambda}(\chi, w) = \begin{pmatrix} r \\ b \\ -rb \end{pmatrix}\ . \tag{3.1}$$

Conversely, if $\boldsymbol{\theta}(\chi) \in \hat{\mathscr{L}}$*, then there exists* $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^\intercal \in \mathbb{Z}_p^3$ *such that* $\boldsymbol{\theta}(\chi) = \Gamma \bullet \boldsymbol{\lambda}$*, i.e.:*

$$\begin{cases} u & = g^{\lambda_1} \\ v & = h^{\lambda_1} \cdot g^{\lambda_2} \\ 1_\mathbb{G} & = u^{\lambda_2} \cdot g^{\lambda_3} \\ 1_\mathbb{G} & = (v/g)^{\lambda_2} \cdot h^{\lambda_3}\ . \end{cases}$$

If we write $(u, v) = (g^r,\ h^r \cdot g^b)$ *(with* $r, b \in \mathbb{Z}_p$*, which is always possible), then the first three equations ensure that* $\lambda_1 = r$*,* $\lambda_2 = b$ *and* $\lambda_3 = -rb$*, while the last equation ensures that* $b(b-1) = 0$*, i.e.,* $b \in \{0, 1\}$*, as it holds that* $(h^r g^b/g)^b h^{-rb} = g^{b(b-1)} = 1$*. Therefore,* $\boldsymbol{\theta}(\chi) \in \hat{\mathscr{L}}$ *if and only if* $\chi \in \mathscr{L}$*, which is exactly what we want.*

We can now define an SPHF as follows:

$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{G}^{1\times 4} \qquad \mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \cdot \Gamma(\chi) \in \mathbb{G}^{1\times 3}$$

$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\chi) = u^{\alpha_1} \cdot v^{\alpha_2} \in \mathbb{G} \qquad \mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(\chi, w) \in \mathbb{G}\,,$$

with λ *being defined as in Equation (3.1). Contrary to all the previous examples,* hp *depend on* χ*. Therefore, the above SPHF is only a GL-SPHF.*

3.1.1.6 Introducing Pairings and Quadratic Equations

All the previous examples were just over cyclic groups. We can actually handle more complex languages over bilinear groups, or even ideal multilinear groups.

Let us consider an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. We naturally extend our operations $\boldsymbol{+}$ and $\bullet$ as follows, for any $x, y \in \mathbb{Z}_p$, $u_1, v_1 \in \mathbb{G}_1$, $u_2, v_2 \in \mathbb{G}_2$ and $u_T, v_T \in \mathbb{G}_T$:

$$\begin{aligned} x \boldsymbol{+} y &= x + y & & & x \bullet y &= x \cdot y \\ u_1 \boldsymbol{+} v_1 &= u_1 \cdot v_1 & & & x \bullet u_1 &= u_1^x \\ u_2 \boldsymbol{+} v_2 &= u_2 \cdot v_2 & & & x \bullet u_2 &= u_2^x \\ u_T \boldsymbol{+} v_T &= u_T \cdot v_T & & u_1 \bullet u_2 = e(u_1, u_2) & x \bullet u_T &= u_T^x \ . \end{aligned}$$

Here is now a simple example.

Example 3.1.4 (encryption of two elements $(\mathsf{M}_1, \mathsf{M}_2)$ satisfying $e(\mathsf{M}_1, \mathsf{M}_2) = E$)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$*, an asymmetric bilinear group of order* p*. The language parameters are* $\mathsf{lpar} := (g_1, h_1, g_2, h_2, E) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_T$ *where* $\mathsf{ek}_1 := (g_1, h_1)$ *and* $\mathsf{ek}_2 := (g_2, h_2)$ *are two random ElGamal encryption keys in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *respectively, while* E *is a group element in* $\mathbb{G}_T$*. We consider the language of ElGamal encryptions of two elements* $\mathsf{M}_1 \in \mathbb{G}_1$ *and* $\mathsf{M}_2 \in \mathbb{G}_2$ *such that* $e(\mathsf{M}_1, \mathsf{M}_2) = E$*:*

$$\begin{aligned} \mathscr{L} = \{ &(u_1, v_1, u_2, v_2)^\intercal \in \mathbb{G}^2 \mid \exists r_1, r_2 \in \mathbb{Z}_p,\ \mathsf{M}_1 \in \mathbb{G}_1,\ \mathsf{M}_2 \in \mathbb{G}_2, \\ &(u_1, v_1, u_2, v_2) = (g^{r_1},\ h^{r_1} \cdot \mathsf{M}_1,\ g^{r_2},\ h^{r_2} \cdot \mathsf{M}_2) \text{ and } e(\mathsf{M}_1, \mathsf{M}_2) = E\} \ . \end{aligned}$$

This language cannot be directly be seen as a DVS, as it is quadratic. However, we can transform $\mathscr{L}$ *into a subspace of the vector space* $\hat{\mathcal{X}} := \mathbb{G}_T^4$ *using the following function* $\boldsymbol{\theta}$ *and the following subspace* $\hat{\mathscr{L}}_\chi$*:*

$$\boldsymbol{\theta}(\chi) := \begin{pmatrix} -u_1 \bullet u_2 \\ u_1 \bullet v_2 \\ v_1 \bullet u_2 \\ v_1 \bullet v_2 \boldsymbol{-} E \end{pmatrix} = \begin{pmatrix} e(u_1, u_2)^{-1} \\ e(u_1, v_2) \\ e(v_1, u_2) \\ e(v_1, v_2)/E \end{pmatrix} \qquad \text{with } \chi = (u_1, v_1, u_2, v_2)^\intercal$$

$$\hat{\mathscr{L}}_\chi := \mathrm{ColSpan}\left(\Gamma(\chi)\right) \quad \subsetneq \quad \mathbb{G}_T^4 =: \hat{\mathcal{X}} \qquad \text{with } \Gamma(\chi) := \begin{pmatrix} g_1 \bullet g_2 & 1_{\mathbb{G}_1} & 1_{\mathbb{G}_2} \\ 1_{\mathbb{G}_T} & g_1 & 1_{\mathbb{G}_2} \\ 1_{\mathbb{G}_T} & 1_{\mathbb{G}_1} & g_2 \\ h_1 \bullet h_2 & h_1 & h_2 \end{pmatrix} \ .$$

Indeed, if $\chi = \{(u_1, v_1, u_2, v_2)\}^\intercal$ *is a word in* $\mathscr{L}$ *with witness* $w = (r_1, r_2, \mathsf{M}_1, \mathsf{M}_2)$*, then*

$$\boldsymbol{\theta}(\chi) = \Gamma(\chi) \bullet \boldsymbol{\lambda}(\chi, w) \qquad \text{with } \boldsymbol{\lambda}(\chi, w) = \begin{pmatrix} -r_1 r_2 \\ r_1 \bullet v_2 \\ r_2 \bullet v_1 \end{pmatrix} = \begin{pmatrix} -r_1 r_2 \\ v_2^{r_1} \\ v_1^{r_2} \end{pmatrix} , \tag{3.2}$$

as:

$$\begin{aligned} \boldsymbol{-} u_1 \bullet u_2 &= (r_1 \bullet g_1) \bullet (r_2 \bullet g_2) = (r_1 r_2) \bullet (g_1 \bullet g_2) \,, \\ u_1 \bullet v_2 &= (r_1 \bullet g_1) \bullet v_2 = (r_1 \bullet v_2) \bullet g_1 \,, \\ v_1 \bullet u_2 &= v_1 \bullet (r_2 \bullet g_2) = (r_2 \bullet v_1) \bullet g_2 \,, \\ v_1 \bullet v_2 \ \boldsymbol{-}\ E &= v_1 \bullet v_2 \ \boldsymbol{-}\ (v_1 \boldsymbol{-} r_1 \bullet h_1) \bullet (v_2 \boldsymbol{-} r_2 \bullet h_2) \\ &= \boldsymbol{-} r_1 \bullet h_1 \bullet r_2 \bullet h_2 \ \boldsymbol{+}\ r_1 \bullet h_1 \bullet v_2 \ \boldsymbol{+}\ v_1 \bullet r_2 \bullet h_2 \\ &= (-r_1 r_2) \bullet (h_1 \bullet h_2) \ \boldsymbol{+}\ (r_1 \bullet h_1) \bullet v_2 \ \boldsymbol{+}\ (r_2 \bullet v_1) \bullet v_2 \ . \end{aligned}$$

Conversely, if $\boldsymbol{\theta}(\chi) \in \hat{\mathscr{L}}$*, then there exists* $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^\mathsf{T} \in \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{G}_2$ *such that* $\boldsymbol{\theta}(\chi) = \Gamma \bullet \boldsymbol{\lambda}$*, which gives four equations (one per row of* Γ*). If we write* $(u_1, v_1, u_2, v_2) = (g^{r_1}, h^{r_1} \cdot \mathsf{M}_1, g^{r_2}, h_2^{r_2} \cdot \mathsf{M}_2)$ *(with* $r_1, r_2 \in \mathbb{Z}_p$*,* $\mathsf{M}_1 \in \mathbb{G}_1$*, and* $\mathsf{M}_2 \in \mathbb{G}_2$*, which is always possible), then the first three equations ensure that* $\lambda_1 = -r_1 r_2$*,* $\lambda_2 = r_1 \bullet v_2$ *and* $\lambda_3 = r_2 \bullet v_1$*, while the last equation ensures that* $e(\mathsf{M}_1, \mathsf{M}_2) = \mathsf{M}_1 \bullet \mathsf{M}_2 = E$*. Therefore,* $\boldsymbol{\theta}(\chi) \in \hat{\mathscr{L}}$ *if and only if* $\chi \in \mathscr{L}$*, which is exactly what we want.*

We can now define an SPHF as follows:

$$\mathsf{hk} := \boldsymbol{\alpha}^\mathsf{T} \xleftarrow{\$} \mathbb{Z}_p^{1\times 4} \qquad \mathsf{hp} := \boldsymbol{\gamma}^\mathsf{T} := \boldsymbol{\alpha}^\mathsf{T} \cdot \Gamma(\chi) \in \mathbb{G}_T \times \mathbb{G}_1 \times \mathbb{G}_2$$
$$\mathsf{H} := \boldsymbol{\alpha}^\mathsf{T} \bullet \boldsymbol{\theta}(\chi) \in \mathbb{G}_T \qquad \mathsf{pH} := \boldsymbol{\gamma}^\mathsf{T} \bullet \boldsymbol{\lambda}(\chi, w) \in \mathbb{G}_T\,,$$

with $\boldsymbol{\lambda}$ *being defined as in Equation (3.2).*

3.1.1.7 Introducing Randomness and Batching

In some cases, we might want to slightly randomize everything to do some batching. Concretely, let us suppose we want to construct an SPHF for the language of tuples of 42 DH pairs. We could construct 42 SPHFs, one for each tuple, and then somehow do their conjunction, e.g., just by multiplying the (projected) hash values of the 42 SPHFs to obtain the resulting (projected) hash values. This was already proposed in [ACP09] and we actually show how to do this algebraically in Section 3.2. However, in both cases, the projection key contains 42 group elements.

Let us show how to get a projection key containing only one group element (and one scalar, which can be generated from the group element, as seen later). We remark that if (u_1, v_1), $\ldots$, (u_{42}, v_{42}) are 42 DH tuples in basis $(g, h) \in \mathbb{G}^2$, then for any $\rho \in \mathbb{Z}_p$:

$$\begin{pmatrix} u \\ v \end{pmatrix} := \begin{pmatrix} u_1 \;+\; \rho \bullet u_2 \;+\; \cdots \;+\; \rho^{41} \bullet u_{42} \\ v_1 \;+\; \rho \bullet v_2 \;+\; \cdots \;+\; \rho^{41} \bullet v_{42} \end{pmatrix} = \begin{pmatrix} u_1 \cdot u_2^{\rho} \cdots u_{42}^{\rho^{41}} \\ v_1 \cdot v_2^{\rho} \cdots v_{42}^{\rho^{41}} \end{pmatrix}$$

is also a DH tuple. Conversely, let us suppose that (u, v) is a DH tuple, and let us prove that (u_1, v_1), $\ldots$, (u_{42}, v_{42}) are also DH tuples, except with probability $41/p$. Let us write $h = g^z$, $u_i = g^{r_i}$, and $v_i = g^{s_i}$, where $z, r_i, s_i \in \mathbb{Z}_p$, for $i \in \{1, \ldots, 42\}$. Let $P(X)$ be the polynomial in $\mathbb{Z}_p[X]$ defined by:

$$P(X) = \sum_{i=1}^{42} (s_i - z r_i) \cdot X^{i-1} \quad .$$

We remark that (u, v) is a DH tuple if and only if $P(\rho) = 0$. As P is a polynomial of degree 41, it has at most 41 roots if it is non-zero. Furthermore, if $P = 0$, then (u_1, v_1), $\ldots$, (u_{42}, v_{42}) are all DH tuples. Therefore, except with probability $41/p$, if (u, v) is a DH tuple, so are (u_1, v_1), $\ldots$, (u_{42}, v_{42}).

To construct an SPHF for the language of 42 DH tuples, we thus just need to construct an SPHF for DDH language on (u, v). The hashing and projection keys for the new language are the original hashing and projection key concatenated with ρ. The new (projected) hash value for $((u_1, v_1), \ldots, (u_{42}, v_{42}))$ is the (projected) hash value for (u, v). The full construction is detailed below.

The resulting SPHF is no more a KV-SPHF, as given ρ (part of the projection key), the adversary can generate an invalid $((u_1, v_1), \ldots, (u_{42}, v_{42}))$ for which it can compute the corresponding hash value: it just generates $r_1, \ldots, r_{42}, s_1, \ldots, s_{42}$ such $P(\rho) = 0$, and sets

$u_i = g^{r_i}$, and $v_i = g^{s_i}$, where $z, r_i, s_i \in \mathbb{Z}_p$, for $i \in \{1, \ldots, 42\}$. This strategy requires to compute the discrete logarithm z of h, but we recall that we consider unbounded adversary for smoothness. Nevertheless, the previous analysis still shows that the resulting SPHF is a $(41/p)$-smooth CS-SPHF, using Proposition 2.1.6.

We could have chosen ρ in a subset of $\mathbb{Z}_p$ of size $41 \cdot 2^{\mathfrak{K}}$. In this case, we would have got $(1/2^{\mathfrak{K}})$-smoothness instead of $(41/p)$-smoothness, with a similar analysis. Furthermore, we just need ρ to be uniform and $\boldsymbol{\gamma}^\intercal$ is already a uniform group element. So ρ can just be extracted from the original projection key $\boldsymbol{\gamma}^\intercal$ for the DDH language, using a deterministic randomness extractor (see Section 2.2.3). Such an extractor exists for some elliptic curves [CFPZ09]. With this last optimization, the size of the projection key and the hashing key is the same for the DDH language, i.e., the language of one DH tuple, than for the language of an arbitrary number of DH tuples.

Example 3.1.5 (ℓ DH tuples)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}, g)$*, with* $\mathbb{G}$ *being a cyclic group of prime order* p *generated by* g*, while the language parameters are* $\mathsf{lpar} := (g, h) \in \mathbb{G}^2$*. We consider the language of* ℓ *DH tuples (for* $\ell \geq 1$*):*

$$\mathscr{L} := \{(u_1, v_1, \ldots, u_\ell, v_\ell) \in \mathbb{G}^{2\ell} \mid \forall i \in \{1, \ldots, \ell\}, \exists r_i \in \mathbb{Z}_p, (u_i, v_i) = (g^{r_i}, h^{r_i})\} \ .$$

To get an efficient SPHF, we first pick a random scalar $\rho \xleftarrow{\$} \{1, \ldots, (\ell-1) \cdot 2^{\mathfrak{K}}\} \subseteq \mathbb{Z}_p$.[1] *Then, we define the following function* $\boldsymbol{\theta}$ *and the following space* $\hat{\mathscr{X}}$ *and subspace* $\hat{\mathscr{L}}$ *(corresponding to the DDH language in basis* (g, h)*):*

$$\boldsymbol{\theta}(\chi, \rho) := \begin{pmatrix} u_1 \ \boldsymbol{+} \ \rho \bullet u_2 \ \boldsymbol{+} \ \cdots \ \boldsymbol{+} \ \rho^{\ell-1} \bullet u_\ell \\ v_1 \ \boldsymbol{+} \ \rho \bullet v_2 \ \boldsymbol{+} \ \cdots \ \boldsymbol{+} \ \rho^{\ell-1} \bullet v_\ell \end{pmatrix} \qquad \text{with } \chi = (u_1, v_1, \ldots, u_\ell, v_\ell) \in \mathbb{G}^{2\ell}$$

$$\hat{\mathscr{L}}_\chi := \mathrm{ColSpan}\,(\Gamma(\chi)) \quad \subsetneq \quad \mathbb{G}^2 =: \hat{\mathscr{X}} \qquad \text{with } \Gamma(\chi) := \begin{pmatrix} g \\ h \end{pmatrix} \ .$$

We can now define a CS-SPHF as follows:

$$\begin{aligned}
&\rho \xleftarrow{\$} \{1, \ldots, 2 \cdot (\ell - 1) \cdot 2^{\mathfrak{K}}\} \subseteq \mathbb{Z}_p \\
&\mathsf{hk} := (\boldsymbol{\alpha}^\intercal, \rho) && \text{with } \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{G}^{1\times 4} \\
&\mathsf{hp} := (\boldsymbol{\gamma}^\intercal, \rho) && \text{with } \boldsymbol{\gamma}^\intercal := \alpha \cdot \Gamma \in \mathbb{G}^{1\times 2} \\
&\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\chi) = \prod_{i=1}^{\ell} u_i^{\rho^{i-1}\alpha_1} \cdot \prod_{i=1}^{\ell} v_i^{\rho^{i-1}\alpha_2} \\
&\mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(w) && \text{with } \lambda(\chi, w) := \sum_{i=1}^{\ell} r_i \rho^{i-1} \ .
\end{aligned}$$

As shown previously, this CS-SPHF is $(1/2^{\mathfrak{K}})$*-smooth.*

Furthermore, as Γ *is independent of* ρ *and due to the way smoothness is proven,* ρ *can instead be defined as* $\mathsf{Ext}(\boldsymbol{\gamma}^\intercal)$*, where* Ext *is a deterministic randomness extractor (see Section 2.2.3). Such an extractor exists for some elliptic curves [CFPZ09].*

3.1.2 Definition

Let us now introduce more formally what a DVS is.

[1] We assume that $(\ell - 1) \cdot 2^{\mathfrak{K}} \leq p$.

3.1.2.1 Graded Rings

As seen in the previous section, it is practical to see cyclic groups, bilinear groups, and even multilinear groups as similar to the field $\mathbb{Z}_p$, with some restrictions on the multiplication. Addition in $\mathbb{Z}_p$ corresponds to the group operation, while multiplication in $\mathbb{Z}_p$ corresponds to the exponentiation of a group element by a scalar or to the pairing of two group elements, or are impossible (e.g., for two group elements of a cyclic group without bilinear map) depending on the case. This enables us to construct *vector spaces* over cyclic groups, bilinear groups, and multilinear groups.

To make this more formal, in Section 4.2.1, we define a new notion called graded ring,[2] which can be seen as a generalization of cyclic groups, bilinear groups, and multilinear groups.

In this chapter, however, we do not formally define graded rings and to avoid technicalities, the group of each element is implicit, and we suppose that above constraints on the multiplications are satisfied. Furthermore, we only consider what will be called graded rings over the finite field $\mathbb{Z}_p$ of prime order p. This basically encompasses cyclic groups $\mathbb{G}$, bilinear groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, and ideal (graded) multilinear groups [GGH13], of prime order p. We use the term *discrete logarithm* of an element similarly to how this term is used with elements of these various groups. The discrete logarithm of a scalar in $\mathbb{Z}_p$ is the scalar itself.

Remark 3.1.6. *For many properties, it is convenient to consider graded ring elements as elements of $\mathbb{Z}_p$ and not to care about the real group of each element, so that all multiplications are allowed. In this case, we say that the property holds "when looking at the discrete logarithms of all the elements". Formally, it means that all the graded rings elements appearing in the property should be replaced by their discrete logarithms.*

The only new notion we will need is the notion of *multiplicatively compatible sub-graded rings*. Informally, $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are two multiplicatively compatible sub-graded rings of some graded ring $\mathfrak{G}$, if it is possible to compute the product $\bullet$ of any element of $\mathfrak{G}_1$ with any element of $\mathfrak{G}_2$, and the result is in $\mathfrak{G}$. Concretely, as a first approach, it is possible to consider that $\mathfrak{G}$ is a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, and that $\mathfrak{G}_1$ and $\mathfrak{G}_2$ corresponds to $\mathbb{G}_1$ and $\mathbb{G}_2$: if $u_1 \in \mathbb{G}_1$ and $u_2 \in \mathbb{G}_2$, $u_1 \bullet u_2 = e(u_1, u_2)$.

3.1.2.2 Diverse Vector Spaces DVSs

Definition 3.1.7. *A diverse vector space (DVS) is defined by a tuple* $\mathcal{V} = (p_{\mathsf{gpar}}, \mathfrak{G}_{\mathsf{gpar}}, (\mathcal{X}_{\mathsf{lpar}}, \mathscr{L}_{\mathsf{lpar}}, \mathscr{R}_{\mathsf{lpar}}, n_{\mathsf{lpar}}, k_{\mathsf{lpar}}, \mathsf{RGen}, \Gamma_{\mathsf{lpar}}, \boldsymbol{\theta}_{\mathsf{lpar}}, \boldsymbol{\lambda}_{\mathsf{lpar}})_{\mathsf{lpar}})_{\mathsf{gpar}}$ *where:*

- $(\mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar}}, (\mathscr{L}_{\mathsf{lpar}})_{\mathsf{lpar}}, (\mathscr{R}_{\mathsf{lpar}})_{\mathsf{lpar}}$ *define a language as in Section 2.3, with implicit setup algorithms* Setup.gpar *and* Setup.lpar*; we recall that* lpar *implicitly contains* gpar*;*
- gpar *corresponds to some global parameter generated by some polynomial-time algorithm* Setup.gpar*;*
- p_{gpar} *is a prime;*
- $\mathfrak{G}_{\mathsf{gpar}}$ *is a graded ring over the finite field* $\mathbb{Z}_{p_{\mathsf{gpar}}}$*; we suppose that* gpar *contains a description of* $\mathfrak{G}_{\mathsf{gpar}}$ *including its order* p_{gpar}*;*

[2] Graded rings were named after graded encodings systems [GGH13] and are unrelated to the mathematical notion of graded rings.

- *n_{lpar} and k_{lpar} are some positive integers;*
- RGen *is a polynomial-time algorithm which takes* lpar *as input and output some element ρ from some set R_{lpar}, implicitly defined by* RGen*;*
- Γ_{lpar} *is a function (computable in polynomial time)*[3] *from the set $\mathcal{X}_{\mathsf{lpar}} \times R_{\mathsf{lpar}}$ to the set of matrices $\mathfrak{G}_{\mathsf{gpar}}^{n_{\mathsf{gpar}} \times k_{\mathsf{lpar}}}$;*
- $\boldsymbol{\theta}_{\mathsf{lpar}}$ *is a function (computable in polynomial time)*[3] *from the set $\mathcal{X}_{\mathsf{lpar}} \times R_{\mathsf{lpar}}$ to the set of column vectors $\mathfrak{G}_{\mathsf{gpar}}^{n_{\mathsf{gpar}}}$;*
- $\boldsymbol{\lambda}_{\mathsf{lpar}}$ *is a function (computable in polynomial time)*[3] *from the set $\mathcal{X}_{\mathsf{lpar}} \times \mathscr{W}_{\mathsf{lpar}} \times R_{\mathsf{lpar}}$ (where $\mathscr{W}_{\mathsf{lpar}}$ is the set of witness for words in $\mathcal{X}_{\mathsf{lpar}}$) to the set of column vectors $\mathfrak{G}_{\mathsf{gpar}}^{k_{\mathsf{gpar}}}$;*

satisfying the following properties: for any security parameter $\mathfrak{K}$, any global parameters $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$*, any language parameters* $\mathsf{lpar} \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$*, and any word $\chi \in \mathcal{X}_{\mathsf{lpar}}$:*

- *the group in which each coordinate of $\boldsymbol{\theta}_{\mathsf{lpar}}(\chi)$ lies (called the* index *of the coordinate, in the formal description of graded rings in Section 4.2.1) is independent of χ;*
- perfect correctness. *For any witness w of χ ($\mathscr{R}_{\mathsf{lpar}}(\chi, w) = 1$), and any $\rho \in R_{\mathsf{lpar}}$:*

$$\boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho) = \Gamma_{\mathsf{lpar}}(\chi, \rho) \bullet \boldsymbol{\lambda}_{\mathsf{lpar}}(\chi, w, \rho)\,;$$

- statistical soundness. *The* DVS *$\mathcal{V}$ is ε-*sound *if, when looking at the discrete logarithms of all the elements (see Remark 3.1.6), when $\chi \in \mathcal{X}_{\mathsf{lpar}} \notin \mathscr{L}_{\mathsf{lpar}}$:*

$$\Pr\left[\boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho) \in \mathrm{ColSpan}\left(\Gamma(\chi, \rho)\right) \,\middle|\, \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right] \leq \varepsilon\,.$$

  The DVS *is* sound *if it is ε-sound, with ε negligible in $\mathfrak{K}$.*

The first property is purely technical and ensures that the hash values of all the words lie in the same group. The second property, perfect correctness, ensures perfect correctness of the associated SPHF defined later. The third property, statistical soundness, is used to prove smoothness of the associated SPHF.

In the sequel, for the sake of simplicity, we often omit the indexes gpar and lpar, when they are clear from the context. When we consider multiple DVSs, global parameters gpar are the same for all the DVSs, but lpar might differ. Please also note that the graded ring $\mathfrak{G}_{\mathsf{gpar}}$ (and its underlying field $\mathbb{Z}_{p_{\mathsf{gpar}}}$) only depends on the global parameters gpar and not on lpar.

Furthermore, we often write:

$$\Gamma := \Gamma(\chi, \rho) \qquad \boldsymbol{\theta} := \boldsymbol{\theta}(\chi, \rho) \qquad \boldsymbol{\lambda} := \boldsymbol{\lambda}(\chi, w) := \boldsymbol{\lambda}(\chi, w, \rho)\,,$$

when χ or w is clear from context, or when the function is constant. We also omit RGen and ρ when they are not used, and we write ($\rho = \perp$ and $R = \{\perp\}$ in this case).

[3] By computable in polynomial time, we mean that there exists a polynomial-time algorithm taking as input lpar and the input of the function (a tuple (χ, ρ) or a tuple (χ, w, ρ)).

In particular, we often write:

$$\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda}) \ .$$

Finally, as in the overview, we write:

$$\hat{\mathcal{X}} := \mathbb{Z}_p^n \qquad\qquad \hat{\mathscr{L}}_\chi := \mathrm{ColSpan}\left(\Gamma(\chi)\right)$$

and we often omit the index χ from $\hat{\mathscr{L}}_\chi$ when it is clear from context.

Let us now illustrate the definition of DVSs on two simple examples seen in the overview.

Example 3.1.8 (DVS for the DDH language)**.** *In this example, we formally give the DVS for the DDH language sketched in Section 3.1.1.1. We use the same global parameters* $\mathsf{gpar} := (p, \mathbb{G}, g)$ *and language parameters* $\mathsf{lpar} := (g, h)$ *as in Example 2.3.1. We recall the DDH language in basis* (g, h)*:*

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (g^r,\ h^r)\} \quad \subsetneq \quad \mathbb{G}^2 = \mathcal{X} \ .$$

We define the DVS for the DDH language as follows:

$$n := 2 \qquad\qquad k := 1$$

$\mathsf{RGen}(\mathsf{lpar})$ *outputs* $\perp$

$$\Gamma(\chi, \rho) := \Gamma := \begin{pmatrix} g \\ h \end{pmatrix}$$

$$\boldsymbol{\theta}(\chi, \rho) := \boldsymbol{\theta} := \begin{pmatrix} u \\ v \end{pmatrix} \qquad\qquad \boldsymbol{\lambda}(\chi, w, \rho) := \begin{pmatrix} r \end{pmatrix} \ ,$$

where:

$$\chi = (u, v) \qquad\qquad w = r \ .$$

For the sake of completeness, we explicitly defined RGen *to output* $\perp$ *in this example. In the remaining of the thesis, we just do not define* RGen *when we are not using it.*

Example 3.1.9 (DVS for ElGamal)**.** *In this example, we formally give the DVS for the ElGamal language sketched in Section 3.1.1.3. We use the same global parameters* $\mathsf{gpar} = (p, \mathbb{G}, g)$ *and language parameters* $\mathsf{lpar} := \mathsf{ek} := (g, h)$ *as in Section 3.1.1.3. We recall the ElGamal language under encryption key* ek*:*

$$\mathscr{L} = \{(\mathsf{m},\ \mathsf{c} = (u, v)) \in \mathscr{M} \times \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ (u, v) = (g^r,\ h^r \cdot \mathcal{G}(\mathsf{m}))\} \ .$$

We define the DVS for this language as follows:

$$n := 2 \qquad\qquad k := 1$$

$$\Gamma(\chi) := \Gamma := \begin{pmatrix} g \\ h \end{pmatrix}$$

$$\boldsymbol{\theta}(\chi) := \begin{pmatrix} u \\ v/\mathcal{G}(\mathsf{m}) \end{pmatrix} \qquad\qquad \boldsymbol{\lambda}(\chi, w) := \begin{pmatrix} r \end{pmatrix} \ ,$$

where:

$$\chi = (\mathsf{m}, \mathsf{c} = (u, v)) \qquad\qquad w = r \ .$$

3.1.2.3 The SPHF Associated to a DVS, GL-DVS, CS-DVS, and KV-DVS

Construction 3.1.10 (PHF from DVS)**.** *Let* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a DVS. Then, we construct a PHF as follows:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *picks a random row vector* $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}$*, generates a random element* $\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})$ *and outputs the hashing key* $\mathsf{hk} := (\boldsymbol{\alpha}^\intercal, \rho)$*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs the projection key* $\mathsf{hp} := (\boldsymbol{\gamma}^\intercal, \rho)$*, where* $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$ *and*
$$\boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma_{\mathsf{lpar}}(\chi, \rho) \in \mathfrak{G}^{1\times k}\,;$$
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs the hash value*
$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho) \in \mathfrak{G}\,,$$
where $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$*;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *outputs the projected hash value*
$$\mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}_{\mathsf{lpar}}(\chi, w, \rho) \in \mathfrak{G}\ .$$

As seen in the examples:

- in the general case, we say that $\mathcal{V}$ is a *GL diverse vector space* (GL-DVS) and we get a GL-SPHF;
- if Γ is independent of χ, we say that $\mathcal{V}$ is a *CS diverse vector space* (CS-DVS) and we get a CS-SPHF;
- if Γ is independent of χ and ρ is not used ($\rho = \perp$ and $R = \{\perp\}$), then we say that $\mathcal{V}$ is a *KV diverse vector space* (KV-DVS) and we get a KV-SPHF. We remark that, in this case, the DVS $\mathcal{V}$ is necessarily 0-sound (i.e., perfectly sound).

Actually, for the last case, we just need to suppose that the DVS $\mathcal{V}$ is perfectly sound. But this is equivalent to the requirement that ρ is not used, as if $\mathcal{V}$ is perfectly sound, we can just fix ρ to an arbitrary value. Therefore, we think that it makes more sense to consider the (apparently stronger) condition $R = \{\perp\}$.

More formally, we have the following security results.

Theorem 3.1.11 (GL-SPHF from GL-DVS)**.** *Let* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be an* ε*-sound DVS or GL-DVS, then the PHF described in Construction 3.1.10 is an* ε*-smooth GL-SPHF.*

Corollary 3.1.12 (CS-SPHF from CS-DVS)**.** *Let* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be an* ε*-sound DVS such that the matrix* Γ *does not depend on* χ *(i.e.,* $\mathcal{V}$ *is a CS-DVS), then the PHF described in Construction 3.1.10 described above is an* ε*-smooth CS-SPHF.*

Corollary 3.1.13 (KV-SPHF from KV-DVS)**.** *Let* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a perfectly sound (0-sound) DVS such that the matrix* Γ *does not depend on* χ *and* $R = \{\perp\}$ *(i.e.,* $\mathcal{V}$ *is a KV-DVS), then the PHF described in Construction 3.1.10 is a perfectly smooth (or 0-smooth) KV-SPHF.*

Proof of Theorem 3.1.11. We need to prove correctness and smoothness.

Prefect correctness. When $\mathscr{R}(\chi, w) = 1$, $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho) \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$, $\mathsf{hp} = (\boldsymbol{\gamma}^\intercal, \rho) \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$, $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$, and $\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$, we have:

$$\begin{aligned}
\mathsf{H} &= \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\chi, \rho) && \text{by definition of } \mathsf{Hash} \\
&= \boldsymbol{\alpha}^\intercal \bullet (\Gamma(\chi, \rho) \bullet \boldsymbol{\lambda}(\chi, w, \rho)) && \text{by correctness of the DVS} \\
&= (\boldsymbol{\alpha}^\intercal \bullet \Gamma(\chi, \rho)) \bullet \boldsymbol{\lambda}(\chi, w, \rho) && \text{by associativity of } \bullet \\
&= \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(\chi, w, \rho) && \text{by definition of } \mathsf{ProjKG} \\
&= \mathsf{pH} && \text{by definition of } \mathsf{ProjHash}
\end{aligned}$$

This proves perfect correctness of the SPHF.

Smoothness. Let $\chi \in \mathcal{X} \setminus \mathscr{L}$. Until the end of the proof, we are only looking at the discrete logarithms of all the group elements we are considering. Let us suppose that ρ is such that $\boldsymbol{\theta} = \boldsymbol{\theta}(\chi, \rho)$ is linearly independent of the columns of $\Gamma = \Gamma(\chi, \rho)$, which happens with probability at least $1 - \varepsilon$, by ε-soundness of the DVS $\mathcal{V}$.

Let us now prove that, in this case, the hash value $\mathsf{H} = \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$ is uniformly random given only the projection key $\mathsf{hp} = (\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma, \rho)$. Let us fix a row vector $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times n}$ and its associated row vector $\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma$. Let H be a scalar in $\mathbb{Z}_p$.[4] Let $S_{\boldsymbol{\gamma}^\intercal, \mathsf{H}}$ be the set of vectors $\boldsymbol{\alpha}'^\intercal$ corresponding to the vector $\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma$ and to the hash value H, i.e., $S_{\boldsymbol{\alpha}^\intercal, \mathsf{H}}$ is the set of solutions $\boldsymbol{\alpha}'^\intercal \in \mathbb{Z}_p^{1\times n}$ of the affine system:

$$\boldsymbol{\alpha}'^\intercal \bullet \begin{pmatrix} \Gamma & \boldsymbol{\theta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\gamma}^\intercal & \mathsf{H} \end{pmatrix} . \tag{3.3}$$

As $\boldsymbol{\theta} \notin \mathrm{ColSpan}(\Gamma)$, there exists a row vector $\boldsymbol{x}^{*\intercal} \in \mathbb{Z}_p^{1\times n}$ satisfying $\boldsymbol{x}^{*\intercal} \bullet \Gamma = \mathbf{0}$ and $\boldsymbol{x}^{*\intercal} \bullet \boldsymbol{\theta} = \mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$. And $\boldsymbol{\alpha}^{*\intercal} = \boldsymbol{\alpha}^\intercal + \boldsymbol{x}^{*\intercal}$ is a particular solution of Equation (3.3), as:

$$\begin{aligned}
\boldsymbol{\alpha}^{*\intercal} \bullet \begin{pmatrix} \Gamma & \boldsymbol{\theta} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{\alpha}^\intercal \bullet \Gamma + \boldsymbol{x}^{*\intercal} \bullet \Gamma & \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} + \boldsymbol{x}^{*\intercal} \bullet \boldsymbol{\theta} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{\gamma}^\intercal + 0 & \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} + (\mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}) \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{\gamma}^\intercal & \mathsf{H} \end{pmatrix} .
\end{aligned}$$

Therefore, we have:

$$S_{\boldsymbol{\gamma}^\intercal, \mathsf{H}} = \left\{ \boldsymbol{\alpha}^\intercal + \boldsymbol{x}^{*\intercal} + \boldsymbol{x}^\intercal \;\middle|\; \boldsymbol{x} \in \ker \begin{pmatrix} \Gamma & \boldsymbol{\theta} \end{pmatrix}^\intercal \right\} , \tag{3.4}$$

and the size of $S_{\boldsymbol{\gamma}^\intercal, \mathsf{H}}$ is independent of H and $\boldsymbol{\gamma}^\intercal$. This means that there is the same number of hashing keys hk, corresponding to any given pair $(\mathsf{hp}, \mathsf{H})$ of a possible projection key hp and a hash value H.

This implies that H looks uniformly random when only given hp, when ρ is such that $\boldsymbol{\theta} \notin \mathrm{ColSpan}(\Gamma)$, which happens with probability at least $1 - \varepsilon$ (and when $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}$). We conclude using Proposition 2.1.6. □

Proof of Corollary 3.1.12. This is a straightforward corollary of Theorem 3.1.11, as the smoothness definition for CS-SPHF is the same as for GL-SPHF. The only difference is that hp does not depend on χ, as Γ and ρ do not depend on χ. □

[4] We look at the discrete logarithm, so everything is a scalar. Otherwise, we would need to take a graded ring element of correct index.

Proof of Corollary 3.1.13. As the DVS $\mathcal{V}$ is perfectly sound, Corollary 3.1.12 ensures that the resulting SPHF is perfectly GL/CS-smooth. Thanks to Remark 2.5.4, this means that the SPHF is perfectly KV-smooth. □

Remark 3.1.14. *Similarly to what we did in Example 3.1.5, if Γ is independent of ρ and if there exists a deterministic randomness extractor for the distribution of the random elements $\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma$ (with $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}$)[5] with large enough output to be used as random coins for* RGen*, then we can set:*

$$\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar}; \mathsf{Ext}(\boldsymbol{\gamma}^\intercal)) \ .$$

Looking at the proof of Theorem 3.1.11, this clearly defines a GL-SPHF or CS-SPHF.

In many applications, if there is not enough entropy in $\boldsymbol{\gamma}^\intercal$, we can use a pseudorandom number generator after the deterministic extractor. This is however no more a real SPHF, as smoothness becomes computational.

3.2 Conjunctions and Disjunctions

Now that we have formally defined DVSs, we can start looking at important generic combinations of DVSs: conjunctions and disjunctions.

3.2.1 Conjunctions

We define the conjunction of two languages $\mathscr{L}_1 \subseteq \mathcal{X}_1$ and $\mathscr{L}_2 \subseteq \mathcal{X}_2$ as the language:

$$\mathscr{L} := \mathscr{L}_1 \times \mathscr{L}_2 = \{(x_1, x_2) \in \mathcal{X}_1 \times \mathcal{X}_2 \mid x_1 \in \mathscr{L}_1 \text{ and } x_2 \in \mathscr{L}_2\} \quad \subseteq \quad \mathcal{X}_1 \times \mathcal{X}_2 =: \mathcal{X} \ .$$

More formally:

$$\mathscr{R}(x = (x_1, x_2),\ w = (w_1, w_2)) = 1 \qquad \Longleftrightarrow \qquad \mathscr{R}_1(x_1, w_1) = 1 \text{ and } \mathscr{R}_2(x_2, w_2) = 1 \ . \quad (3.5)$$

This is a generalization of the classical notion of conjunction: if $\mathcal{X}_1 = \mathcal{X}_2$, $(x_1, x_1) \in \mathscr{L}$ if and only if $x_1 \in \mathscr{L}_1 \cap \mathscr{L}_2$ (or in other words, x_1 is in $\mathscr{L}_1$ and in $\mathscr{L}_2$).

Global parameters gpar are supposed to be the same and to be generated the same way for $\mathscr{L}_1$, $\mathscr{L}_2$, and $\mathscr{L}$: $\mathsf{Setup.gpar}_1 = \mathsf{Setup.gpar}_2 = \mathsf{Setup.gpar}$. Parameters lpar for $\mathscr{L}$ are just the concatenation of parameters lpar_1 for $\mathscr{L}_1$ and lpar_2 for $\mathscr{L}_2$. More precisely, on input gpar, Setup.lpar generates $(\mathsf{lpar}_1, \mathsf{ltrap}_1) \xleftarrow{\$} \mathsf{Setup.lpar}_1(\mathsf{gpar})$ and $(\mathsf{lpar}_2, \mathsf{ltrap}_2) \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar})$ and outputs $(\mathsf{lpar} := (\mathsf{lpar}_1, \mathsf{lpar}_2),\ \mathsf{ltrap} := (\mathsf{ltrap}_1, \mathsf{ltrap}_2))$.

Let $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two DVSs over the same graded ring $\mathfrak{G}$. Let us first give the intuition of the construction of a DVS $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ for the conjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$, in the simple case where there is no ρ. We recall that a DVS is a way to represent a language $\mathscr{L}' \subseteq \mathcal{X}'$ as a subspace $\hat{\mathscr{L}}'$ of some vector space $\hat{\mathcal{X}}'$, through a map $\boldsymbol{\theta}'$ from $\mathcal{X}$ to $\hat{\mathcal{X}}$. To construct $\mathcal{V}$, we can first map pairs $x = (x_1, x_2) \in \mathcal{X}$ to:

$$\boldsymbol{\theta}(x) = \begin{pmatrix} \boldsymbol{\theta}_1(x_1) \\ \boldsymbol{\theta}_2(x_2) \end{pmatrix} \in \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2 =: \hat{\mathcal{X}} \ .$$

[5] When Γ depends on x, this extractor has to work with any x. We remark that if the first column of Γ is never zero, the first coordinate of $\boldsymbol{\gamma}^\intercal$ is a uniform random group element. If more than one coordinate of $\boldsymbol{\gamma}^\intercal$ is used, linear relations between coordinates have to be taken into account.

Then, we remark that:

$$\mathfrak{x} \in \mathscr{L} \quad \Longleftrightarrow \quad \boldsymbol{\theta}(\mathfrak{x}) \in \hat{\mathscr{L}}_1 \times \hat{\mathscr{L}}_2 =: \hat{\mathscr{L}} \ .$$

Furthermore, the set $\hat{\mathscr{L}} = \hat{\mathscr{L}}_1 \times \hat{\mathscr{L}}_2$ is a subspace of the vector space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, as a cartesian product of the subspace $\hat{\mathscr{L}}_1$ of $\mathcal{X}_1$ and the subspace $\hat{\mathscr{L}}_2$ of $\mathcal{X}_2$. Therefore, this defines a DVS.

Formally, we have the following construction in the general case.

Construction 3.2.1 (conjunction of two DVSs)**.** *Let* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs over the same graded ring* $\mathfrak{G}$*. The conjunction DVS* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *is defined as follows:*

$$\begin{aligned}
&\mathscr{L} := \mathscr{L}_1 \times \mathscr{L}_2 && \mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \\
&\mathscr{R} \text{ is defined as in Equation (3.5)} \\
&n := n_1 + n_2 && k := k_1 + k_2 \\
&\mathsf{RGen}(\mathsf{lpar}) \text{ outputs } \rho := (\rho_1, \rho_2) \text{ where } \rho_1 \xleftarrow{\$} \mathsf{RGen}_1(\mathsf{lpar}_1) \text{ and } \rho_2 \xleftarrow{\$} \mathsf{RGen}_2(\mathsf{lpar}_2) \\
&\Gamma(\mathfrak{x}, \rho) := \begin{pmatrix} \Gamma_1(\mathfrak{x}_1, \rho_1) & \mathbf{0} \\ \mathbf{0} & \Gamma_2(\mathfrak{x}_2, \rho_2) \end{pmatrix} \\
&\boldsymbol{\theta}(\mathfrak{x}, \rho) := \begin{pmatrix} \boldsymbol{\theta}_1(\mathfrak{x}_1, \rho_1) \\ \boldsymbol{\theta}_2(\mathfrak{x}_2, \rho_2) \end{pmatrix} && \boldsymbol{\lambda}(\mathfrak{x}, \mathfrak{w}, \rho) := \begin{pmatrix} \boldsymbol{\lambda}_1(\mathfrak{x}_1, \mathfrak{w}_1, \rho_1) \\ \boldsymbol{\lambda}_2(\mathfrak{x}_2, \mathfrak{w}_2, \rho_2) \end{pmatrix} ,
\end{aligned}$$

where:

$$\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \qquad \mathfrak{w} = (\mathfrak{w}_1, \mathfrak{w}_2) \ .$$

We have the following lemma. Its proof is a simple variant of the proof of the slightly more complicated Lemma 3.2.6.

Lemma 3.2.2. *Let* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *be two DVSs. If* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *are* ε_1*-sound and* ε_2*-sound respectively, then the conjunction DVS* $\mathcal{V}$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *defined in Construction 3.2.1 is a* $(\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2)$*-sound DVS.*

Historical note 3.2.3. *Conjunctions of SPHFs were already proposed in [ACP09], but were not algebraic, because they used an* xor *operation. Furthermore they were performed at the SPHF level, while we are doing conjunctions of DVSs. Conjunctions of DVSs were formally introduced in [ABP15c] but were implicit in many previous papers.*

3.2.2 Disjunctions

We define the disjunction of two languages $\mathscr{L}_1 \subseteq \mathcal{X}_1$ and $\mathscr{L}_2 \subseteq \mathcal{X}_2$ as the language:

$$\mathscr{L} := (\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2) = \{(\mathfrak{x}_1, \mathfrak{x}_2) \in \mathcal{X}_1 \times \mathcal{X}_2 \mid \mathfrak{x}_1 \in \mathscr{L}_1 \text{ or } \mathfrak{x}_2 \in \mathscr{L}_2\} \ \subseteq \ \mathcal{X}_1 \times \mathcal{X}_2 =: \mathcal{X} \ .$$

More formally:

$$\mathscr{R}(\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2),\ \mathfrak{w} = (\mathfrak{w}_1, \mathfrak{w}_2)) = 1 \qquad \Longleftrightarrow \qquad \mathscr{R}_1(\mathfrak{x}_1, \mathfrak{w}_1) = 1 \text{ or } \mathscr{R}_2(\mathfrak{x}_2, \mathfrak{w}_2) = 1 \ . \quad (3.6)$$

If w_1 is a witness for x_1, $w = (w_1, \perp)$ is a witness for $x = (x_1, x_2)$, while if w_2 is a witness for x_2, $w = (\perp, w_2)$ is a witness for x. This is a generalization of the classical notion of disjunction: if $\mathcal{X}_1 = \mathcal{X}_2$, $(x_1, x_1) \in \mathscr{L}$ if and only if $x_1 \in \mathscr{L}_1 \cup \mathscr{L}_2$ (or in other words, x_1 is in $\mathscr{L}_1$ or in $\mathscr{L}_2$).

Global parameters and language parameters are defined as in Section 3.2.1.

We remark that contrary to conjunctions, if $\mathcal{X}_1$ and $\mathcal{X}_2$ are two vector spaces and $\mathscr{L}_1$ and $\mathscr{L}_2$ are subspaces of these two vector spaces respectively, then $\mathcal{X}$ is a vector space, but $\mathscr{L}$ is not necessarily a subspace of $\mathcal{X}$. Actually, $\mathscr{L}$ is a subspace of $\mathcal{X}$ if and only if $\mathscr{L}_1 = \mathcal{X}_1$ and $\mathscr{L}_2 = \mathcal{X}_2$. Even worse, the vector space spanned by $\mathscr{L}$ is $\mathcal{X}$, so constructions need to be much more subtle.

We propose two constructions:

- one for GL-DVSs which has the advantage of working with the same graded ring as the original DVSs but which yield a GL-DVS even if the original DVSs are KV-DVSs;
- and one for CS-DVSs and KV-DVSs which preserve their CS/KV character but which requires multiplicatively compatible sub-graded rings. Concretely, this means that we require a bilinear map between elements of the first DVS and elements of the second DVS.

3.2.2.1 Disjunctions of GL-DVSs

The construction is not really difficult but is hard to explain in English. One maybe not completely helpful way to describe it is to say that we move $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ inside the matrix Γ and we use the first row of Γ as a way to "select" $\boldsymbol{\theta}_1$ or $\boldsymbol{\theta}_2$. Let us now give the formulas which should make this intuition clear.

Construction 3.2.4 (GL disjunction)**.** *Let* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *two DVSs over the same graded ring* $\mathfrak{G}$*. We define the GL disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *as follows:*

$$\mathscr{L} := (\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2) \qquad \mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2$$

$\mathscr{R}$ *is defined as in Equation* (3.6)

$$n := n_1 + n_2 + 1 \qquad k := k_1 + k_2 + 2$$

$\mathsf{RGen}(\mathsf{lpar})$ *outputs* $\rho := (\rho_1, \rho_2)$ *where* $\rho_1 \xleftarrow{\$} \mathsf{RGen}_1(\mathsf{lpar}_1)$ *and* $\rho_2 \xleftarrow{\$} \mathsf{RGen}_2(\mathsf{lpar}_2)$

$$\Gamma(x, \rho) := \begin{pmatrix} \mathbf{0}_{k_1}^\intercal & 1_{\mathbb{Z}_p} & \mathbf{0}_{k_2}^\intercal & 1_{\mathbb{Z}_p} \\ \Gamma_1 & \boldsymbol{\theta}_1 & 0_{n_1 \times k_2} & \mathbf{0}_{n_1} \\ 0_{n_2 \times k_1} & \mathbf{0}_{n_2} & \Gamma_2 & \boldsymbol{\theta}_2 \end{pmatrix}$$

$$\boldsymbol{\theta}(\mathcal{X}, \rho) := \begin{pmatrix} [\top, -1] \\ [\top, 0] \\ \vdots \\ [\top, 0] \end{pmatrix} \in \mathfrak{G}_{\top}^{n} \qquad \boldsymbol{\lambda}(\mathcal{X}, \mathcal{W}, \rho) := \begin{cases} \begin{pmatrix} \boldsymbol{\lambda}_1 \\ -1_{\mathbb{Z}_p} \\ \mathbf{0}_{k_2} \\ 0 \\ \mathbf{0}_{k_1} \\ 0 \\ \boldsymbol{\lambda}_2 \\ -1_{\mathbb{Z}_p} \end{pmatrix} & \text{if } \mathscr{R}(\mathcal{X}_1, \mathcal{W}_1) = 1 \\ & \\ & \text{else, if } \mathscr{R}(\mathcal{X}_2, \mathcal{W}_2) = 1 \end{cases} ,$$

where:

$$\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2) \qquad \mathcal{W} = (\mathcal{W}_1, \mathcal{W}_2) ,$$

and $\mathfrak{G}_{\top}$ is the group of the elements of maximal index (namely the group $\mathbb{G}$ if $\mathfrak{G} = (p, \mathbb{G})$ is a cyclic group and the group $\mathbb{G}_T$ if $\mathfrak{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is a bilinear group), while $[\top, x]$ represents the element of discrete logarithm $x \in \mathbb{Z}_p$ in $\mathfrak{G}_{\top}$.

It is important to remark that when $\mathcal{X}_1 \in \mathscr{L}_1$ and $\mathcal{X}_2 \in \mathscr{L}_2$, there exist at least two witnesses for $\mathcal{X}$: $(\mathcal{W}_1, \perp)$ and $(\perp, \mathcal{W}_2)$. Both witnesses might yield a different projected hash value pH, if the projection key if not honestly generated.

Before proving the soundness of the construction, let us give an example of GL disjunction of two DVSs.

Example 3.2.5 (GL disjunction of two DDH languages)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}, g)$*, with $\mathbb{G}$ being a cyclic group of prime order p generated by g, while the language parameters are* $\mathsf{lpar} := (g_1, h_1, g_2, h_2) \in \mathbb{G}^4$*. We consider the language of the tuples (u_1, v_1, u_2, v_2) such that (u_1, v_1) is a DH tuple in basis (g_1, h_1) or (u_2, v_2) is a DH tuple in basis (g_2, h_2):*

$$\mathscr{L} := \{(u_1, v_1, u_2, v_2) \in \mathbb{G}^4 \mid \exists (r_1, r_2) \in \mathbb{Z}_p^2,\ (u_1, v_1) = (g^{r_1}, h^{r_1}) \text{ or } (u_2, v_2) = (g^{r_2}, h^{r_2})\} .$$

In other words, if $(u_1, v_1, u_2, v_2) \in \mathscr{L}$, either (u_1, v_1) is an ElGamal ciphertext of $1_{\mathbb{G}}$ with encryption key $\mathsf{ek}_1 := (g_1, h_1)$, or (u_2, v_2) is an ElGamal ciphertext of $1_{\mathbb{G}}$ with encryption key $\mathsf{ek}_2 := (g_2, h_2)$, or both (u_1, v_1) and (u_2, v_2) are ElGamal ciphertexts of $1_{\mathbb{G}}$.

This language is the disjunction of the DDH languages in basis (g_1, h_1) and (g_2, h_2) respectively. Applying Construction 3.2.4 to Example 3.1.8, we get the following DVS for $\mathscr{L}$:

$$n := 5 \qquad k := 4$$

$$\Gamma(\mathcal{X}) := \begin{pmatrix} 0 & g & 0 & g \\ g_1 & u_1 & 0 & 0 \\ h_1 & v_1 & 0 & 0 \\ 0 & 0 & g_2 & u_2 \\ 0 & 0 & h_2 & v_2 \end{pmatrix}$$

$$\boldsymbol{\theta}(\chi) := \begin{pmatrix} g^{-1} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \end{pmatrix} \qquad \boldsymbol{\lambda}(\chi, w) := \begin{cases} \begin{pmatrix} r_1 \\ -1_{\mathbb{Z}_p} \\ 0 \\ 0 \\ 0 \\ 0 \\ r_2 \\ -1_{\mathbb{Z}_p} \end{pmatrix} & \text{if } (u_1, v_1) = (g^{r_1}, h^{r_1}) \\ & \\ & \text{otherwise, if } (u_2, v_2) = (g^{r_2}, h^{r_2}) \end{cases} ,$$

where:

$$\chi = (u_1, v_1, u_2, v_2) \qquad\qquad w = (r_1, r_2) \ .$$

Let us now prove the soundness of the construction.

Lemma 3.2.6. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs. If* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *are* ε_1*-sound and* ε_2*-sound respectively, then the GL disjunction DVS* $\mathcal{V}$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *defined in Construction 3.2.4 is a* $(\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2)$*-sound DVS.*

Proof. Correctness is straightforward. Let us prove soundness. For that, we are only looking at the discrete logarithms of all the group elements we are considering.

We need to prove that if $\chi = (\chi_1, \chi_2) \in \mathcal{X} \setminus \mathscr{L}$ (i.e., $\chi_1 \in \mathcal{X}_1 \setminus \mathscr{L}_1$ and $\chi_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$), then $\boldsymbol{\theta} = \boldsymbol{\theta}(\chi, \rho)$ is linearly independent of the columns of $\Gamma = \Gamma(\chi, \rho)$, with probability at least $\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$ when $\rho_1 \xleftarrow{\$} \mathsf{RGen}_1(\mathsf{lpar}_1)$, $\rho_2 \xleftarrow{\$} \mathsf{RGen}_2(\mathsf{lpar}_2)$, and $\rho := (\rho_1, \rho_2)$.

By ε_1-soundness of $\mathcal{V}_1$ and ε_2-soundness of $\mathcal{V}_2$-soundness, we know that $\boldsymbol{\theta}_1(\chi_1, \rho_1)$ and $\boldsymbol{\theta}_2(\chi_2, \rho_2)$ are linearly independent of the columns of $\Gamma_1(\chi_1, \rho_1)$ and $\Gamma_2(\chi_2, \rho)$ respectively with probability at least $1 - \varepsilon_1$ and $1 - \varepsilon_2$. As these two events are independent, both happen at the same time with probability at least $(1 - \varepsilon_1) \cdot (1 - \varepsilon_2) = 1 - \varepsilon_1 - \varepsilon_2 + \varepsilon_1\varepsilon_2$.

Let us now suppose we are in the case where both these events happen. We just need to prove that in this case, $\boldsymbol{\theta}$ is linearly independent of the columns of Γ. By contradiction, let us suppose that there exists a vector $\boldsymbol{\lambda} \in \mathbb{Z}_p^k$ such that $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$. We necessarily have $\lambda_{k_1+1} \neq 0$ or $\lambda_{k_1+k_2+1} \neq 0$ (or both), due to the first row of the matrix Γ, as the $(k_1 + 1)$-th column of Γ is $(1_{\mathbb{Z}_p}, \boldsymbol{\theta}_1^\intercal, \mathbf{0}^\intercal)^\intercal$ and the $(k_1 + k_2 + 1)$-th column of Γ is its last column $(1_{\mathbb{Z}_p}, \mathbf{0}^\intercal, \boldsymbol{\theta}_2^\intercal)^\intercal$. If $\lambda_{k_1+1} \neq 0$, we get that $\boldsymbol{\theta}_1$ is in the column span of Γ_1, as we have:

$$\boldsymbol{\theta}_1 = \Gamma_1 \bullet \boldsymbol{\lambda}' \qquad \text{with } \boldsymbol{\lambda}' = -(\lambda_i)_{i=1,\ldots,k_1} \ / \ \lambda_{k_1+1} \ . \tag{3.7}$$

This is impossible. Similarly, if $\lambda_{k_1+k_2+2} \neq 0$, we get that $\boldsymbol{\theta}_2$ is in the column span of Γ_2 which is also impossible. That concludes the proof. □

Historical note 3.2.7. *Similarly to conjunctions of SPHFs, disjunctions of GL-SPHFs were already proposed in [ACP09], but were not algebraic at all. Furthermore, converting them to disjunctions of DVSs is not completely straightforward and was officially done for the first time in [BCPW15].*

3.2.2.2 Disjunctions of CS-DVSs and KV-DVSs

In the previous construction, the new matrix Γ necessarily depends on the word χ, as it embeds $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$. To construct disjunction DVSs for CS-DVS and KV-DVS which preserve the KV/CS character, we cannot use the same idea. Let us try to use the same trick as for conjunctions and define:

$$\boldsymbol{\theta}(\chi) = \begin{pmatrix} \boldsymbol{\theta}_1(\chi_1) \\ \boldsymbol{\theta}_2(\chi_2) \end{pmatrix} \in \hat{\mathcal{X}} =: \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2 \ .$$

To get a DVS for the disjunction, we would then need to set:

$$\hat{\mathscr{L}} = (\hat{\mathscr{L}}_1 \times \hat{\mathcal{X}}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2) \ .$$

Unfortunately, in that case, $\hat{\mathscr{L}}$ is not a subspace of $\hat{\mathcal{X}}$ (except if $\hat{\mathscr{L}}_1 = \hat{\mathcal{X}}_1$ and $\hat{\mathscr{L}}_2 = \hat{\mathcal{X}}_2$, which is not a really interesting case) and even worse, $\mathrm{Span}\left(\hat{\mathscr{L}}\right) = \hat{\mathcal{X}}$. To solve this issue, we use the tensor product $\otimes$ instead of the cartesian product $\times$, and $\boldsymbol{\theta}$ becomes: $\boldsymbol{\theta} = \boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2$. Intuitively, this adds more dimension which makes everything work.

Tensor product of vector spaces over graded rings. Before formally showing our construction, let us briefly recall notation and some properties for tensor product and adapt them to vector spaces over graded rings. Let $\mathfrak{G}_1$ and $\mathfrak{G}_2$ be two multiplicatively compatible sub-graded rings of $\mathfrak{G}$. Let V_1 be a n_1-dimensional vector space over $\mathfrak{G}_1$ and V_2 be a n_2-dimensional vector space over $\mathfrak{G}_2$. Let $(\boldsymbol{e}_{1,i})_{i=1,\dots,n_1}$ and $(\boldsymbol{e}_{2,i})_{i=1,\dots,n_2}$ be bases of V_1 and V_2 respectively. Then the *tensor product* V of V_1 and V_2, denoted $V = V_1 \otimes V_2$ is the $n_1 n_2$-dimensional vector space over $\mathfrak{G}$ generated by the free family $(\boldsymbol{e}_{1,i} \otimes \boldsymbol{e}_{2,j})_{\substack{i=1,\dots,n_1 \\ j=1,\dots,n_2}}$. The operator $\otimes$ is bilinear, and if $\boldsymbol{u} = \sum_{i=1}^{n_1} u_i \bullet \boldsymbol{e}_{1,i}$ and $\boldsymbol{v} = \sum_{j=1}^{n_2} v_j \bullet \boldsymbol{e}_{2,j}$, then:

$$\boldsymbol{u} \otimes \boldsymbol{v} = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (u_i \bullet v_j) \bullet (\boldsymbol{e}_{1,i} \otimes \boldsymbol{e}_{2,j}) \ .$$

More generally, we can define the tensor product of two matrices $M \in \mathfrak{G}_1^{k \times m}$ and $M' \in \mathfrak{G}_2^{k' \times m'}$, $T = M \otimes M' \in \mathfrak{G}^{kk' \times mm'}$ by

$$T_{(i-1)k'+i',(j-1)m'+j'} = M_{i,j} \bullet M'_{i',j'} \qquad \text{for} \begin{cases} i = 1, \dots, k \,, \\ i' = 1, \dots, k' \,, \\ j = 1, \dots, m \,, \\ j' = 1, \dots, m' \ . \end{cases}$$

In other words, T is the following matrix by blocks:

$$T = \begin{pmatrix} M_{1,1} \bullet M' & \dots & M_{1,m} \bullet M' \\ \vdots & & \vdots \\ M_{k,1} \bullet M' & \dots & M_{k,m} \bullet M' \end{pmatrix} \ .$$

And if $M \in \mathfrak{G}_1^{k \times m}$, $M' \in \mathfrak{G}_2^{k' \times m'}$, $N \in \mathfrak{G}_1^{m \times n}$, the we have:

$$(M \otimes M') \bullet (N \otimes N') = (M \bullet N) \otimes (M' \bullet N') \ . \tag{3.8}$$

This equality is essential: most of our proofs relying on tensor products use it.

Construction. Let us now formally show the construction.

Construction 3.2.8 (CS/KV disjunction)**.** *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *two DVSs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. In particular, we suppose that global parameters for* $\mathscr{L}$*,* $\mathscr{L}_1$*, and* $\mathscr{L}_2$ *define the same graded ring* $\mathfrak{G}$*, but* Γ_1*,* $\boldsymbol{\theta}_1$*, and* $\boldsymbol{\lambda}_1$ *are only over* $\mathfrak{G}_1$*, while* Γ_2*,* $\boldsymbol{\theta}_2$*, and* $\boldsymbol{\lambda}_2$ *are only over* $\mathfrak{G}_2$*. We define the CS/KV disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *as follows:*

$$
\begin{aligned}
\mathscr{L} &\coloneqq (\mathscr{L}_1 \times \mathcal{X}_2) \cup (\mathcal{X}_1 \times \mathscr{L}_2) & \mathcal{X} &\coloneqq \mathcal{X}_1 \times \mathcal{X}_2 \\
\mathscr{R} &\text{ is defined as in Equation (3.6)} \\
n &\coloneqq n_1 \cdot n_2 & k &\coloneqq k_1 \cdot n_2 + n_1 \cdot k_2 \\
\mathsf{RGen}(\mathsf{lpar}) &\text{ outputs } \rho \coloneqq (\rho_1, \rho_2) \text{ where } \rho_1 \xleftarrow{\$} \mathsf{RGen}_1(\mathsf{lpar}_1) \text{ and } \rho_2 \xleftarrow{\$} \mathsf{RGen}_2(\mathsf{lpar}_2) \\
\Gamma(\rho) &\coloneqq \begin{pmatrix} \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2 \end{pmatrix} \\
\boldsymbol{\theta}(\chi, \rho) &\coloneqq \boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2 \\
\boldsymbol{\lambda}(\chi, w, \rho) &\coloneqq \begin{cases} \begin{pmatrix} \boldsymbol{\lambda}_1 \otimes \boldsymbol{\theta}_2 \\ \mathbf{0}_{n_1 k_2} \end{pmatrix} & \text{if } \mathscr{R}(\chi_1, w_1) = 1 \\ \begin{pmatrix} \mathbf{0}_{k_1 n_2} \\ \boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2 \end{pmatrix} & \text{otherwise, if } \mathscr{R}(\chi_2, w_2) = 1 \end{cases} ,
\end{aligned}
$$

where:

$$
\chi = (\chi_1, \chi_2) \qquad\qquad w = (w_1, w_2) \ .
$$

Before proving the soundness of the construction, let us show it on an example.

Example 3.2.9 (KV disjunction of two DDH languages)**.** *The global parameters* gpar *consist of a bilinear group* $(p, \mathbb{G}_1, \mathbb{G}_2 \mathbb{G}_T, e, g_1, g_2)$*, while the language parameters are* $\mathsf{lpar} \coloneqq (g_1, h_1, g_2, h_2) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$*. We consider the language of the tuples* (u_1, v_1, u_2, v_2) *such that* (u_1, v_1) *is a DH tuple in basis* (g_1, h_1) *or* (u_2, v_2) *is a DH tuple in basis* (g_2, h_2)*:*

$$
\mathscr{L} \coloneqq \{(u_1, v_1, u_2, v_2) \in \mathbb{G}^4 \mid \exists (r_1, r_2) \in \mathbb{Z}_p^2,\ (u_1, v_1) = (g^{r_1}, h^{r_1}) \text{ or } (u_2, v_2) = (g^{r_2}, h^{r_2})\} \ .
$$

In other words, if $(u_1, v_1, u_2, v_2) \in \mathscr{L}$*, either* (u_1, v_1) *is an ElGamal ciphertext of* $1_{\mathbb{G}_1}$ *with encryption key* $\mathsf{ek}_1 \coloneqq (g_1, h_1)$*, or* (u_2, v_2) *is an ElGamal ciphertext of* $1_{\mathbb{G}_2}$ *with encryption key* $\mathsf{ek}_2 \coloneqq (g_2, h_2)$*, or both* (u_1, v_1) *and* (u_2, v_2) *are ElGamal ciphertexts of* $1_{\mathbb{G}_1}$ *and* $1_{\mathbb{G}_2}$ *respectively.*

This language is the disjunction of the DDH languages in basis (g_1, h_1) *and* (g_2, h_2) *respectively. Applying Construction 3.2.8 to Example 3.1.8, we get the following DVS for*

$\mathscr{L}$:

$$n := 4 \qquad k := 4$$

$$\Gamma(\mathfrak{x}) := \begin{pmatrix} g_1 & 0 & g_2 & 0 \\ 0 & g_1 & h_2 & 0 \\ h_1 & 0 & 0 & g_2 \\ 0 & h_1 & 0 & h_2 \end{pmatrix}$$

$$\boldsymbol{\theta}(\mathfrak{x}) := \begin{pmatrix} u_1 \bullet u_2 \\ u_1 \bullet v_2 \\ v_1 \bullet u_2 \\ v_1 \bullet v_2 \end{pmatrix} = \begin{pmatrix} e(u_1, u_2) \\ e(u_1, v_2) \\ e(v_1, u_2) \\ e(v_1, v_2) \end{pmatrix}$$

$$\boldsymbol{\lambda}(\mathfrak{x}, \mathfrak{w}) := \begin{cases} \begin{pmatrix} r_1 \bullet u_2 \\ r_1 \bullet v_2 \\ 0 \\ 0 \\ 0 \\ 0 \\ r_2 \bullet u_1 \\ r_2 \bullet v_1 \end{pmatrix} = \begin{pmatrix} u_2^{r_1} \\ v_2^{r_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ u_1^{r_2} \\ v_1^{r_2} \end{pmatrix} & \textit{if } (u_1, v_1) = (g^{r_1}, h^{r_1}) \\ & \textit{otherwise, if } (u_2, v_2) = (g^{r_2}, h^{r_2}) \end{cases},$$

where:

$$\mathfrak{x} = (u_1, v_1, u_2, v_2) \qquad \mathfrak{w} = (r_1, r_2) \ .$$

Let us now prove the soundness of the construction.

Lemma 3.2.10. *Let $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two CS-DVSs over two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. If $\mathcal{V}_1$ and $\mathcal{V}_2$ are ε_1-sound and ε_2-sound respectively, then the CS/KV disjunction DVS $\mathcal{V}$ of $\mathcal{V}_1$ and $\mathcal{V}_2$ defined in Construction 3.2.8 is a $(\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2)$-sound CS-DVS.*

We have the following immediate corollary.

Corollary 3.2.11. *Let $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two KV-DVSs over two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. Then the CS/KV disjunction DVS $\mathcal{V}$ of $\mathcal{V}_1$ and $\mathcal{V}_2$ defined in Construction 3.2.8 is a KV-DVS.*

Proof of Lemma 3.2.10. We start the proof as the proof of Lemma 3.2.6. Let $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \in \mathcal{X} \setminus \mathscr{L}$. We fix ρ such that $\boldsymbol{\theta}_1 \notin \mathrm{ColSpan}\,(\Gamma_1) = \hat{\mathscr{L}}_1$ and $\boldsymbol{\theta}_2 \notin \mathrm{ColSpan}\,(\Gamma_2) = \hat{\mathscr{L}}_2$. To conclude the proof, we just need to prove that this implies that

$$\boldsymbol{\theta} = \boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2 \quad \notin \quad \mathrm{ColSpan}\,(\Gamma) = \hat{\mathscr{L}} = \hat{\mathscr{L}}_1 \otimes \mathcal{X}_2 + \mathcal{X}_1 \otimes \hat{\mathscr{L}}_2 \,,$$

where the symbol $+$ between vector spaces corresponds to the sum of vector spaces: if E and F are two vector spaces then $E + F = \mathrm{Span}\,(E \cup F)$.

For $i \in \{1, 2\}$, let E_i be an arbitrary supplementary vector space of $\hat{\mathscr{L}}_i$ containing $\boldsymbol{\theta}_i$. We have:

$$\hat{\mathcal{X}}_1 = \hat{\mathscr{L}}_1 \oplus E_1 \qquad\qquad \hat{\mathcal{X}}_2 = \hat{\mathscr{L}}_2 \oplus E_2\,,$$

where $\oplus$ denotes a direct sum of vector spaces. We then get:

$$\begin{aligned}
\hat{\mathscr{L}} &= \hat{\mathscr{L}}_1 \otimes \hat{\mathcal{X}}_2 + \hat{\mathcal{X}}_1 \otimes \hat{\mathscr{L}}_2 \\
&= \left(\hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2 \;+\; \hat{\mathscr{L}}_1 \otimes E_2\right) \;+\; \left(\hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2 \;+\; E_1 \otimes \hat{\mathscr{L}}_2\right) \\
&= \hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2 \quad \oplus \quad \hat{\mathscr{L}}_1 \otimes E_2 \quad \oplus \quad E_1 \otimes \hat{\mathscr{L}}_2\,,
\end{aligned}$$

$$\begin{aligned}
\hat{\mathcal{X}} &= \hat{\mathcal{X}}_1 \otimes \hat{\mathcal{X}}_2 \\
&= \hat{\mathscr{L}}_1 \otimes \hat{\mathscr{L}}_2 \quad \oplus \quad \hat{\mathscr{L}}_1 \otimes E_2 \quad \oplus \quad E_1 \otimes \hat{\mathscr{L}}_2 \quad \oplus \quad E_1 \otimes E_2\;.
\end{aligned}$$

As $\boldsymbol{\theta}_1 \in E_1$ and $\boldsymbol{\theta}_2 \in E_2$, we have $\boldsymbol{\theta} = \boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2 \in E_1 \otimes E_2 \notin \hat{\mathscr{L}}$. This concludes the proof. □

The above proof was proposed by Victor Shoup, as a simplification of the original cumbersome proof in [ABP15c]. We thank him for this proof. In this thesis, we propose a third proof of Lemma 3.2.10, as a straightforward corollary of Proposition 4.3.11. Contrary to the above proof, this third proof uses no advanced tools on tensor products, but just rely on Equation (3.8).

Historical note 3.2.12. *Disjunctions of KV-DVSs were introduced in [ABP15c], while disjunctions of CS-DVSs were never published before.*

3.3 Application to Non-Interactive Zero-Knowledge Arguments

An important application of disjunctions of KV-DVSs is the construction of NIZK with constant-size proofs. An SPHF for some language $\ddot{\mathscr{L}} := \mathscr{L}_1$ (which can be constructed from some KV-DVS $\mathcal{V}_1$) can be seen as a designated-verifier and honest-verifier NIZK: only the user (or verifier) generating the hashing key hk and publishing the projection key hp can verify the validity of a proof of some word (which is the hash value of this word). Furthermore, the verifier may cheat on the projection key to learn some information about the witness. To get rid of these two limitations (designated and honest verifier), we use a second KV-DVS $\mathcal{V}_2$ for a hard-subset-membership language, and we consider the KV disjunction $\mathcal{V}$ of $\mathcal{V}_1$ and $\mathcal{V}_2$. This second KV-DVS $\mathcal{V}_2$ makes the proof publicly verifiable and zero-knowledge.

3.3.1 Overview

Let us construct a NIZK for some language $\ddot{\mathscr{L}} = \mathscr{L}_1$. We suppose that we have a KV-DVS $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ for $\mathscr{L}_1$, together with another KV-DVS $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ for a hard-subset-membership language $\mathscr{L}_2$, such that $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are two multiplicatively compatible sub-graded rings of some graded ring $\mathfrak{G}$. Furthermore, we consider the disjunction KV-DVS $\mathcal{V}$ of $\mathcal{V}_1$ and $\mathcal{V}_2$.

For the sake of simplicity, we suppose that $\mathscr{L}_2 = \hat{\mathscr{L}}_2$, $\mathcal{X}_2 = \hat{\mathcal{X}}_2 = \mathbb{Z}_p^{n_2}$, $\boldsymbol{\theta}_2 = \boldsymbol{\theta}_2(x_2) = x_2$ and $\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_2(x_2, w_2) = w_2$, for any word x_2 and any witness w_2. The common reference string of the NIZK is a projection key $\mathsf{hp} = \boldsymbol{\gamma}^\intercal$ for the disjunction of $\mathscr{L}_1$ and $\mathscr{L}_2$, while the

trapdoor (to simulate proofs) is the hashing key $\mathsf{hk} = \boldsymbol{\alpha}^\intercal$. Essentially, a proof $\boldsymbol{\pi}^\intercal = (\pi_{i_2})_{i_2}^\intercal$ for a statement $\ddot{\chi} = \chi_1$ is just a vector of the hash values of $(\chi_1, \boldsymbol{e}_{2,i_2})$ where $(\boldsymbol{e}_{2,i_2})_{i_2}$ are the scalar vectors of the canonical base of $\hat{\mathcal{X}}_2$. These hash values are $\pi_{i_2} = \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{e}_{2,i_2})$, and can also be computed from the projection key hp and a witness w_1 for χ_1. We can also write:

$$\boldsymbol{\pi}^\intercal = \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \mathsf{Id}_{n_2}) \ .$$

The basic idea is that a valid proof for a word $\chi_1 \in \mathscr{L}_1$ enables us to compute the projected hash value H' of (χ_1, χ_2) for any word $\chi_2 \in \hat{\mathcal{X}}_2$, by linearly combining elements of the proof, since any word $\boldsymbol{\theta}_2 = \chi_2$ can be written as a linear combination of $(\boldsymbol{e}_{2,i_2})_{i_2}$:

$$\begin{aligned}\mathsf{H}' &:= \boldsymbol{\pi}^\intercal \bullet \boldsymbol{\theta}_2 = \sum_{i_2} \pi_{i_2} \bullet \theta_{2,i_2} = \sum_{i_2} \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{e}_{2,i_2}) \bullet \theta_{2,i_2} \\ &= \sum_{i_2} \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1 \otimes (\theta_{2,i_2} \bullet \boldsymbol{e}_{2,i_2})) = \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \,,\end{aligned}$$

if $\boldsymbol{\theta}_2 = \sum_{i_2} \theta_{2,i_2} \bullet \boldsymbol{e}_{1,i_2}$. Hence, for any word $\chi_2 \in \mathscr{L}_2$ for which we know a witness, we can compute the hash value of (χ_1, χ_2), either using a valid proof for χ_1 (as H' above), or directly using a witness $w_2 = \boldsymbol{\lambda}_2$ of $\chi_2 = \boldsymbol{\theta}_2$ and the projection key hp (as for any SPHF for a disjunction), as

$$\mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet \begin{pmatrix} \mathbf{0}_{k_1 n_2} \\ \boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2 \end{pmatrix} \ .$$

To check a proof, we basically check whether for any word $\chi_2 \in \mathscr{L}_2$ with witness $\boldsymbol{\lambda}_2$, these two ways of computing the hash value of (χ_1, χ_2) yields the same result, i.e., we check whether

$$\mathsf{H}' = \boldsymbol{\pi}^\intercal \bullet \boldsymbol{\theta}_2 = \boldsymbol{\gamma}^\intercal \bullet \begin{pmatrix} \mathbf{0}_{k_1 n_2} \\ \boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2 \end{pmatrix} = \mathsf{pH} \ .$$

Thanks to the linearity of the language $\mathscr{L}_2$, it is sufficient to make this test for a family of words χ_2 which generate $\mathscr{L}_2$, such as the columns of Γ_2. We recall that the witness $w_2 = \boldsymbol{\lambda}_2$ for the j_2-th column of Γ_2 is just the j_2-th vector in the canonical basis of $\mathbb{Z}_p^{k_2}$. Therefore, to check $\boldsymbol{\pi}^\intercal$, we just need to check whether

$$\boldsymbol{\pi}^\intercal \bullet \Gamma_2 \stackrel{?}{=} \boldsymbol{\gamma}^\intercal \bullet \begin{pmatrix} \mathbf{0}_{k_1 n_2 \times k_2} \\ \boldsymbol{\theta}_1 \otimes \mathsf{Id}_{k_2} \end{pmatrix} \ .$$

The trapdoor, i.e., the hashing key hk, clearly enables us to simulate any proof, and the resulting proofs are perfectly indistinguishable from normal ones, hence the perfect zero-knowledge property. Moreover, the soundness comes from the fact that a proof for a word $\chi_1 \notin \mathscr{L}_1$ can be used to solve the subset-membership problem for $\mathscr{L}_2$.

More precisely, let us consider a soundness adversary which takes as input the projection key hp and which outputs a word $\chi_1 \notin \mathscr{L}_1$ and a valid proof $\boldsymbol{\pi}^\intercal$ for χ_1. On the one hand, such a valid proof enables us to compute the hash value H' of (χ_1, χ_2) for any word $\chi_2 \in \mathscr{L}_2$, by linearly combining elements of the proofs (as seen above), and the validity of the proof ensures the resulting value H' is correct if $\chi_2 \in \mathscr{L}_2$. On the other hand, we can also compute a hash value H of (χ_1, χ_2) for any $\chi_2 \in \mathcal{X}_2$ using the hashing key hk. Then, if $\chi_2 \in \mathscr{L}_2$, necessarily $\mathsf{H} = \mathsf{H}'$, while if $\chi_2 \notin \mathscr{L}_2$, the smoothness ensures that H looks completely random when given only hp. Since H' does not depend on hk but only on hp, it is different from H with overwhelming probability. Therefore, we can use such an adversary to solve the subset-membership problem for $\mathscr{L}_2$.

3.3.2 Construction

Construction 3.3.1 (NIZK from disjunctions of KV-DVSs)**.** *We suppose that we have a KV-DVS* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *for* $\ddot{\mathscr{L}} = \mathscr{L}_1$*, together with another KV-DVS* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *for a hard-subset-membership language* $\mathscr{L}_2$*, such that* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *are two multiplicatively compatible sub-graded rings of some graded ring* $\mathfrak{G}$*. Furthermore, we consider the KV disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*. We recall that* $n = n_1 n_2$ *and* $k = k_1 n_2 + n_1 k_2$*.*

We construct a NIZK for $\ddot{\mathscr{L}} := \mathscr{L}_1$ *as follows:*

- $\mathsf{NIZK.Setup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar_2(gpar)}$ *for* $\mathscr{L}_2$*, together with a hashing key* hk *and the associated projection key* hp *for* $\mathcal{V}$ *as follows:*

$$\begin{aligned} \mathsf{hk} &:= \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}\,, \\ \mathsf{hp} &:= \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma \quad \in \mathfrak{G}^{1\times k}\,, \end{aligned}$$

  and outputs $\mathsf{crs} := (\mathsf{lpar}_2, \mathsf{hp})$*. In the sequel, we split* $\boldsymbol{\gamma}^\intercal$ *in two parts:*

$$\begin{aligned} \boldsymbol{\gamma}_1^\intercal &:= (\gamma_i)_{i=1,\dots,k_1 n_2}^\intercal = \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times(k_1 n_2)}\,, \\ \boldsymbol{\gamma}_2^\intercal &:= (\gamma_i)_{i=k_1 n_2+1,\dots,n}^\intercal = \boldsymbol{\alpha}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \quad \in \mathfrak{G}_2^{1\times(n_1 k_2)}\,; \end{aligned}$$

- $\mathsf{NIZK.Sim_1}(\mathsf{lpar}_1)$ *works as* $\mathsf{NIZK.Setup}$ *except it also outputs the following trapdoor:*

$$\mathsf{trap} := \mathsf{hk} = \boldsymbol{\alpha}^\intercal \quad \in \mathbb{Z}_p^{1\times n}\,;$$

- $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, x_1, w_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1(x_1, w_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times n_2}\,;$$

- $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, x_1, \boldsymbol{\pi}^\intercal)$ *checks the following equation:*

$$\boldsymbol{\pi}^\intercal \bullet \Gamma_2 \stackrel{?}{=} \boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{k_2})\,; \tag{3.9}$$

- $\mathsf{NIZK.Sim_2}(\mathsf{trap}, \mathsf{tag}, x_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times n_2}\,.$$

Tags tag *are not used in this construction.*

We remark that the size of the proof $\boldsymbol{\pi}^\intercal$ is independent of the language $\mathscr{L}_1$ and only depend on the second DVS $\mathcal{V}_2$ which can be fixed. That is why, we say that our NIZK has constant-size proofs.

3.3.3 Completeness and Security

Theorem 3.3.2. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DVSs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. We suppose that* $\mathscr{L}_2$ *is a hard-subset-membership language.*

Then the NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 3.3.1 is perfectly complete, perfectly zero-knowledge, and sound. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against soundness of the NIZK, we can construct an adversary* $\mathcal{B}$ *against hard subset membership of* $\mathscr{L}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{sound}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{sub\text{-}memb}(\mathcal{B}, \mathfrak{K}) + \frac{1}{p} \ .$$

Proof. Completeness. If the proof $\boldsymbol{\pi}^\intercal$ has been generated correctly, the left hand side of the verification equation (Equation (3.9)) is equal to

$$\begin{aligned}
\boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1 \otimes \mathsf{Id}_{n_2}) \bullet \Gamma_2 &= (\boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2})) \bullet (\boldsymbol{\lambda}_1 \otimes \mathsf{Id}_{n_2}) \bullet (\mathsf{Id}_1 \otimes \Gamma_2) \\
&= \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \bullet ((\boldsymbol{\lambda}_1 \bullet \mathsf{Id}_1) \otimes (\mathsf{Id}_{n_2} \bullet \Gamma_2)) \\
&= \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \bullet (\boldsymbol{\lambda}_1 \otimes \Gamma_2) \\
&= \boldsymbol{\alpha}^\intercal \bullet ((\Gamma_1 \bullet \boldsymbol{\lambda}_1) \otimes (\mathsf{Id}_{n_2} \bullet \Gamma_2)) \ ,
\end{aligned}$$

while the right hand side is equal to:

$$\boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \mathsf{Id}_{k_2}) = \boldsymbol{\alpha}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \bullet (\boldsymbol{\theta}_1 \otimes \mathsf{Id}_{k_2}) = \boldsymbol{\alpha}^\intercal \bullet ((\mathsf{Id}_{n_1} \bullet \boldsymbol{\theta}_1) \otimes (\Gamma_2 \bullet \mathsf{Id}_{k_2})) \, ,$$

which is the same as the left hand side, since $\Gamma_1 \bullet \boldsymbol{\lambda}_1 = \mathsf{Id}_{n_1} \bullet \boldsymbol{\theta}_1$ and $\mathsf{Id}_{n_2} \bullet \Gamma_2 = \Gamma_2 \bullet \mathsf{Id}_{k_2}$. Hence the (perfect) *completeness*. Another way to see it, is that the i_2-th column of the right hand side of Equation (3.9) is the hash value of "$(\boldsymbol{\theta}_1, \Gamma_2 \bullet \boldsymbol{e}_{2,i_2})$" computed using the witness $\boldsymbol{\lambda}_2 = \boldsymbol{e}_{2,i_2}$, while the i_2-th column of the left hand side is this hash value computed using the witness $\boldsymbol{\lambda}_1$.

Zero-Knowledge. The (perfect) *zero-knowledge* property comes from the fact that the normal proof $\boldsymbol{\pi}^\intercal$ for $x_1 \in \mathscr{L}_1$ with witness w_1 is:

$$\boldsymbol{\pi}^\intercal = \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1 \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \bullet (\boldsymbol{\lambda}_1 \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\alpha}^\intercal \bullet ((\Gamma_1 \bullet \boldsymbol{\lambda}_1) \otimes (\mathsf{Id}_{n_2} \bullet \mathsf{Id}_{n_2})) \, , \quad (3.10)$$

which is equal to the simulated proof for x_1, as $\boldsymbol{\theta}_1 = \Gamma_1 \bullet \boldsymbol{\lambda}_1$ and $\mathsf{Id}_{n_2} \bullet \mathsf{Id}_{n_2} = \mathsf{Id}_{n_2}$.

Soundness. It remains to prove the soundness property, under the hard subset membership of $\mathscr{L}_2$. We just need to show that if the adversary $\mathcal{A}$ is able to generate a valid proof $\boldsymbol{\pi}^\intercal$ for a word $x_1 \notin \mathscr{L}_1$, then we can use $\boldsymbol{\pi}^\intercal$ to check if a word x_2 is in $\mathscr{L}_2$ or not. More precisely, we define the adversary $\mathcal{B}$ as follows: on input a word $x_2 \in \mathcal{X}_2$, it runs $\mathcal{A}$ to get a proof $\boldsymbol{\pi}^\intercal$ for a word $x_1 \in \mathcal{X}_1$, then it computes H be the hash value of (x_1, x_2) using hk, and the value $\mathsf{H}' := \boldsymbol{\pi}^\intercal \bullet \boldsymbol{\theta}_2$. The adversary $\mathcal{B}$ then returns 0 if the proof $\boldsymbol{\pi}^\intercal$ is correct (i.e., satisfies Equation (3.9)) and $\mathsf{H} = \mathsf{H}'$ (indicating it guesses that $x_2 \in \mathscr{L}_2$), and outputs 1 otherwise.

On the one hand, if $x_2 \in \mathscr{L}_2$ (and whether $x_1 \in \mathscr{L}_1$ or not), there exists a witness w_2 such that $\boldsymbol{\theta}_2(x_2) = \Gamma_2 \bullet \boldsymbol{\lambda}_2(x_2, w_2)$ and so, thanks to Equation (3.9), we have:

$$\mathsf{H}' = \boldsymbol{\pi}^\intercal \bullet \Gamma_2 \bullet \boldsymbol{\lambda}_2 = \boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \mathsf{Id}_{k_2}) \bullet \boldsymbol{\lambda}_2 = \boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2) = \mathsf{H} \, ,$$

the last equality coming from the correctness of the SPHF and the fact the second to last expression is just the hash value of $(\mathfrak{x}_1, \mathfrak{x}_2)$ computed using ProjHash and witness w_2. Therefore, we have:

$$\Pr\left[\mathsf{Exp}^{\texttt{sub-memb-0}}(\mathcal{B}, \mathfrak{K}) = 1\right] = \Pr\left[\boldsymbol{\pi}^\intercal \text{ generated by } \mathcal{A} \text{ is not correct}\right],$$

where $\mathsf{Exp}^{\texttt{sub-memb-0}}$ is defined in Figure 2.3.

On the other hand, if $\mathfrak{x}_2 \notin \mathscr{L}_2$ and $\mathfrak{x}_1 \notin \mathscr{L}_1$, then $(\mathfrak{x}_1, \mathfrak{x}_2) \notin \mathscr{L}$. So H looks uniformly random by smoothness and the probability that $\mathsf{H}' = \mathsf{H}$ is at most $1/|\Pi| = 1/p$.

$$\Pr\left[\mathsf{Exp}^{\texttt{sub-memb-1}}(\mathcal{B}, \mathfrak{K}) = 1\right] \geq \Pr\left[\boldsymbol{\pi}^\intercal \text{ generated by } \mathcal{A} \text{ is not correct}\right] + \Pr\left[\boldsymbol{\pi}^\intercal \text{ is correct and } \mathfrak{x}_1 \notin \mathscr{L}_1\right] - 1/p \ .$$

As the probability that $\boldsymbol{\pi}^\intercal$ is correct and $\mathfrak{x}_1 \notin \mathscr{L}_1$ is exactly $\mathsf{Adv}^{\mathsf{sound}}(\mathcal{A}, \mathfrak{K})$, we get the expected result.

We remark that $\mathcal{B}$ does not need to check whether $\mathfrak{x}_1 \in \mathscr{L}_1$. This is important as it might not be possible to do this check in polynomial time. □

Remark 3.3.3. *In [KW15, Section 3.1], Kiltz and Wee proposed the same scheme, but stated differently, without tensor products. They also manage to reduce the soundness to the following computational assumption: it is hard to find a non-zero vector in $\boldsymbol{\beta}^\intercal = \mathfrak{G}_1^{1\times n_2}$ such that $\boldsymbol{\beta}^\intercal \bullet \Gamma_2 = \mathbf{0}_{1\times k_2}$. This assumption is weaker that the hard-subset-membership assumption we are using as finding such a vector enables to solve the subset-membership problem: to check whether $\mathfrak{x}_2 \in \mathscr{L}_2$, we check whether $\boldsymbol{\beta}^\intercal \bullet \boldsymbol{\theta}_2 = 0$. As our scheme is identical to the one in [KW15, Section 3.1], we could also reduce soundness to this computational assumption.*

For the sake of completeness, here is a sketch of the reduction. We define an adversary $\mathcal{B}$ against this computational assumption as follows: it runs $\mathcal{A}$ and get a proof $\boldsymbol{\pi}^\intercal$ for a word $\mathfrak{x}_1 \in \mathcal{X}_1$. Then it outputs

$$\boldsymbol{\beta}^\intercal := \boldsymbol{\pi}^\intercal - \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1(\mathfrak{x}_1) \otimes \mathsf{Id}_{n_2}) \ .$$

Equations (3.9) and (3.10) directly imply that $\boldsymbol{\beta}^\intercal \bullet \Gamma_2 = \mathbf{0}_{1\times k_2}$. Therefore, we just need to prove that $\boldsymbol{\beta}^\intercal \neq \mathbf{0}_{1\times n_2}$ with probability at least $1 - 1/p$. For any word $\mathfrak{x}_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$, we have:

$$\boldsymbol{\beta}^\intercal \bullet \boldsymbol{\theta}_2(\mathfrak{x}_2) = \boldsymbol{\pi}^\intercal \bullet \boldsymbol{\theta}_2 - \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1(\mathfrak{x}_1) \otimes \mathsf{Id}_{n_2}) \bullet \boldsymbol{\theta}_2 = \mathsf{H}' - \mathsf{H},$$

with H' and H defined as in the proof above, because $(\boldsymbol{\theta}_1(\mathfrak{x}_1) \otimes \mathsf{Id}_{n_2}) \bullet \boldsymbol{\theta}_2 = \boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2$. But as $\mathfrak{x}_2 \notin \mathscr{L}_2$, smoothness ensures that H is uniformly random and $\mathsf{H}' - \mathsf{H}$ is 0 with probability $1/p$. Thus $\boldsymbol{\beta}^\intercal \neq \mathbf{0}_{1\times n_2}$ with probability at least $1 - 1/p$.

3.4 More Examples

3.4.1 Matrix Decisional Diffie-Hellman Assumptions (MDDH)

In an independent work [EHK+13], Escala et al. introduced an algebraic framework for Diffie-Hellman-like assumptions, over cyclic groups. They generalized the DDH assumption and many of its variants (such as the DLin assumption, introduced by Boneh, Boyen, and Shacham in [BBS04]), into the *matrix decisional Diffie-Hellman* (MDDH) family of assumptions.

Let us provide a slight generalization of the MDDH assumption to any graded ring of prime order.

Definition 3.4.1. *Let $\mathfrak{G}$ be a graded ring of prime order p. Let n and k be two positive integers. Let $\tilde{\boldsymbol{v}}$ be some index for this graded ring. Let $\mathcal{D}$ be a distribution of matrices in $\mathbb{Z}_p^{n\times k}$ (samplable in polynomial time), the $\mathcal{D}$-MDDH* assumption *(over $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$) states that the two following distributions are computationally indistinguishable:*

$$\left\{([A],[A\cdot \boldsymbol{r}]) \mid A \xleftarrow{\$} \mathcal{D}; \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k\right\}$$
$$\left\{([A],[\boldsymbol{u}]) \mid A \xleftarrow{\$} \mathcal{D}; \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^n\right\},$$

where $[A]$ is the matrix $([\tilde{\boldsymbol{v}}, A_{i,j}])_{\substack{i=1,\dots,n\\ j=1,\dots,k}} \in \mathfrak{G}_{\tilde{\boldsymbol{v}}}$.

Concretely, for $\mathfrak{G}$ a cyclic group $(p,\mathbb{G},g)$, $[A]$ is the matrix $(g^{A_{i,j}})_{\substack{i=1,\dots,n\\ j=1,\dots,k}}$.

We can define an associated KV-DVS $\mathcal{V} = (p,\mathfrak{G},\mathcal{X},\mathscr{L},\mathscr{R},n,k,\mathsf{RGen},\Gamma,\boldsymbol{\theta},\boldsymbol{\lambda})$ as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ generates the graded ring $\mathsf{gpar} := \mathfrak{G}$ for the security parameter $\mathfrak{K}$;
- $\mathsf{Setup.lpar}(\mathsf{gpar})$ generates and random matrix $A \xleftarrow{\$} \mathcal{D}$ and outputs $\mathsf{lpar} := [A] \in \mathfrak{G}_{\tilde{\boldsymbol{v}}}^{n\times k}$ and $\mathsf{ltrap} = A$;
- the matrix Γ_{lpar} is exactly the matrix $[A] = \mathsf{lpar}$;
- the functions θ and $\boldsymbol{\lambda}$ are the identity functions;
- the language is defined by:

$$\mathscr{L} := \hat{\mathscr{L}} := \{[\boldsymbol{u}] \mid \exists \boldsymbol{r} \in \mathbb{Z}_p^k,\ \boldsymbol{u} = A\cdot \boldsymbol{r}\} \quad \subseteq \quad \mathfrak{G}_{\tilde{\boldsymbol{v}}}^n =: \hat{\mathcal{X}} =: \mathcal{X}\ .$$

The $\mathcal{D}$-MDDH assumption corresponds the hard-subset-membership property of the language $\mathscr{L}$, up to an additive term $1/p$, as the probability that a uniform vector $[\boldsymbol{u}] \xleftarrow{\$} \mathfrak{G}^n$ is in $\mathscr{L}$ is at most $1/p$. Furthermore, we can define an SPHF associated to this DVS, which exactly corresponds to the SPHF in [EHK+13].

We should point out that, although the work of Escala et al. is similar to our work regarding this construction of SPHF, we looked at the problem from two distinct perspectives and for different goals. Escala et al. provide a deep and insightful analysis of MDDH assumptions, their relations (for different distributions $\mathcal{D}$), and ways to prove them easily in generic (multilinear) group models. On the other hand, we are more interested in SPHFs for more complex languages and their applications to the construction of NIZK for example. Many of our constructions use a DVS associated to a hard-subset-membership language, which can be instantiated with any secure MDDH assumption. This difference of point of view also explains the difference of notation in the two works: in [EHK+13], all the discrete logarithms of the elements appearing in the assumptions are known (by the algorithm generating an instance of the MDDH assumption) and the use of the bracket notation ($[A]$) is extremely practical in that case, while in our work, the elements we consider might come from somewhere else (such as an ElGamal public key) and their discrete logarithms might not be known.

3.4.2 Cramer-Shoup Encryption

Similarly to the language of ElGamal ciphertexts in Section 3.1.1.3, we can also consider the language of Cramer-Shoup ciphertexts (see Section 2.2.2.3). Global parameters $\mathsf{gpar} := (p, \mathbb{G}, g, \mathcal{H})$, while language parameters $\mathsf{lpar} := \mathsf{ek} := (g_1, g_2, c, d, h) \in \mathbb{G}^5$ consist of a Cramer-Shoup encryption key. The language is and defined by:

$$\mathscr{L} = \{(\ell, \mathsf{m}, \mathsf{c} = (u_1, u_2, v, w)) \in \{0,1\}^* \times \mathscr{M} \times \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p, \\ (u_1, u_2, v, w) = (g_1^r,\, g_2^r,\, h^r \cdot \mathcal{G}(\mathsf{m}),\, (cd^\xi)^r) \text{ with } \xi = \mathcal{H}(\ell, u_1, u_2, v)\}\,,$$

where $\mathcal{G}$ is a reversible map from the message set $\mathscr{M}$ to $\mathbb{G}$. We write $\mathsf{M} = \mathcal{G}(\mathsf{m})$.

GL-DVS. Constructing a GL-DVS for this language is similar to constructing a DVS for the language of ElGamal ciphertexts. We can construct such a DVS $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ as follows:

$$n := 4 \qquad\qquad k := 1$$

$$\Gamma(\chi) := \begin{pmatrix} g_1 \\ g_2 \\ h \\ cd^\xi \end{pmatrix} \qquad \boldsymbol{\theta}(\chi) := \begin{pmatrix} u_1 \\ u_2 \\ v/\mathsf{M} \\ w \end{pmatrix} \qquad \boldsymbol{\lambda}(\chi, w) := r\,,$$

where $\chi = (\ell, \mathsf{m}, (u_1, u_2, v, w))$, $w = r$, and $\xi = \mathcal{H}(\ell, u_1, u_2, v)$.

This DVS is a GL-DVS as Γ depends on ξ which itself depends on χ.

Historical note 3.4.2. *The resulting GL-SPHF was known since [KOY09; GL06]. However, the KV-SPHF resulting from the following KV-DVS is much more recent and was proposed in [BBC+13c]. Its discovery helped constructing the most efficient one-round PAKE at the time.*

KV-DVS. To get a KV-DVS, we need to find a way to make Γ independent of ξ. The idea is to add a row and a column to Γ. We can construct a KV-DVS $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ for the language of Cramer-Shoup ciphertexts as follows:

$$n := 5 \qquad\qquad k := 2$$

$$\Gamma := \begin{pmatrix} g_1 & 0 \\ 0 & g_1 \\ g_2 & 0 \\ h & 0 \\ c & d \end{pmatrix} \qquad \boldsymbol{\theta}(\chi) := \begin{pmatrix} u_1 \\ u_1^\xi \\ u_2 \\ v/\mathsf{M} \\ w \end{pmatrix} \qquad \boldsymbol{\lambda}(\chi, w) := \begin{pmatrix} r \\ r\xi \end{pmatrix}\,,$$

where $\chi = (\ell, \mathsf{m}, (u_1, u_2, v, w))$, $w = r$, and $\xi = \mathcal{H}(\ell, u_1, u_2, v)$. Soundness comes from the fact that if $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$ and $u_1 = g_1^r$, then necessarily $\boldsymbol{\lambda} = (r, r\xi)^\intercal$.

Validity of Cramer-Shoup ciphertexts. If we remove from the matrix Γ and the vector $\boldsymbol{\theta}$ the row corresponding to the elements h and v/M (respectively), the previous two DVSs just check the validity of a Cramer-Shoup ciphertext, without looking at the exact plaintext.

All our examples of DVSs for languages using ElGamal ciphertexts (such as Examples 3.1.3 and 3.1.4 and Section 3.4.3) can therefore just be extended to DVSs for the same languages using Cramer-Shoup ciphertexts by doing the conjunction of these DVSs with the DVSs used to check validity of Cramer-Shoup ciphertexts. In most cases, matrices can be slightly optimized or compressed. We leave these optimizations to the reader.

3.4.3 Encryption of Plaintexts Satisfying a System of Quadratic Equations

To demonstrate the power of DVSs, let us show how to construct a KV-DVS for the encryption of plaintexts satisfying a system of bilinear equations.

Example 3.4.3. *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$*, an asymmetric bilinear group of prime order* p*. The language parameters are* $\mathsf{lpar} := (g_1, h_1, g_2, h_2, g_T, h_T) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_T^2$ *where* $\mathsf{ek}_1 := (g_1, h_1)$*,* $\mathsf{ek}_2 := (g_2, h_2)$*, and* $\mathsf{ek}_T := (g_T, h_T)$ *are three random ElGamal encryption keys in* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *respectively. Let* ν_1*,* ν_2*, and* ν_T *be three non-negative integers. We consider the language of tuples*

- *of ElGamal ciphertexts* $(\mathsf{c}_{\omega,i} := (u_{\omega,i}, v_{\omega,i}))_{i=1,\dots,\nu_\omega}$ *of plaintexts* $(\mathsf{M}_{\omega,i})_{i=1,\dots,\nu_\omega}$ *under the encryption key* ek_ω*, for* $\omega \in \{1, 2, T\}$*,*
- *and of group elements* $(A_{\omega,i})_{i=1,\dots,\nu_\omega} \in \mathbb{G}_\omega^{\nu_\omega}$ *for* $\omega \in \{1, 2\}$ *and* $B \in \mathbb{G}_T$*,*
- *and of scalars* $(a_{i,j})_{\substack{i=1,\dots,\nu_1 \\ j=1,\dots,\nu_2}} \in \mathbb{Z}_p^{\nu_1 \times \nu_2}$ *and* $(a_{T,i})_{i=1,\dots,\nu_T} \in \mathbb{Z}_p^{\nu_T}$*,*

such that the following equation is satisfied:

$$\prod_{i=1}^{\nu_1} e(\mathsf{M}_{1,i}, A_{2,i}) \cdot \prod_{j=1}^{\nu_2} e(A_{1,j}, \mathsf{M}_{2,j}) \cdot \prod_{i=1}^{\nu_1} \prod_{j=1}^{\nu_2} e(\mathsf{M}_{1,i}, \mathsf{M}_{2,j})^{a_{i,j}} \cdot \prod_{k=1}^{\nu_T} \mathsf{M}_{T,k}^{a_{T,k}} = B \ .$$

This equation can also be written:

$$\left(\sum_{i=1}^{\nu_1} A_{2,i} \bullet \mathsf{M}_{1,i}\right) + \left(\sum_{j=1}^{\nu_2} A_{1,j} \bullet \mathsf{M}_{2,j}\right) + \left(\sum_{i=1}^{\nu_1} \sum_{j=1}^{n_2} a_{i,j} \bullet \mathsf{M}_{1,i} \bullet \mathsf{M}_{2,j}\right) + \left(\sum_{k=1}^{n_T} a_{T,k} \bullet \mathsf{M}_{T,k}\right) = B \ .$$

The witness w *for such a word is the tuple of scalars*

$$w := ((r_{1,i})_{i=1,\dots,\nu_1}, (r_{2,i})_{i=1,\dots,\nu_2}, (r_{T,i})_{i=1,\dots,\nu_T}) \,,$$

such that $u_{\omega,i} = g_\omega^{r_{\omega,i}}$ *and* $v_{\omega,i} = h_\omega^{r_{\omega,i}} \cdot \mathsf{M}_{\omega,i}$*, for* $\omega \in \{1, 2, T\}$ *and* $i \in \{1, \dots, \nu_\omega\}$*.*

The KV-DVS is defined as follows:

$$n := 5$$
$$k := 4$$
$$\Gamma := \begin{pmatrix} g_1 \bullet g_2 & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_T \\ h_1 \bullet h_2 & h_1 & h_2 & h_T \end{pmatrix} = \begin{pmatrix} e(g_1, g_2) & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_T \\ e(h_1, h_2) & h_1 & h_2 & h_T \end{pmatrix}$$

$$
\boldsymbol{\theta} := \begin{pmatrix}
- \sum_i \sum_j a_{i,j} \bullet u_{1,i} \bullet u_{2,j} \\
\left(\sum_i \sum_j a_{i,j} \bullet u_{1,i} \bullet e_{2,j}\right) + \left(\sum_i A_{2,i} \bullet u_{1,i}\right) \\
\left(\sum_i \sum_j a_{i,j} \bullet e_{1,i} \bullet u_{2,j}\right) + \left(\sum_j A_{1,j} \bullet u_{2,j}\right) \\
\sum_i a_{T,i} \bullet u_{T,i} \\
\left(\sum_i \sum_j a_{i,j} \bullet e_{1,i} \bullet e_{2,j}\right) + \left(\sum_i A_{2,i} \bullet e_{1,i}\right) + \left(\sum_j A_{1,j} \bullet e_{2,j}\right) + \left(\sum_k a_{T,k} \bullet e_{T,k}\right) - B
\end{pmatrix}
$$

$$
= \begin{pmatrix}
\prod_i \prod_j e(u_{1,i}, u_{2,j})^{a_{i,j}} \\
\left(\prod_i \prod_j e(u_{1,i}, e_{2,j})^{a_{i,j}}\right) \cdot \left(\prod_i e(u_{1,i}, A_{2,i})\right) \\
\left(\prod_i \prod_j e(e_{1,i}, u_{2,j})^{a_{i,j}}\right) \cdot \left(\prod_j e(A_{1,j}, u_{2,j})\right) \\
\prod_i u_{T,i}^{a_{T,i}} \\
\left(\prod_i \prod_j e(e_{1,i}, e_{2,j})^{a_{i,j}}\right) \cdot \left(\prod_i e(e_{1,i}, A_{2,i})\right) \cdot \left(\prod_j e(A_{1,j}, e_{2,j})\right) \cdot \left(\prod_k e(a_{T,k}, e_{T,k})\right) \cdot B^{-1}
\end{pmatrix}
$$

$$
\boldsymbol{\lambda} := \begin{pmatrix}
- \sum_i \sum_j a_{i,j} \cdot r_{1,i} \cdot r_{2,j} \\
\left(\sum_i \sum_j r_{1,i} \bullet a_{i,j} \bullet e_{2,j}\right) + \left(\sum_i A_{2,i} \bullet r_{1,i}\right) \\
\left(\sum_i \sum_j r_{2,i} \bullet a_{i,j} \bullet e_{1,j}\right) + \left(\sum_j A_{1,j} \bullet r_{2,j}\right) \\
\sum_k r_{T,k}
\end{pmatrix}
$$

$$
= \begin{pmatrix}
- \sum_i \sum_j a_{i,j} \cdot r_{1,i} \cdot r_{2,j} \\
\left(\prod_i \prod_j e_{2,j}^{r_{1,i} \cdot a_{i,j}}\right) \cdot \left(\prod_i A_{2,i}^{r_{1,i}}\right) \\
\left(\prod_i \prod_j e_{1,j}^{r_{2,i} \cdot a_{i,j}}\right) \cdot \left(\prod_j A_{1,j}^{r_{2,j}}\right) \\
\sum_k r_{T,k}
\end{pmatrix} .
$$

Using this example and conjunctions, we can easily construct DVSs for languages of encryptions of plaintexts satisfying a system of quadratic equations.

Other examples of DVSs can also be found in [BBC+13c].

Chapter 4

Diverse Modules

This chapter constitutes the core of this thesis. It formally describes *diverse modules* (DMs), which are extensions of DVSs to modules over rings $\mathbb{Z}_M$ instead of just vector spaces (or modules) over finite fields $\mathbb{Z}_p$. Contrary to DVSs, DMs also handle languages based on the *quadratic residuosity* (QR) assumption and the *decisional composite residuosity* (DCR) assumption for example.

The direct construction of a PHF from a DM is not smooth but satisfies a slightly weaker notion called universality we recall in the first section. Then, we formally define DMs. Finally, similarly to what we did for DVSs, we show some ways to combine or enhance DMs: conjunctions, disjunctions, and t-sound extensions. A t-sound DM yield a t-universal projective hash function. The notion of t-universality was introduced by Cramer and Shoup in [CS02] for $t = 2$ and has many applications, such as $(t-1)$-time simulation-sound NIZK, when combined with disjunctions.

Contents

4.1 Universality and Smoothness **86**
- 4.1.1 Motivation 86
- 4.1.2 Universal Projective Hash Functions 86
- 4.1.3 Weakly Universal Projective Hash Functions 87

4.2 Diverse Modules (DMs) **88**
- 4.2.1 Graded Rings 89
- 4.2.2 Diverse Modules, Universal PHFs, and Tools for Composite Order . . 92
- 4.2.3 Link with Diverse Groups 102

4.3 Conjunctions and Disjunctions **103**
- 4.3.1 Conjunctions 103
- 4.3.2 Disjunctions 103

4.4 t-Universality, t-Smoothness, and t-Soundness **111**
- 4.4.1 t-Universality and t-Smoothness 112
- 4.4.2 t-Soundness 114
- 4.4.3 Construction of t-Sound Tag-CS-DMs and Tag-CS-DVSs 115

4.1 Universality and Smoothness

We first show why smoothness is too strong when working over cyclic groups of composite order. Then, we introduce the notion of universality and show how to transform a universal PHF into a smooth PHF.

4.1.1 Motivation

Let us consider again the DDH language $\mathscr{L}$ over a cyclic group $(M, \mathbb{G}, g)$ defined in Example 2.3.1:

$$\mathscr{L} = \{(u, v) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_M,\ (u, v) = (g^r, h^r)\} \quad \subsetneq \quad \mathbb{G}^2 = \mathcal{X} \ .$$

We now use the letter M to denote the order of the group $\mathbb{G}$. We recall the classical SPHF for this language:

$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_M^{1\times 2} \qquad \mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \cdot \begin{pmatrix} g \\ h \end{pmatrix} \in \mathbb{G}$$

$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} u \\ v \end{pmatrix} = u^{\alpha_1} \cdot v^{\alpha_2} \qquad \mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet r \ .$$

This PHF is perfectly smooth when $M = p$ is a prime number.

We may want to extend the previous construction to the case where M is a composite number $M = pq$, with p and q being two distinct prime numbers. Unfortunately, in this case, the SPHF is not smooth anymore: if $\chi = (g^p, h^{2p}) \notin \mathscr{L}$, the hash value of χ is

$$\mathsf{H} = \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} g^p \\ h^{2p} \end{pmatrix} = p \bullet \boldsymbol{\alpha}^\intercal \bullet \begin{pmatrix} g \\ h^2 \end{pmatrix}$$

and lies in $\mathbb{G}_q$ the subgroup of $\mathbb{G}$ of order q. It is therefore not at all indistinguishable from a uniform element in $\mathbb{G}$.

However, we still remark that the value H is uniformly random in $\mathbb{G}_q$, given hp. Thus it has high min entropy. Universality captures this notion of high min entropy.

We could use a randomness extractor (see Section 2.2.3) to extract from H a uniformly random string (when $\chi \notin \mathscr{L}$). The seed seed can just be put in the projection key hp. This works perfectly with CS/GL smoothness, as in this case the seed is really independent from the word χ. But with KV smoothness, this is not necessarily the case and such extractor does not work. Luckily, in many cases, universality is a sufficient property and no extraction is required.

Historical note 4.1.1. *In [CS02], smoothness was introduced after universality. In this thesis, we introduce the two notions in the reverse order.*

4.1.2 Universal Projective Hash Functions

4.1.2.1 Universality

Definition 4.1.2. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε-universal *if for any* lpar*, any word* $\chi \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$*, any projection key* hp*, and any hash*

value $\mathsf{H} \in \Pi$*, we have*

$$\Pr\left[\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x) = \mathsf{H} \text{ and } \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, x) = \mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right] \\ \leq \varepsilon \cdot \Pr\left[\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, x) = \mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right] .$$

It is universal *if it is* ε*-universal with* ε *is negligible in* $\mathfrak{K}$*.*

Another way to look at universality is to say that the PHF is $1/2^k$-universal if the hash value $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x)$ of a word $x \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ has min entropy at least k, conditioned on $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) = \mathsf{hp}$. Intuitively, a PHF is universal if it is hard to guess the hash value of a word $x \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$, given only the projection key hp.

4.1.2.2 Approximate Universality

As noted by Cramer and Shoup in [CS02], it is often practical to consider an approximate version of universality.

Definition 4.1.3. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *and a PHF* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *(with algorithms not necessarily running in polynomial time) for the same language* $\mathscr{L} \subseteq \mathcal{X}$ *are said to be* ε-close*, if for any* lpar*, the following distributions are* ε*-close:*

$$\left\{(\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, x), \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x))_{x \in \mathcal{X}_{\mathsf{lpar}}} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right\}$$
$$\left\{(\mathsf{ProjKG}'(\mathsf{hk}, \mathsf{lpar}, x), \mathsf{Hash}'(\mathsf{hk}, \mathsf{lpar}, x))_{x \in \mathcal{X}_{\mathsf{lpar}}} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}'(\mathsf{lpar})\right\},$$

where $(\cdot, \cdot)_{x \in \mathcal{X}_{\mathsf{lpar}}}$ *is a tuple of* $|\mathcal{X}_{\mathsf{lpar}}|$ *pairs* $(\cdot, \cdot)$*, one for each* $x \in \mathcal{X}_{\mathsf{lpar}}$*.*

In this section, we do not require algorithms $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ to run in polynomial time.

Definition 4.1.4. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε'-close to be ε-universal*, if there exists an* ε*-universal PHF* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *(with algorithms not necessarily running in polynomial time) that is* ε'*-close to the PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*. It is* close to be universal *or* approximately universal *if it is* ε'*-close to be* ε*-universal with* ε' *and* ε *negligible in* $\mathfrak{K}$*.*

It should be noted that an PHF that is close to be universal is not necessarily universal, as it is possible that for one of its projection keys, there is only one possible hash value for some word $x \in \mathcal{X} \setminus \mathscr{L}$. We remark that if a KV-SPHF (resp. CS-SPHF, GL-SPHF) is ε'-close to an ε-smooth KV-SPHF (resp. CS-SPHF, GL-SPHF), then the former KV-SPHF (resp. CS-SPHF, GL-SPHF) is $(\varepsilon' + \varepsilon)$-smooth.

4.1.3 Weakly Universal Projective Hash Functions

While the SPHFs we construct from KV-DVSs over $\mathbb{Z}_p$ can easily be proven $1/p$-universal, this is not the case of the ones we construct from GL-DVSs or CS-DVSs. Let us now consider a weaker form of approximation which enables us to define weak versions of universal PHFs corresponding to GL-SPHFs and CS-SPHFs.

4.1.3.1 Weakly Approximate Universality

Definition 4.1.5. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *and a PHF* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *(with algorithms not necessarily running in polynomial time) for the same language* $\mathscr{L} \subseteq \mathcal{X}$ *are said to be* ε-weakly-close*, if for any* lpar*, for any* $\chi \in \mathcal{X}_{\mathsf{lpar}}$*, the following distributions are* ε*-close:*

$$\left\{(\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi), \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right\}$$
$$\left\{(\mathsf{ProjKG}'(\mathsf{hk}, \mathsf{lpar}, \chi), \mathsf{Hash}'(\mathsf{hk}, \mathsf{lpar}, \chi)) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}'(\mathsf{lpar})\right\} .$$

Definition 4.1.6. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε'-weakly-close *to be* ε-universal*, if there exists an* ε*-universal PHF* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *(with algorithms not necessarily running in polynomial time) that is* ε'*-weakly-close to the PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*. It is* weakly close to be universal *or* weakly approximately universal *if it is* ε'*-weakly-close to be* ε*-universal with* ε' *and* ε *negligible in* $\mathfrak{K}$*.*

We remark that if a CS-SPHF (resp. GL-SPHF) is ε'-weakly-close to an ε-smooth CS-SPHF (resp. GL-SPHF), then the former CS-SPHF (resp. GL-SPHF) is $(\varepsilon' + \varepsilon)$-smooth. However, this is not true with KV-SPHFs.

4.1.3.2 Weakly Approximate Universality and GL/CS Smoothness

A weakly approximate universal PHF can be transformed into a GL/CS-smooth PHF. We recall that this does not work for KV smoothness. More precisely, we have the following proposition.

Proposition 4.1.7. *Let* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *be an* ε'*-weakly-close to be* ε*-universal PHF. Let* $\beta = -\log_2 \varepsilon$*. Let* m *be a positive integer. Let* Ext *be the extractor defined in Theorem 2.2.10. Let us define a PHF* $(\mathsf{HashKG}', \mathsf{ProjKG}', \mathsf{Hash}', \mathsf{ProjHash}')$ *with range* $\Pi' = \{0,1\}^m$ *as follows:*

- $\mathsf{HashKG}'(\mathsf{lpar})$ *generates a hashing key* $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$ *and a seed* $\mathsf{seed} \xleftarrow{\$} \mathsf{Ext}$*, and outputs the new hashing key* $\mathsf{hk}' := (\mathsf{hk}, \mathsf{seed})$*;*
- $\mathsf{ProjKG}'(\mathsf{hk}', \mathsf{lpar})$ *computes the projection key* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$ *and outputs the new projection key* $\mathsf{hp}' := (\mathsf{hp}, \mathsf{seed})$*;*
- $\mathsf{Hash}'(\mathsf{hk}', \mathsf{lpar}, \chi)$ *computes the hash value* $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *and outputs the new hash value* $\mathsf{H}' := \mathsf{Ext}(\mathsf{seed}, \mathsf{H})$*;*
- $\mathsf{ProjHash}'(\mathsf{hp}', \mathsf{lpar}, \chi, w)$ *computes the projected hash value* $\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *and outputs the new projected hash value* $\mathsf{pH}' := \mathsf{Ext}(\mathsf{seed}, \mathsf{pH})$*.*

Then, this new PHF is $(\varepsilon' + 2^{(m-\beta)/2})$*-GL/CS-smooth.*

Proof. The proof is immediate from Theorem 2.2.10. □

4.2 Diverse Modules (DMs)

Before introducing diverse modules, let us first formally define graded rings which were sketched in Section 3.1.2.1.

4.2.1 Graded Rings

Graded rings are a generalization of cyclic and bilinear groups and can be used as a practical abstraction of multilinear maps coming from the framework of Garg, Gentry and Halevi [GGH13] and subsequent candidates [CLT13; CLT15; GGH15]. We should however warn the reader that, unfortunately when this thesis was written, due to many attacks [GGH13; CHL+15; CGH+15; HJ15; MF15; CLR15; Mar16], no current candidate multilinear map seems to be useful for applications of hash proof systems, as such applications generally rely on assumptions in the base groups and require encodings of 0. Understanding what ideal multilinear maps can bring us is still an interesting theoretical question.

Furthermore, graded rings are useful even just to deal with cyclic groups, bilinear groups, and groups arising from factorization-based assumptions (e.g., DCR and QR), as they enable to simplify notation.

4.2.1.1 Set of Indexes

A set of indexes Λ is a finite subset of $\mathbb{N}^\tau$ of the form $\{0,\dots,\kappa_1\}\times\cdots\times\{0,\dots,\kappa_\tau\}$, where $\tau,\kappa_1,\dots,\kappa_\tau$ are positive integers. In addition to considering the addition law $+$ over Λ, we also consider Λ as a bounded lattice, with the two following laws:

$$\sup(\tilde{\boldsymbol{v}},\tilde{\boldsymbol{v}}') = (\max(\tilde{\boldsymbol{v}}_1,\tilde{\boldsymbol{v}}'_1),\dots,\max(\tilde{\boldsymbol{v}}_\tau,\tilde{\boldsymbol{v}}'_\tau)) \quad \inf(\tilde{\boldsymbol{v}},\tilde{\boldsymbol{v}}') = (\min(\tilde{\boldsymbol{v}}_1,\tilde{\boldsymbol{v}}'_1),\dots,\min(\tilde{\boldsymbol{v}}_\tau,\tilde{\boldsymbol{v}}'_\tau)).$$

We also write $\tilde{\boldsymbol{v}} < \tilde{\boldsymbol{v}}'$ (resp. $\tilde{\boldsymbol{v}} \le \tilde{\boldsymbol{v}}'$) if and only if for all $i\in\{1,\dots,\tau\}$, $\tilde{\boldsymbol{v}}_i < \tilde{\boldsymbol{v}}'_i$ (resp. $\tilde{\boldsymbol{v}}_i \le \tilde{\boldsymbol{v}}'_i$). Let $\bar{0}=(0,\dots,0)$ and $\top=(\kappa,\dots,\kappa)$, be the minimal and maximal elements.

4.2.1.2 Graded Rings

The (κ,τ)-graded ring over a finite commutative ring R is the set $\mathfrak{G}=\Lambda\times R=\{[\tilde{\boldsymbol{v}},x]\mid \tilde{\boldsymbol{v}}\in\Lambda, x\in R\}$, where $\Lambda=\{0,\dots,\kappa\}^\tau$, with two binary operations $(\boldsymbol{+},\bullet)$ defined as follows:

- for every $u_1=[\tilde{\boldsymbol{v}}_1,x_1], u_2=[\tilde{\boldsymbol{v}}_2,x_2]\in\mathfrak{G}$: $u_1 \boldsymbol{+} u_2 \coloneqq [\sup(\tilde{\boldsymbol{v}}_1,\tilde{\boldsymbol{v}}_2), x_1+x_2]$;
- for every $u_1=[\tilde{\boldsymbol{v}}_1,x_1], u_2=[\tilde{\boldsymbol{v}}_2,x_2]\in\mathfrak{G}$: $u_1\bullet u_2 \coloneqq [\tilde{\boldsymbol{v}}_1+\tilde{\boldsymbol{v}}_2, x_1\cdot x_2]$ if $\tilde{\boldsymbol{v}}_1+\tilde{\boldsymbol{v}}_2\in\Lambda$, or $\perp$ otherwise, where $\perp$ means the operation is undefined and cannot be done.

We remark that $\bullet$ is only a partial binary operation and we use the following convention: $\perp \boldsymbol{+} u = u \boldsymbol{+} \perp = u\bullet\perp = \perp\bullet u = \perp$, for any $u\in\mathfrak{G}\cup\{\perp\}$. Let $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$ be the additive group $\{u=[\tilde{\boldsymbol{v}}',x]\in\mathfrak{G}\mid \tilde{\boldsymbol{v}}'=\tilde{\boldsymbol{v}}\}$ of graded ring elements of index $\tilde{\boldsymbol{v}}$.

Both $\boldsymbol{+}$ and $\bullet$ are associative and commutative, over $\mathfrak{G}\cup\{\perp\}$. More precisely, for any $u_1,u_2,u_3\in\mathfrak{G}\cup\{\perp\}$: $u_1\boldsymbol{+}(u_2\boldsymbol{+}u_3)=(u_1\boldsymbol{+}u_2)\boldsymbol{+}u_3$, $u_1\bullet(u_2\bullet u_3)=(u_1\bullet u_2)\bullet u_3$, $u_1\boldsymbol{+}u_2=u_2\boldsymbol{+}u_1$, and $u_1\bullet u_2=u_2\bullet u_1$. In particular, if $u_1\bullet(u_2\bullet u_3)\neq\perp$ (i.e., if this expression is well-defined), then $(u_1\bullet u_2)\bullet u_3\neq\perp$. In addition, the operation $\bullet$ is distributive over $\boldsymbol{+}$: for any $u_1,u_2,u_3\in\mathfrak{G}\cup\{\perp\}$, $u_1\bullet(u_2\boldsymbol{+}u_3)=u_1\bullet u_2\boldsymbol{+}u_1\bullet u_3$.

We call elements of index $\bar{0}$ *scalars* and we write $\mathfrak{G}_{\bar{0}}$ the ring of scalars.

Thanks to the previous properties, we can make natural use of vector and matrix operations over graded ring elements. In particular, we say that $\mathfrak{G}^n$ and $\mathfrak{G}^{1\times n}$ are *modules* over the graded ring $\mathfrak{G}$. The canonical basis $(\boldsymbol{e}_i)_{i=1}^n$ of $\mathfrak{G}^n$ is defined as usual, except the vectors of the canonical basis are of index $\bar{0}$ (i.e., are scalars).

4.2.1.3 Sub-Graded Rings and Multiplicative Compatibility

A sub-graded ring of a Λ-graded ring $\mathfrak{G}$ is a subset $\mathfrak{G}_{\leq \tilde{\boldsymbol{v}}} = \{u = [\tilde{\boldsymbol{v}}', x] \in \mathfrak{G} \mid \tilde{\boldsymbol{v}}' \leq \tilde{\boldsymbol{v}}\}$ of $\mathfrak{G}$, where $\tilde{\boldsymbol{v}} \in \Lambda$. A sub-graded ring is itself a graded ring. Two sub-graded ring $\mathfrak{G}_1 = \mathfrak{G}_{\leq \tilde{\boldsymbol{v}}_1}$ and $\mathfrak{G}_2 = \mathfrak{G}_{\leq \tilde{\boldsymbol{v}}_2}$ are said to be *multiplicatively compatible* if $\tilde{\boldsymbol{v}}_1 + \tilde{\boldsymbol{v}}_2 \in \Lambda$, or in other words, if it is possible to multiply any element of $\mathfrak{G}_{\leq \tilde{\boldsymbol{v}}_1}$ by an element of $\mathfrak{G}_{\leq \tilde{\boldsymbol{v}}_2}$.

4.2.1.4 Restrictions and Computational Assumptions

Representation. As usual, when we are referring to a graded ring, we are implicitly referring to a family of graded rings indexed by the security parameter $\mathfrak{K}$. We suppose that graded ring elements (including elements of the base ring R) can be represented by bit strings of size *polynomial in* $\mathfrak{K}$, and that the size of the index set Λ if also polynomial in $\mathfrak{K}$.

We suppose that non-scalars have a *unique representation*. This makes notation simpler. In case of non-unique representation, all our constructions can be adapted, as soon as we have a way to randomize the representation and to extract a canonical representation (from which addition and multiplication might not be allowed), as in current candidates for multilinear maps [GGH13].

To handle cases where the order of the base ring is not known, we allow scalars to have non-unique representations. Furthermore the size of their representations might increase when addition and multiplication are performed. Details are given later.

Ring $R = \mathbb{Z}_M$. We also restrict ourselves to the case where R is of the form $\mathbb{Z}_M$ with $M \geq 2$ being some positive integer, as we have not found any graded ring over a more general base ring that might be useful for PHFs in cryptography.

Computational assumptions. Let $\mathfrak{G}$ be a graded ring over the base ring $R = \mathbb{Z}_M$ and let gpar be global parameters containing a description of $\mathfrak{G}$ (later, we will often just write $\mathsf{gpar} = \mathfrak{G}$). The following operations are supposed to be polynomial in $\mathfrak{K}$:

- testing the membership to $\mathfrak{G}$,
- computing the index of an element (the representation might simply contain the index of the element);
- addition and multiplication of two graded ring elements;
- computation of the elements $[\tilde{\boldsymbol{v}}, 0]$, $[\tilde{\boldsymbol{v}}, 1]$, and $[\tilde{\boldsymbol{v}}, -1]$ for any index $\tilde{\boldsymbol{v}}$;[1]
- generation of random scalars (elements of index $\bar{0}$). We denote by $\mathsf{GenScalar}$ a polynomial-time algorithm taking as input the parameters gpar defining the graded ring and outputting a random scalar. The algorithm $\mathsf{GenScalar}$ is $\varepsilon_{\mathsf{GenScalar}}$-*almost-uniform*, if two following distributions are $\varepsilon_{\mathsf{GenScalar}}$-close:

$$\left\{u \xleftarrow{\$} \mathsf{GenScalar}(\mathsf{gpar})\right\} \qquad \left\{u \xleftarrow{\$} R\right\} \quad .$$

[1] We remark that this might not always be possible with some multilinear map candidates. As currently, no candidate can be used for our construction, we do not investigate this issue more precisely, and we consider ideal multilinear maps where these elements can be generated efficiently.

We suppose that GenScalar is *almost uniform*, i.e., it is $\varepsilon_{\mathsf{GenScalar}}$-uniform with $\varepsilon_{\mathsf{GenScalar}}$ negligible in $\mathfrak{K}$. We denote by $\mathsf{GenScalar}^{k\times n}$ the polynomial-time algorithm which takes as input the global parameters gpar and generate a matrix $A \in \mathfrak{G}_{\bar{0}}{}^{k\times n}$, by generating each coefficient $A_{i,j}$ independently as follows: $A_{i,j} \xleftarrow{\$} \mathsf{GenScalar}(\mathsf{gpar})$.

We do not require the order M of the ring R to be efficiently computable, although it has to be finite (and at most exponential in $\mathfrak{K}$). Similarly, in general, we do not require the inversion of scalars to be feasible in polynomial time. However, when we consider prime-order graded rings, we always suppose that the order $M = p$ is known.

4.2.1.5 Examples of Graded Rings

Cyclic groups and bilinear groups of prime order. Let us now show that cyclic groups and bilinear groups of prime order p can be seen as graded rings over $R = \mathbb{Z}_p$:

Cyclic groups: $\Lambda := \{0, 1\}$. More precisely, elements $[0, x]$ of index 0 correspond to scalars $x \in \mathbb{Z}_p$ and elements $[1, x]$ of index 1 correspond to group elements $g^x \in \mathbb{G}$.

Asymmetric bilinear groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$: $\Lambda := \{0, 1\}^2$. More precisely, we can consider the following map: $[(0, 0), x]$ corresponds to $x \in \mathbb{Z}_p$, $[(1, 0), x]$ corresponds to $g_1^x \in \mathbb{G}_1$, $[(0, 1), x]$ corresponds to $g_2^x \in \mathbb{G}_2$ and $[(1, 1), x]$ corresponds to $e(g_1, g_2)^x \in \mathbb{G}_T$.

The two non-trivial sub-graded rings of this bilinear group are $\mathfrak{G}_{\leq(1,0)}$ and $\mathfrak{G}_{\leq(0,1)}$. These two sub-graded rings are multiplicatively compatible, since $(1, 0) + (0, 1) = (1, 1)$. By abuse of notation, we often call these sub-graded rings: $\mathbb{G}_1$ and $\mathbb{G}_2$.

Symmetric bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, e, g)$: $\Lambda := \{0, 1, 2\}$. More precisely, we can consider the following map: $[0, x]$ corresponds to $x \in \mathbb{Z}_p$, $[1, x]$ corresponds to $g^x \in \mathbb{G}$, and $[2, x]$ corresponds to $e(g, g)^x \in \mathbb{G}_T$.

The non-trivial sub-graded ring of this bilinear group is $\mathfrak{G}_{\leq 1}$. This sub-graded ring is multiplicatively compatible with itself, since $1 + 1 = 2$. By abuse of notation, we often call this sub-graded ring: $\mathbb{G}$.

Cyclic groups and bilinear groups of composite order. Everything before can be extended to the case of composite order cyclic group and bilinear groups, as used for example in the BGN cryptosystem [BGN05].

Factorization-based groups. Let us now show two examples where the group order is hard to compute.

Example 4.2.1 ($\mathbb{J}_N$)**.** *Let $N = pq$ be a (public) modulus, with p and q two distinct (secret)* safe prime numbers*, i.e., such that there exists two odd primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$. Unfortunately the multiplicative group $\mathbb{Z}_N^*$ is not cyclic and it seems hard to add a graded ring structure to it.*

Instead, we consider $\mathbb{J}_N$ the subgroup of $\mathbb{Z}_N^*$ containing the elements of Jacobi symbol 1. It is well known that this is a cyclic group of order $M = 2N'$ where $N' = p'q'$. We can then add a structure of graded ring over it like on classic cyclic groups, with $\Lambda = \{0, 1\}$ and $R = \mathbb{Z}_M = \mathbb{Z}_{2N'}$. The only difference is that computing the order M and inverting a scalar is hard, when p and q are not known.

Elements of index $\top$ can be represented directly and uniquely by elements in $\mathbb{J}_N$: concretely, if g is a generator of $\mathbb{J}_N$, we can represent $[\top, x]$ by $g^x \in \mathbb{J}_N$. The difficulty is to represents scalars $\mathfrak{G}_{\bar{0}} \approx \mathbb{Z}_M$. The naive way would be to represent them as elements of $\{0, \ldots, M-1\}$. The problem is that we do not necessarily know M, and M is hard to compute without knowing the factorization of N. For this graded ring, we therefore use non-unique representation, and represents a scalar by an integer not necessarily in $\{0, \ldots, M-1\}$. Operations on scalars correspond to operations on integers without reduction modulo M, so the size of the representation is not a priori bounded. But in all our applications, this does not matter, as only a constant number of operations are done over scalars. Some care should be taken when such an integer is revealed, as this not only reveal its real value (modulo M) but also leak some information about the previous operations. That is why, we often require that no scalars are public (e.g., in a projection key).

Generating a random scalar (in $\mathbb{Z}_M = \mathbb{Z}_{2p'q'}$) can be done by picking uniformly at random an integer x in $\{0, \ldots, \lfloor N/2 \rfloor\}$. The distribution of $x \bmod M$ is $(1/2 \cdot (1/p' + 1/q'))$-close to be uniform, using Proposition 2.1.7, as $\lfloor N/2 \rfloor = 2p'q' + p' + q'$.

Example 4.2.2 ($\mathbb{J}'_{N^2}$)**.** *Similarly to Example 4.2.1, we can also consider the group $\mathbb{J}'_{N^2}$, the subgroup of $\mathbb{Z}^*_{N^2}$ of elements x such that $x \bmod N$ has Jacobi symbol 1. This is also a cyclic group and its order is $2NN'$.*

Historical note 4.2.3. *Graded rings were first introduced in [BBC+13c]. However, before this thesis, graded rings were always considered over a finite field $R = \mathbb{Z}_p$, similarly to what we did in Chapter 3.*

4.2.2 Diverse Modules, Universal PHFs, and Tools for Composite Order

4.2.2.1 Definition

A *diverse module* (DM) is defined similarly to a *diverse vector space* (DVS) except we consider general graded rings over $R = \mathbb{Z}_M$, instead of graded rings over $R = \mathbb{Z}_p$ with p being a prime number. More formally, we have the following definition.

Definition 4.2.4. *A diverse module (DM) is defined by a tuple $\mathcal{M} = (M_{\mathsf{gpar}}, , \mathfrak{G}_{\mathsf{gpar}}, (\mathcal{X}_{\mathsf{lpar}}, \mathscr{L}_{\mathsf{lpar}}, \mathscr{R}_{\mathsf{lpar}}, n_{\mathsf{lpar}}, k_{\mathsf{lpar}}, \mathsf{RGen}, \Gamma_{\mathsf{lpar}}, \boldsymbol{\theta}_{\mathsf{lpar}}, \boldsymbol{\lambda}_{\mathsf{lpar}})_{\mathsf{lpar}})_{\mathsf{gpar}}$ satisfying the same properties as a DVS in Definition 3.1.7, except the graded ring is no more supposed to be over a finite field $\mathbb{Z}_p$ but instead is over $\mathbb{Z}_M$. Furthermore, in case of non-unique representation of scalars, we suppose that no coordinate of $\Gamma_{\mathsf{lpar}}(\chi, \rho)$ has index $\bar{0}$.*

The restriction on the coordinates of $\Gamma_{\mathsf{lpar}}(\chi, \rho)$ in case of non-unique representation of scalars is just to ensure that the projection key (in the PHF construction) will not yield more information than what it would in case of unique representation. Concretely, with the graded ring $\mathbb{J}_N$ defined in Example 4.2.1, if $\Gamma_{\mathsf{lpar}}(\chi, \rho)$ only contained scalars, the projection key of the constructed PHF would contain $\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma_{\mathsf{lpar}}(\chi, \rho)$ a scalar row vector (where $\boldsymbol{\alpha}^\intercal$ is also a scalar row vector), which may reveal too much information on $\boldsymbol{\alpha}^\intercal$ if this is not reduced modulo M. This restriction can be lifted, if we have a way to randomize the representations of scalars, as in [GGH13].

We define *GL diverse modules* (GL-DMs), *CS diverse modules* (CS-DMs), and *KV diverse modules* (KV-DMs), similarly to GL-DVSs, CS-DVSs, and KV-DVSs (see Section 3.1.2.3).

4.2.2.2 The PHF associated to a DM

This is similar to the construction of a SPHF from a DVS (Construction 3.1.10) with several differences:

- scalars are no more in $\mathbb{Z}_p$ and generating a uniform scalar might not be perfect and needs to be done with GenScalar;
- the resulting PHF is not necessarily smooth but only (weakly-)close to be universal;
- the row vector $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times n}$ is extended into a matrix $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_M^{m\times n}$: in the previous construction, $\boldsymbol{\alpha}^\intercal$ defined a (random) linear map from $\mathbb{Z}_p^n$ to $\mathbb{Z}_p$, and now it defines a (random) linear map from $\mathbb{Z}_M^n$ to $\mathbb{Z}_M^m$; this helps to make the universality error probability negligible in $\mathfrak{K}$, when M contains a small prime factor p, as in that case, even when the DM is perfectly sound, the resulting PHF is only $(1/p^m)$-universal. This extension could have been used with DVS but is often just completely useless in that case, as $1/p$ is already negligible in $\mathfrak{K}$;
- and finally, the proofs are more involved as modules are much more complicated to deal with than vector spaces.

More formally, we have the following construction.

Construction 4.2.5 (PHF from DM)**.** *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM. Let m be a positive integer. Then we construct a PHF as follows:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *picks a random matrix* $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathsf{GenScalar}^{m\times n}(\mathsf{gpar})$*, generates a random element* $\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})$*, and outputs the hashing key* $\mathsf{hk} := (\boldsymbol{\alpha}^\intercal, \rho)$*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs the projection key* $\mathsf{hp} := (\boldsymbol{\gamma}^\intercal, \rho)$*, where* $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$ *and*
$$\boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma_{\mathsf{lpar}}(\chi, \rho) \in \mathfrak{G}^{m\times k}\,;$$
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs the hash value*
$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho) \in \mathfrak{G}^m\,,$$
where $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$*;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w)$ *outputs the projected hash value*
$$\mathsf{pH} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\lambda}_{\mathsf{lpar}}(\chi, w, \rho) \in \mathfrak{G}^m\ .$$

When p is a prime number, $m = 1$, and GenScalar uniformly samples an element from $\mathbb{Z}_p$, we get exactly Construction 3.1.10.

We have the following security results.

Theorem 4.2.6 (weakly approximately universal PHF from GL-DM)**.** *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be an ε-sound GL-DM with a $\varepsilon_{\mathsf{GenScalar}}$-almost-uniform algorithm* GenScalar*. Let p be the smallest prime factor of M. Then the PHF described in Construction 4.2.5 is $(mn\varepsilon_{\mathsf{GenScalar}} + \varepsilon)$-weakly-close to be $(1/p^m)$-universal. In particular if $\mathcal{M}$ is sound,* GenScalar *almost uniform, and $1/p^m$ is negligible in $\mathfrak{K}$, then the resulting PHF is weakly approximately universal.*

Theorem 4.2.7 (approximately universal PHF from KV-DM)**.** *Let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a perfectly sound KV-DM with an* $\varepsilon_{\mathsf{GenScalar}}$*-almost-uniform algorithm* GenScalar*. Let* p *be the smallest prime factor of* M*. Then the PHF described in Construction 4.2.5 is* $(mn\varepsilon_{\mathsf{GenScalar}})$*-close to be* $(1/p^m)$*-universal. In particular if* $\varepsilon_{\mathsf{GenScalar}}$ *almost uniform and* $1/p^m$ *is negligible in* $\mathfrak{K}$*, then the resulting PHF is approximately universal.*

When $M = p$ is a prime, the proofs of Theorem 4.2.6 and Theorem 4.2.7 are similar to the proofs of Theorem 3.1.11. But when M is a composite number, this is not anymore the case.

4.2.2.3 Tools for Composite Order

Let us introduce some useful tools to deal with $\mathbb{Z}_M$ when $M \geq 2$ is a composite number.

Chinese remainder theorem (CRT). Let $M \geq 2$ be a positive integer and $M = p_1^{e_1} \cdots p_r^{e_r}$ be its prime decomposition. The *Chinese remainder theorem* (CRT) helps us to move from $\mathbb{Z}_M$ to $\mathbb{Z}_{p_i^{e_i}}$. Let us recall a special case of this theorem.

Theorem 4.2.8 (CRT)**.** *Let* $M \geq 2$ *be a positive integer and* $M = p_1^{e_1} \cdots p_r^{e_r}$ *be its prime decomposition. Then, the following map:*

$$\phi : \left(\begin{array}{rcl} \mathbb{Z}_M & \to & \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}} \\ x & \mapsto & \phi(x) = (x \bmod p_1^{e_1}, \ \ldots, \ x \bmod p_r^{e_r}) \end{array} \right)$$

is a ring isomorphism. Its inverse ϕ^{-1} *is denoted* crt *and defined by:*

$$\mathsf{crt} : \left(\begin{array}{rcl} \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}} & \to & \mathbb{Z}_M \\ (x_1, \ldots, x_r) & \mapsto & \mathsf{crt}(x_1, \ldots, x_r) = \displaystyle\sum_{i=1}^{r} x_i \cdot \frac{M}{p_i^{e_i}} \cdot \left(\left(\frac{M}{p_i^{e_i}} \right)^{-1} \bmod p_i^{e_i} \right) \end{array} \right) .$$

We extend the notation $\mathsf{crt}(x_1, \ldots, x_r)$ to vectors or matrices: if $M_1, \ldots, M_r$ are matrices in $\mathbb{Z}_{p_1^{e_1}}^{k \times n}, \ldots, \mathbb{Z}_{p_r^{e_r}}^{k \times n}$ respectively, then $\mathsf{crt}(M_1, \ldots, M_r)$ is the matrix M such that $M_{i,j} = \mathsf{crt}(M_{1,i,j}, \ldots, M_{r,i,j})$.

Valuation. Let us define the notion of valuation.

Definition 4.2.9. *Let* p *be a prime number and* e *be a positive integer. Let* $M = p^e$*. The* valuation $\nu(x)$ *of an element* $x \in \mathbb{Z}_M$ *is the smallest non-negative integer* k *such that* $x \bmod p^k = 0$*.*

We remark that $0 \leq \nu(x) \leq e$, and $\nu(x) = 0$ if and only if x is invertible, while $\nu(x) = e$ if and only if $x = 0$ (in $\mathbb{Z}_M$). Furthermore, if x and y are two elements in $\mathbb{Z}_M$, then

$$\nu(x \cdot y) = \nu(x) + \nu(y) \ .$$

Intuitively, the valuation $\nu(x)$ indicates "how much x is invertible".
More precisely, we have the following lemma.

Lemma 4.2.10 (division modulo prime power)**.** *Let* p *be a prime number and* e *a positive integer. Let* $M = p^e$*. Let* x *and* y *be two elements in* $\mathbb{Z}_M$*. If* $\nu(x) \geq \nu(y)$*, there exists an element* $z \in \mathbb{Z}_M$ *such that* $z \cdot y = x$*.*

Proof. Let us first prove the existence of z. We first remark that $y/p^{\nu(y)}$ (defined as an integer, for example by looking at a representation of y in $\{0, \dots, M-1\}$) is invertible in $\mathbb{Z}_M$. Let $\tilde{y}$ be the inverse of $y/p^{\nu(y)}$. Then, we can set $z \coloneqq (x/p^{\nu(y)}) \cdot \tilde{y}$, where $x/p^{\nu(y)}$ is well defined as $\nu(x) \geq \nu(y)$. We indeed have:

$$z \cdot y = (x/p^{\nu(y)}) \cdot \tilde{y} \cdot y = x \cdot \left(\tilde{y} \cdot (y/p^{\nu(y)})\right) \quad .$$

□

Existence of solutions to a particular system. For the proof of universality, we also use the following proposition.

Proposition 4.2.11. *Let p be a prime number, and e, n, and k be positive integers. Let $M = p^e$. Let $A \in \mathbb{Z}_M^{k \times n}$ be a matrix. Let $\boldsymbol{b}^\intercal \in \mathbb{Z}_M^{1 \times n}$ be a row vector. If $\boldsymbol{b}^\intercal$ is linearly independent of the rows of the matrix A, then there exists a vector $\boldsymbol{x} \in \mathbb{Z}_M^n$ such that:*

$$\begin{cases} A \cdot \boldsymbol{x} = \boldsymbol{0}_k \\ \boldsymbol{b}^\intercal \cdot \boldsymbol{x} = p^{e-1} \end{cases} ,$$

or in other words:

$$\begin{pmatrix} A \\ \boldsymbol{b}^\intercal \end{pmatrix} \cdot \boldsymbol{x} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p^{e-1} \end{pmatrix} \quad .$$

Proof. Let $\bar{A} \in \mathbb{Z}_M^{(k+1) \times n}$ be the matrix:

$$\bar{A} \coloneqq \begin{pmatrix} A \\ \boldsymbol{b}^\intercal \end{pmatrix} \quad .$$

From a high level point of view, the proof consists in applying Gaussian elimination to the matrix $\bar{A}$, in which at each step we take the coefficient with the smallest valuation as pivot, as in [Jou09, Section 3.3.3.1].

Formally, we do a proof by induction on k.

- *Base case:* $k = 0$. We just need that for any non-zero row vector $\boldsymbol{b}^\intercal \in \mathbb{Z}_M^{1 \times n}$, there exists a column vector $\boldsymbol{x} \in \mathbb{Z}_M^n$ such that $\boldsymbol{b}^\intercal \cdot \boldsymbol{x} = p^{e-1}$. Let us choose $j^* \in \{1, \dots, n\}$ such that $b_{j^*} \neq 0$. Let $c \in \mathbb{Z}_M$ be an arbitrary element such that $b_{j^*} \cdot c = p^{e-1}$. Such an element exists according to Lemma 4.2.10, as $\nu(b_{j^*}) \leq e - 1 = \nu(p^{e-1})$, because $b_{j^*} \neq 0$. We can then construct a vector $\boldsymbol{x} \in \mathbb{Z}_M^n$ as follows: $x_j \coloneqq 0$ for $j \neq j^*$, and $x_{j^*} \coloneqq c$.

- *Induction case:* $k \geq 1$. We have two cases:

  1. There exists an index j^* such that the valuation $\nu(b_{j^*})$ is strictly smaller than the valuations of all the entries of A: for all $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, n\}$, $\nu(A_{i,j}) \geq \nu(b_{j^*}) + 1$. In this case, let $c \in \mathbb{Z}_M$ be an arbitrary element such that $b_{j^*} \cdot c = p^{e-1}$ and let $\boldsymbol{x} \in \mathbb{Z}_M^n$ be the vector defined as follows: $x_j \coloneqq 0$ for $j \neq j^*$, and $x_{j^*} \coloneqq c$. We clearly have $\boldsymbol{b}^\intercal \cdot \boldsymbol{x} = p^{e-1}$. To conclude this case, we just need to prove that $A \cdot \boldsymbol{x} = \boldsymbol{0}$, or in other words that $A_{i,j^*} \cdot c = 0$, for any $i \in \{1, \dots, k\}$. The latter equality comes from the fact that

     $$\nu(A_{i,j^*} \cdot c) = \nu(A_{i,j^*}) + \nu(c) \geq (\nu(b_{j^*}) + 1) + \nu(c) = 1 + \nu(b_{j^*} \cdot c) = 1 + \nu(p^{e-1}) = e \quad .$$

2. There does not exists such an index j^*. In this case, we arbitrarily take two indices i^* and j^* such that the valuation of A_{i^*,j^*} is minimum, among the valuations of all the entries $A_{i,j}$ of A, and therefore also over all the entries $\bar{A}_{i,j}$ of $\bar{A}$. We transform the matrix $\bar{A}$ as follows: for each $i \neq i^*$, we add to the i-th row of $\bar{A}$ the i^*-th row of $\bar{A}$ multiplied by an arbitrary element $c_i \in \mathbb{Z}_M$ such that $A_{i^*,j^*} \cdot c_i = -A_{i,j^*}$ (such an element exists thanks to Lemma 4.2.10). We also permute the i^*-row with the first one, and the j^*-column with the first one. We write the resulting matrix as follows:

$$\bar{A}' = \left(\begin{array}{c|c} d & \boldsymbol{e}^\intercal \\ \hline 0 & \\ \vdots & A'' \\ 0 & \\ \hline 0 & \boldsymbol{b}''^\intercal \end{array}\right) \quad \text{with} \quad \begin{cases} d = A_{i^*,j^*} \in \mathbb{Z}_M \,, \\ \boldsymbol{e}^\intercal = (A_{i^*,j})_{j \in \{1,\ldots,n\} \setminus \{j^*\}}{}^\intercal \in \mathbb{Z}_M^{1\times(n-1)} \,, \\ A'' \in \mathbb{Z}_M^{(k-1)\times(n-1)} \,, \\ \boldsymbol{b}''^\intercal \in \mathbb{Z}_M^{1\times(n-1)} \,. \end{cases}$$

By induction hypothesis, there exists a column vector $\boldsymbol{x}'' \in \mathbb{Z}_M^{n-1}$ such that $A'' \cdot \boldsymbol{x}'' = \boldsymbol{0}$ and $\boldsymbol{b}''^\intercal \cdot \boldsymbol{x}'' = p^{e-1}$. Let c be an arbitrary element $c \in \mathbb{Z}_M$ such that $d \cdot c = -\boldsymbol{e}^\intercal \cdot \boldsymbol{x}''$. Such an element exists according to Lemma 4.2.10, as the valuation of $d = A_{i^*,j^*}$ is not larger than the valuation of e_j for any j (by choice of i^* and j^*), and therefore it is not larger than the valuation of $\boldsymbol{e}^\intercal \cdot \boldsymbol{x}''$. We can then define

$$\boldsymbol{x}' = \begin{pmatrix} c \\ \boldsymbol{x}'' \end{pmatrix} \in \mathbb{Z}_M^n \,.$$

We have

$$\bar{A}' \cdot \boldsymbol{x}' = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p^{e-1} \end{pmatrix} \,.$$

We can finally construct $\boldsymbol{x} \in \mathbb{Z}_M^n$ which corresponds to $\boldsymbol{x}'$ with the first coordinate and the j^*-th coordinate permuted. Since the row operations done on the matrix $\bar{A}$ to obtain $\bar{A}'$ can be undone by subtracting to the i-th row, the i^*-th row multiplied by c_i (for $i \neq i^*$), we have:

$$\bar{A} \cdot \boldsymbol{x} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p^{e-1} \end{pmatrix} \,.$$

This concludes the proof of this case.

This concludes the proof. □

Invertibility. Let us now introduce a convenient lemma that we do not use in the security proof of Theorems 4.2.6 and 4.2.7, but that is useful in other contexts.

Lemma 4.2.12. *Let $M \geq 2$ be a positive integer. Let $x \in \mathbb{Z}_M$. The element x is invertible in $\mathbb{Z}_M$ if and only if, for any prime factor p dividing M, it is invertible modulo p.*

Proof. Let $M = p_1^{e_1} \cdots p_r^{e_r}$ be the prime decomposition of M. The CRT theorem ensures that if x is invertible in $\mathbb{Z}_M$, then it is invertible modulo $p_i^{e_i}$ for all $i \in \{1, \ldots, r\}$, and thus also modulo p for any prime factor p dividing M.

Conversely, let us suppose that x is invertible modulo p for any prime factor p of M. Using the CRT theorem, we just need to prove the case where M is a prime power $M = p^e$. This is done by remarking that x is invertible modulo p if and only if it is coprime with p, which is equivalent to being coprime with p^e and therefore also to being invertible modulo p^e. □

Existence of specific vectors in kernel. Let us conclude with a proposition that we do not use in the security proof of Theorems 4.2.6 and 4.2.7 but that is useful later. Furthermore, it uses the same proof techniques as for Proposition 4.2.11.

Proposition 4.2.13. *Let $M \geq 2$ be a positive integer. Let $A \in \mathbb{Z}_M^{k \times n}$ be a matrix, with $k > n$. There exists a vector $\boldsymbol{x} \in \mathbb{Z}_M$ such that one of its entries is invertible modulo M. Then there exists a vector $\boldsymbol{x} \in \mathbb{Z}_M^n$ and an integer $j \in \{1, \ldots, n\}$, such that $A \cdot \boldsymbol{x} = \mathbf{0}$ and x_j is invertible in $\mathbb{Z}_M$.*

Proof. We start by remarking that thanks to the CRT (Theorem 4.2.8), we just need to prove the theorem in the case where M is a power of a prime number: $M = p^e$, where p is a prime number and e is a positive integer. As for the proof of Proposition 4.2.11, from a high level point of view, the proof consists in applying Gaussian elimination to the matrix $\bar{A}$, in which at each step we take the coefficient with the smallest valuation as pivot, as in [Jou09, Section 3.3.3.1].

Let us now prove the proposition by induction on n.

- *Base case:* $n = 1$. The matrix A is just a row vector of at least 2 entries. We consider two cases:
  1. There exists an index j^* such that $A_{1,j^*} = 0$. In this case, we can construct $\boldsymbol{x} \in \mathbb{Z}_M^n$ as follows: $x_j \coloneqq 0$ for $j \neq j^*$, and $x_{j^*} = 1$.
  2. Otherwise, we arbitrarily take an index j^* such that the valuation $\nu(A_{1,j^*})$ is minimum among the valuations of all the entries of A. Let j' be another distinct index and let $c \in \mathbb{Z}_M$ be an arbitrary element such that $A_{1,j'} = c \cdot A_{1,j^*}$. Then, we can construct a vector $\boldsymbol{x}$ as follows: $x_j \coloneqq 0$ for $j \neq j^*, j'$, $x_{j'} = 1$, $x_{j'} = 1$, and $x_{j^*} = -c$.
- *Induction case:* $n \geq 1$. We arbitrarily take two indices i^* and j^* such that the valuation $\nu(A_{i^*,j^*})$ is minimum among the valuations of all the entries of A. We then transform the matrix A similarly to what was done in the second case of the induction case of the proof of Proposition 4.2.11 and conclude in a similar way. Basically, after permuting the i^*-column and the first one, and after doing some invertible elementary row operations, we get a matrix of the form:

$$A' = \left(\begin{array}{c|c} d & \boldsymbol{e}^\intercal \\ \hline 0 & \\ \vdots & A'' \\ 0 & \end{array}\right) \quad \text{with} \begin{cases} d = A_{i^*,j^*} \in \mathbb{Z}_M \,, \\ \boldsymbol{e}^\intercal = (A_{i^*,j})_{j \in \{1,\ldots,n\} \setminus \{j^*\}}{}^\intercal \in \mathbb{Z}_M^{1 \times (n-1)} \,, \\ A'' \in \mathbb{Z}_M^{(k-1) \times (n-1)} \,. \end{cases}$$

  We then use the induction hypothesis to construct a vector $\boldsymbol{x''} \in \mathbb{Z}_M^{n-1}$ such that $A'' \cdot \boldsymbol{x''} = \mathbf{0}$ and at least one of the entry of this vector $\boldsymbol{x''}$ is invertible in $\mathbb{Z}_M$. Let c be

an arbitrary element $c \in \mathbb{Z}_M$ such that $d \cdot c = -\boldsymbol{e}^\intercal \cdot \boldsymbol{x''}$. Such an element exists according to Lemma 4.2.10, as the valuation of $d = A_{i^*,j^*}$ is not larger than the valuation of e_j for any j (by choice of i^* and j^*), and therefore it is not larger than the valuation of $\boldsymbol{e}^\intercal \cdot \boldsymbol{x''}$. We can then define

$$\boldsymbol{x'} = \begin{pmatrix} c \\ \boldsymbol{x''} \end{pmatrix} \in \mathbb{Z}_M^n \ .$$

We have $A' \cdot \boldsymbol{x'} = \boldsymbol{0}$. We can finally construct $\boldsymbol{x} \in \mathbb{Z}_M^n$ which corresponds to $\boldsymbol{x'}$ with the first coordinate and the j^*-th coordinate permuted.

This concludes the proof. □

4.2.2.4 Security Proofs

Let us now prove Theorems 4.2.6 and 4.2.7. The first idea of the proof is to decompose the order M into its factors $p_i^{e_i}$ and reduce the proof to the case where M is a prime power: $M = p_i^{e_i}$. Then, we follow the proof of Theorem 3.1.11, with one big difference: in the latter proof, we need at some point to find a particular solution $\boldsymbol{x}^{*\intercal}$ to some linear system, namely $\boldsymbol{x}^{*\intercal} \bullet \Gamma = \boldsymbol{0}$ and $\boldsymbol{x}^{*\intercal} \bullet \boldsymbol{\theta} =$ "something", with $\boldsymbol{\theta}$ linearly independent of columns of Γ. When we are over a field $\mathbb{Z}_p$ (as in Theorem 3.1.11), it is easy to prove that such a solution always exists. Here, however, we are over a ring of the form $\mathbb{Z}_{p^e}$ and we need to be very careful about the "something" (namely, choose "something" to be p^{e-1}) and use Proposition 4.2.11.

For the reader familiar with diverse groups, we give some additional intuition to this proof in Section 4.2.3.

Proof of Theorem 4.2.6. Perfect correctness is proven exactly as for DVS (Theorem 3.1.11). Let us now prove smoothness. Until the end of the proof, we are only looking at the discrete logarithms of all the group elements we are considering. In particular, we suppose that all the entries of our vectors and matrices are scalars in $\mathbb{Z}_M$.

Let us suppose that ρ is such that $\boldsymbol{\theta} = \boldsymbol{\theta}(\chi, \rho)$ is linearly independent of the columns of $\Gamma = \Gamma(\chi, \rho)$, which happens with probability at least $1 - \varepsilon$, by ε-soundness of the DVS $\mathcal{V}$. Furthermore, let us assume that $\boldsymbol{\alpha}^\intercal$ is sampled uniformly in $\mathbb{Z}_M^{m \times n}$.

Let us now prove that, for any given projection key hp, the hash value H is uniform in some set containing at least p^m elements. This will prove that the PHF is $(mn\varepsilon_{\mathsf{GenScalar}} + 2\varepsilon)$-weakly-close to be $(1/p^m)$-universal, thanks to Proposition 2.1.6 and to $\varepsilon_{\mathsf{GenScalar}}$-almost-uniformity of $\mathsf{GenScalar}$, which ensures that $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_M^{m \times n}$ is $(mn\varepsilon_{\mathsf{GenScalar}})$-statistically-indistinguishable from $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathsf{GenScalar}^{m \times n}(\mathsf{gpar})$.

Finally, we remark that we can focus on the case $m = 1$. The general case $m \geq 1$ will directly follow, as the rows of $\boldsymbol{\alpha}^\intercal$ are independent, and so are the coefficients of the vector $\mathsf{H} = \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$.

Case: $M = p$ is a prime. Let us start with this simple case. We have $M = p = p$. We can conclude the proof exactly as in the proof of Theorem 3.1.11: for any given projection key hp, H is a uniformly random in $\mathbb{Z}_p = \mathbb{Z}_p$ (a set of size at least p) given only hp.

Case: $M = p^e$ is a power of a prime e. The proof of Theorem 3.1.11 does not work anymore directly. Actually, H might not be uniformly random in $\mathbb{Z}_{p^e}$ at all. For example, if the valuation of all the coordinates of $\boldsymbol{\theta}$ is $e - 1$, then the valuation of H is necessarily $e - 1$ or e. We remark that there is basically only one problematic step: the existence of a vector

$\boldsymbol{x}^{*\intercal} \in \mathbb{Z}_p^{1\times n}$ satisfying $\boldsymbol{x}^{*\intercal} \bullet \Gamma = \mathbf{0}$ and $\boldsymbol{x}^{*\intercal} \bullet \boldsymbol{\theta} = \mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$, for some vector $\boldsymbol{\alpha}^\intercal$. Fortunately, we can fix this step using Proposition 4.2.11.

Formally, let e^* be the minimum non-negative integer such that there exists a vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_p^{1\times n}$ such that $\boldsymbol{x}^\intercal \bullet \Gamma = \mathbf{0}$ and $\boldsymbol{x}^\intercal \bullet \boldsymbol{\theta} = p^{e^*}$. Proposition 4.2.11 ensures that $e^* \leq e-1$, as $\boldsymbol{\theta} \notin \mathrm{ColSpan}\,(\Gamma)$.

Let us now follow the proof of Theorem 3.1.11, and adapt it to our case. Let us fix a row vector $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times k}$ and its associated row vector $\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma$. Let us prove that the hash value H is uniform in the set

$$S'_{\boldsymbol{\gamma}^\intercal} = \{\mathsf{H}' \in \mathbb{Z}_M \mid \nu(\mathsf{H}' - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}) \geq e^*\} \ .$$

First, let us show that a hash value H corresponding to $\boldsymbol{\gamma}^\intercal$ is necessary in this set. By contradiction, if this is not the case, there exists $\boldsymbol{\alpha}'^\intercal \in \mathbb{Z}_p^{1\times k}$ such that:

$$\begin{cases} \boldsymbol{\alpha}'^\intercal \bullet \Gamma = \boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma \\ e' := \nu(\boldsymbol{\alpha}'^\intercal \bullet \boldsymbol{\theta} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}) < e^* \end{cases}$$

Let $\boldsymbol{x}'^\intercal = \boldsymbol{\alpha}'^\intercal - \boldsymbol{\alpha}^\intercal$ and let c be such that $c \bullet (\boldsymbol{x}'^\intercal \bullet \boldsymbol{\theta}) = p^{e'}$, which exists thanks to Lemma 4.2.10 and to the fact that $\nu(\boldsymbol{x}'^\intercal \bullet \boldsymbol{\theta}) = e' = \nu(p^{e'})$. Finally, we get that $\boldsymbol{x}^\intercal := c \bullet \boldsymbol{x}^\intercal$ is such that $\boldsymbol{x}^\intercal \bullet \Gamma = \mathbf{0}$ and $\boldsymbol{x}^\intercal \bullet \boldsymbol{\theta} = p^{e'}$ with $e' < e^*$, which is impossible by choice of e^*. This concludes this first step. This step was not necessary in the proof of Theorem 3.1.11 because the hash value was uniform in the whole set $\mathbb{Z}_p$.

Second, let $\mathsf{H} \in S'_{\boldsymbol{\gamma}^\intercal}$ and let us now show that the set $S_{\boldsymbol{\gamma}^\intercal,\mathsf{H}}$ (as defined in the proof of Theorem 3.1.11) contains a number of vectors $\boldsymbol{\alpha}'^\intercal$ independent of H. Let $c \in \mathbb{Z}_M$ be such that $c \cdot p^{e^*} = \mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}$, which exists as $\nu(\mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}) \geq e^*$. Let $\boldsymbol{x}'^{*\intercal} \in \mathbb{Z}_M^{1\times n}$ be such that $\boldsymbol{x}'^{*\intercal} \bullet \Gamma = \mathbf{0}$ and $\boldsymbol{x}'^{*\intercal} \bullet \boldsymbol{\theta} = p^{e^*}$. This row vector exists by definition of e^*. Let $\boldsymbol{x}^{*\intercal} = c \bullet \boldsymbol{x}'^{*\intercal}$. Then $\boldsymbol{\alpha}^{*\intercal} = \boldsymbol{\alpha}^\intercal + \boldsymbol{x}^{*\intercal}$ is a particular solution of Equation (3.3) on page 67, as:

$$\begin{aligned} \boldsymbol{\alpha}^{*\intercal} \bullet \begin{pmatrix} \Gamma & \boldsymbol{\theta} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{\alpha}^\intercal \bullet \Gamma + \boldsymbol{x}^{*\intercal} \bullet \Gamma & \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} + \boldsymbol{x}^{*\intercal} \bullet \boldsymbol{\theta} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\gamma}^\intercal + (c \bullet 0) & \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} + (c \bullet p^{e^*}) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\gamma}^\intercal + 0 & \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} + (\mathsf{H} - \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{\gamma}^\intercal & \mathsf{H} \end{pmatrix} \ . \end{aligned}$$

Therefore, as in the proof of Theorem 3.1.11, we get:

$$S_{\boldsymbol{\gamma}^\intercal,\mathsf{H}} = \left\{ \boldsymbol{\alpha}^\intercal + \boldsymbol{x}^{*\intercal} + \boldsymbol{x}^\intercal \;\middle|\; \boldsymbol{x} \in \ker \begin{pmatrix} \Gamma & \boldsymbol{\theta} \end{pmatrix}^\intercal \right\} ,$$

and the size of $S_{\boldsymbol{\gamma}^\intercal,\mathsf{H}}$ is independent of H and $\boldsymbol{\gamma}^\intercal$. This proves that H is uniform in $S'_{\boldsymbol{\gamma}^\intercal}$. Furthermore, $S'_{\boldsymbol{\gamma}^\intercal}$ contains $p^{e-e^*} \geq p = p$ elements. This concludes the case $M = p^e$.

Case: M is a general composite number. Let us now handle the case where M is a composite number. Let $M = p_1^{e_1} \cdots p_r^{e_r}$ be the prime decomposition of M.

Let us show that there exists at least one integer $i \in \{1, \ldots, r\}$ such that $\boldsymbol{\theta} \bmod p_i^{e_i} \notin \mathrm{ColSpan}\,(\Gamma \bmod p_i^{e_i})$. By contradiction, if this is not the case, this means that for all $i \in \{1, \ldots, r\}$, there exists a vector $\boldsymbol{\lambda}_i \in \mathbb{Z}_{p_i^{e_i}}^n$, such that $\boldsymbol{\theta} \bmod p_i^{e_i} = \Gamma \bullet \boldsymbol{\lambda}_i \bmod p_i^{e_i}$. Let $\boldsymbol{\lambda} = \mathsf{crt}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_r)$. We have $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$. This is impossible.

Using the previous case, for all i such that $\boldsymbol{\theta} \bmod p_i^{e_i} \notin \mathrm{ColSpan}\,(\Gamma \bmod p_i^{e_i})$, $\mathsf{H} \bmod p_i^{e_i}$ is uniform in a set of size at least $p_i \geq p$. On the other hand, when $\boldsymbol{\theta} \bmod p_i^{e_i} \in \mathrm{ColSpan}\,(\Gamma \bmod p_i^{e_i})$, H is completely determined by $\boldsymbol{\gamma}^\intercal$. Using the CRT again, as what happens modulo each $p_i^{e_i}$ is independent, we get that H is uniform in a set containing at least p elements.

This concludes the general case. □

Proof of Theorem 4.2.7. This proof is similar to the previous one. The only difference is that the DM is perfectly sound. This only changes the beginning of the proof. □

4.2.2.5 Reduced DMs

The proofs of Theorems 4.2.6 and 4.2.7 work by reducing the DM modulo the prime powers $p_i^{e_i}$ in the prime decomposition of the order M. This idea is useful outside these proofs and it is sometimes convenient to look at a reduced version of a DM.

Let us introduce some new notations. If M is a positive integer, M' is a factor of M, and f is a function from some set S to the set of matrices $\mathbb{Z}_M^{k\times n}$, then we denote by $f \bmod M'$ the function from S to the set of matrices $\mathbb{Z}_{M'}^{k\times n}$ defined by $(f \bmod M')(x) = f(x) \bmod M'$, for all $x \in S$. We can extend the modular reduction $\bmod M'$ to graded ring elements, by looking at their discrete logarithm. The operation might not be performed in polynomial time.

Now we can introduce the notion of reduced DM.

Definition 4.2.14. *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM. Let $M' \geq 2$ be a factor of M. The* reduced DM $\mathcal{M}$ modulo M' *is denoted by $\mathcal{M} \bmod M'$ and is defined as:*

$$(M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma \bmod M', \boldsymbol{\theta} \bmod M', \boldsymbol{\lambda} \bmod M') \ .$$

The functions Γ, $\boldsymbol{\theta}$, and $\boldsymbol{\lambda}$ might not be efficiently computable. This does not matter, as the only property we consider on reduced DM is soundness and it only works on the discrete logarithms of the elements.

We have the following theorem.

Theorem 4.2.15 (PHF from reduced DM)**.** *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM with an $\varepsilon_{\mathsf{GenScalar}}$-almost-uniform algorithm* GenScalar*. Let $M' \geq 2$ be a factor of M, such that $\mathcal{M} \bmod M'$ is ε-sound. Let p be the smallest prime factor of M'. Then the PHF described in Construction 4.2.5 is $(mn\varepsilon_{\mathsf{GenScalar}} + \varepsilon)$-weakly-close to be $(1/p^m)$-universal. If in addition $\mathcal{M}$ is a KV-DM, this PHF is $(mn\varepsilon_{\mathsf{GenScalar}})$-close to be $(1/p^m)$-universal.*

This is a refinement of Theorems 4.2.6 and 4.2.7, as when $M' = M$, we get the previous theorems. But, in addition, when the smallest prime factor of M' is larger than the smallest prime factor of M, this improves the bounds of the previous theorems. This is therefore convenient, when M has some small prime factors but M' does not.

Proof. Let $M = p_1^{e_1} \cdots p_r^{e_r}$ be the prime decomposition of M. Without loss of generality, we can write $M = p_1^{e'_1} \cdots p_{r'}^{e'_{r'}}$ the prime decomposition of M', where $r' \leq r$, $e'_1 \leq e_1$, …, $e'_{r'} \leq e_{r'}$. Let M'' be $M'' = p_1^{e_1} \cdots p_{r'}^{e_{r'}}$. We remark that M'' divides M. The smallest prime factor of M'' is the same as the smallest prime factor of M'. Furthermore, if $\mathcal{M} \bmod M'$ is ε-sound, so is $\mathcal{M} \bmod M''$, as if a linear relation is satisfied modulo M'', it is also satisfied modulo M'. Therefore, we can suppose without loss of generality that $M' = M''$.

We can now conclude using a proof similar to the one for the general case of Theorem 4.2.6: the CRT basically ensures that everything that happened modulo M'' is independent of what happens modulo M/M'', and modulo M'', we already get (weakly approximate) universality. □

4.2.2.6 Simple Examples

One of the simplest example is the following.

Example 4.2.16 (DDH with composite order)**.** *We can trivially extend the DVS for the DDH language over a cyclic group $\mathbb{G}$ of prime order p (Example 3.1.8) to a DM for the DDH language over a cyclic group $\mathbb{G}$ of known composite order M. The DM is a perfectly sound KV-DM and the resulting PHF is $1/p$-universal, with p being the smallest prime factor of M. Notice that, when M is a prime p, then $M = p = p$.*

More generally, most examples in Chapter 3 directly work with cyclic groups of composite order, with the important exception of disjunctions of languages which do not work as expected as shown later.

In all the previous examples, the order M was known. Let us now show a simple example of DM over a graded ring with a hard-to-compute order, namely over $\mathbb{J}_N$, with $N = pq$ being a product of two distinct secret safe primes.

Example 4.2.17 (QR)**.** *We work in the graded ring $\mathfrak{G} = \mathbb{J}_N$, with N being the product of two distinct safe primes $p = 2p' + 1$ and $q = 2q' + 1$, as defined in Example 4.2.1. The global parameters are $\mathsf{gpar} = (\mathbb{J}_N, g)$, with g a generator of $\mathbb{J}_N$. There are no language parameters ($\mathsf{lpar} = \perp$) and we consider the language of the quadratic residues of $\mathbb{J}_N$:*

$$\mathscr{L} := \{u \in \mathbb{J}_N \mid \exists r,\, u = g^{2r}\} \quad \subseteq \quad \mathcal{X} = \mathbb{J}_N \ .$$

We can define this language by a KV-DM with $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ the identity functions, and:

$$n := 1 \qquad\qquad k := 1$$
$$\Gamma := \left(g^2\right) \ .$$

Its reduction modulo 2 gives the following DM:

$$n := 1 \qquad\qquad k := 1$$
$$\Gamma := \left([\top, 0]\right)$$
$$\boldsymbol{\theta}(\chi) := \begin{cases} 1 & \textit{if } u \textit{ is a quadratic residue} \\ 0 & \textit{otherwise} \end{cases}$$
$$\boldsymbol{\lambda}(\chi, w) := r \bmod 2\,,$$

where

$$\chi = u \qquad\qquad w = r \ .$$

It is also perfectly sound.

4.2.3 Link with Diverse Groups

In this section, we discuss the link between *diverse modules* (DMs) and *diverse groups* introduced by Cramer and Shoup in [CS02].

Let us first informally recall the notion of diverse groups and their associated PHFs. A diverse group for a language $\hat{\mathscr{L}} \subseteq \hat{\mathcal{X}}$ is a tuple $(\hat{\mathscr{L}}, \hat{\mathcal{X}}, \Pi, \Phi)$ where:

- $(\hat{\mathcal{X}}, \boldsymbol{+})$ is a finite Abelian additive group;
- $\hat{\mathscr{L}}$ is a subgroup of $\hat{\mathcal{X}}$;
- $(\Pi, \boldsymbol{+})$ is a finite Abelian additive group;
- Φ is a subgroup of the group of homomorphisms from $\hat{\mathcal{X}}$ to Π;

satisfying the following property:

- *Diversity.* For any $\theta \in \hat{\mathcal{X}} \backslash \hat{\mathscr{L}}$, there exists a homomorphism $\phi \in \Phi$ such that $\phi(\hat{\mathscr{L}}) = \{0\}$ and $\phi(\theta) \neq 0$. For any set $S \subseteq \hat{\mathcal{X}}$, $\phi(S)$ denotes the image of the set S by the function ϕ.

We also suppose that the language $\hat{\mathscr{L}} \subseteq \mathcal{X}$ is generated by the elements $h_1, \ldots, h_k \in \hat{\mathcal{X}}$ and that a witness $\boldsymbol{\lambda}$ of a word $\theta \in \hat{\mathscr{L}}$ is a vector of integers $\boldsymbol{\lambda} \in \mathbb{Z}^k$ such that

$$\theta = \lambda_1 \bullet \theta_1 \ \boldsymbol{+} \ \cdots \ \boldsymbol{+} \ \lambda_k \bullet h_k \ .$$

We can then construct a PHF associated to such a diverse group, as follows:

- $\mathsf{HashKG}(\mathsf{lpar})$ picks a random homomorphism $\phi \xleftarrow{\$} \Phi$ and outputs the hashing key $\mathsf{hk} := \phi$;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$ outputs the projection key $\mathsf{hp} := \boldsymbol{\gamma}^\intercal := (\phi(h_1), \ldots, \phi(h_k)) \in \Pi^k$;
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \theta)$ outputs the hash value $\phi(\theta) \in \Pi$;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \theta, \boldsymbol{\lambda})$ outputs the projected hash value

$$\mathsf{pH} := \lambda_1 \bullet \gamma_1 \ \boldsymbol{+} \ \cdots \ \boldsymbol{+} \ \lambda_k \bullet \gamma_k .$$

Cramer and Shoup showed that such a PHF is $(1/p)$-universal where p is the smallest prime factor of the order of the quotient group $\hat{\mathcal{X}}/\hat{\mathscr{L}}$.

We first remark that if $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ is a DM, then for any m, if $\Pi := \mathbb{Z}_M^m$ and Φ is the set of linear maps from $\hat{\mathcal{X}} = \mathbb{Z}_M^n$ to Π, then $(\hat{\mathscr{L}}, \hat{\mathcal{X}}, \Pi, \Phi)$ is a diverse group. Actually, the hard part of the proof of Theorem 4.2.6 consists in proving the existence of some vector $\boldsymbol{x'}^{*\intercal}$ satisfying $\boldsymbol{x'}^{*\intercal} \cdot \Gamma = \mathbf{0}$ and $\boldsymbol{x'}^{*\intercal} \cdot \boldsymbol{\theta} \neq 0$, which exactly corresponds to the diversity property.

We may wonder if any diverse group can also be represented by a diverse module. If we forget about complexity of the various algorithms and if we are allowed to slightly change $\hat{\mathcal{X}}$, Π, and Φ, this is actually the case. Let $(\hat{\mathscr{L}}, \hat{\mathcal{X}}, \Pi, \Phi)$ be a diverse group. The fundamental theorem of finitely generated Abelian groups says that we can decompose $\hat{\mathcal{X}}$ as:

$$\hat{\mathcal{X}} = \mathbb{Z}_{M_1} \times \cdots \mathbb{Z}_{M_r} \ ,$$

with $M_1, \ldots, M_R$ being positive integers ≥ 2, such that M_i divides M_{i+1} for all $i \in \{1, \ldots, r-1\}$. We can embed $\hat{\mathcal{X}}$ into the $\mathbb{Z}_M$-submodule $\mathbb{Z}_M^r$ with $M := M_r$. From now on, we suppose that $\hat{\mathcal{X}} = \mathbb{Z}_M^r$. Then, $\hat{\mathscr{L}}$ can now be seen as a submodule of $\mathbb{Z}_M^r$, generated by the column vectors $h_1, \ldots, h_k$.

Next, if we restrict Π to $\bigcup_{\phi \in \Phi} \phi(\mathcal{X})$, the group Π becomes a submodule of some module $\mathbb{Z}_M^m$. We can then suppose that Π is exactly $\mathbb{Z}_M^m$. Finally, we remark that group homomorphisms in Φ are also linear maps from $\mathcal{X} = \mathbb{Z}_M^n$ to $\mathbb{Z}_M^m = \Pi$. We can extend the group Φ to the group of all linear maps. This keeps the diversity property.

At the end, we get a KV-DM with $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ the identity functions. However, we should point out that while all the transformations we made are mathematically correct, they might not be efficient at all, and it might be possible that the resulting KV-DM cannot be used to construct a PHF.

Nevertheless, we do not know of any useful diverse group which cannot be seen as a diverse module. And using diverse modules offers much more structure to work on.

4.3 Conjunctions and Disjunctions

Let us now study conjunctions and disjunctions of DMs. Conjunctions of DMs are similar to conjunctions of DVSs. Disjunctions are much more subtle, when the order M is composite.

4.3.1 Conjunctions

To construct a DM corresponding to the conjunction of two languages $\mathscr{L}_1$ and $\mathscr{L}_2$ represented by two DMs, we can use exactly the same construction as for DVSs (Construction 3.2.1).

Construction 4.3.1 (conjunction of two DMs)**.** *It is exactly the same as Construction 3.2.1 except that the order of the graded ring might now be a composite number M instead of necessarily being a prime number p.*

Everything works exactly the same, as when $\hat{\mathscr{L}}_1$ and $\hat{\mathscr{L}}_2$ are submodules of $\hat{\mathcal{X}}_1$ and $\hat{\mathcal{X}}_2$ respectively, $\hat{\mathscr{L}}_1 \times \hat{\mathscr{L}}_2$ is also a submodule of $\hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2$.

Lemma 4.3.2. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two DMs. If $\mathcal{M}_1$ and $\mathcal{M}_2$ are ε_1-sound and ε_2-sound respectively, then the conjunction DM $\mathcal{M}$ of $\mathcal{M}_1$ and $\mathcal{M}_2$ defined in Construction 4.3.1 is a $(\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2)$-sound DM.*

4.3.2 Disjunctions

We could hope that disjunctions also work as easily as conjunctions. Unfortunately, this is not the case: the straightforward adaptations of Constructions 3.2.4 and 3.2.8 to DMs yields DMs which are not sound, even for really simple examples, as the disjunction of DDH languages of composite order (Examples 3.2.5 and 3.2.9).

Intuitively, in Constructions 3.2.4 and 3.2.8, the main issue is that due to linearity and CRT, if $\boldsymbol{\theta} \in \mathrm{ColSpan}\,(\Gamma)$, this only implies that modulo each factor $p_i^{e_i}$ of M, $\boldsymbol{\theta}_1 \bmod p_i^{e_i} \in \mathrm{ColSpan}\,(\Gamma_1 \bmod p_i^{e_i})$ or $\boldsymbol{\theta}_2 \bmod p_i^{e_i} \in \mathrm{ColSpan}\,(\Gamma_2 \bmod p_i^{e_i})$. But if M has two distinct

prime factors, it is perfectly possible that:

$$\begin{cases} \boldsymbol{\theta}_1 \bmod p \in \mathrm{ColSpan}\,(\Gamma_1 \bmod p) \\ \boldsymbol{\theta}_2 \bmod p \notin \mathrm{ColSpan}\,(\Gamma_2 \bmod p) \\ \boldsymbol{\theta}_1 \bmod q \notin \mathrm{ColSpan}\,(\Gamma_1 \bmod q) \\ \boldsymbol{\theta}_2 \bmod q \in \mathrm{ColSpan}\,(\Gamma_2 \bmod q) \end{cases}$$

and therefore, $\boldsymbol{\theta}_1 \notin \mathrm{ColSpan}\,(\Gamma_1)$ and $\boldsymbol{\theta}_2 \notin \mathrm{ColSpan}\,(\Gamma_2)$, while $\boldsymbol{\theta} \in \mathrm{ColSpan}\,(\Gamma)$.

In this section, we show that the language, for which the disjunction of DMs (using Constructions 3.2.4 and 3.2.8) is sound, still has some interesting properties and can be seen as a "relaxed" disjunction of the two original languages. More precisely, we prove that if $\boldsymbol{\theta}_2 \notin \mathrm{ColSpan}\,(\Gamma_2)$ and does not behave strangely modulo any factor of M, then $\boldsymbol{\theta} \in \mathrm{ColSpan}\,(\Gamma)$ implies that $\boldsymbol{\theta}_1 \in \mathrm{ColSpan}\,(\Gamma_1)$ and $\chi_1 \in \mathscr{L}$. The notion of "not behaving strangely modulo any factor of M" is slightly different for Construction 3.2.4 and Construction 3.2.8 and is captured by the notions of *non-degenerated words* and *strongly non-degenerated words* respectively. But first, we need to introduce this relaxed notion of soundness, and show its influence on universality.

4.3.2.1 Relaxed Soundness and Relaxed Universality

Definition 4.3.3. *A DM* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *is* ε-$\mathscr{S}$-sound*, where* $\mathscr{S}$ *is a subset of* $\mathcal{X}$*, if for any word* $\chi \in \mathscr{S}$*:*

$$\Pr\left[\boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho) \in \mathrm{ColSpan}\,(\Gamma_{\mathsf{lpar}}(\chi, \rho)) \;\middle|\; \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right] \leq \varepsilon\,.$$

when looking at the discrete logarithms of all the elements (see Remark 3.1.6). It is $\mathscr{S}$-sound *if it is* ε-$\mathscr{S}$*-sound, with* ε *negligible in* $\mathfrak{K}$*.*

We remark that classical soundness (resp. ε-soundness) corresponds exactly to $(\mathcal{X} \setminus \mathscr{L})$-soundness (resp. ε-$(\mathcal{X} \setminus \mathscr{L})$-soundness).

Definition 4.3.4. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε-$\mathscr{S}$-universal*, where* $\mathscr{S}$ *is a subset of* $\mathcal{X}$*, if for any* lpar*, any word* $\chi \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$*, any projection key* hp*, and any hash value* $\mathsf{H} \in \Pi$*, we have*

$$\begin{aligned} \Pr\left[\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi) = \mathsf{H} \text{ and } \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) = \mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right] \\ \leq \varepsilon \cdot \Pr\left[\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) = \mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})\right]\,. \end{aligned}$$

It is $\mathscr{S}$-universal *if it is* ε-$\mathscr{S}$*-universal with* ε *is negligible in* $\mathfrak{K}$*.*

We can extend Theorems 4.2.6, 4.2.7 and 4.2.15 to use relaxed soundness, as follows.

Theorem 4.3.5 (PHF from DM with relaxed soundness)**.** *Let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a DM with an* $\varepsilon_{\mathsf{GenScalar}}$*-almost-uniform algorithm* $\mathsf{GenScalar}$*. Let* $\mathscr{S}$ *be a subset of* $\mathcal{X}$*. Let* $M' \geq 2$ *be a factor of* M*, such that* $\mathcal{M} \bmod M'$ *is* ε-$\mathscr{S}$*-sound. Let* p *be the smallest prime factor of* M'*. Then the PHF described in Construction 4.2.5 is* $(mn\varepsilon_{\mathsf{GenScalar}} + \varepsilon)$*-weakly-close to be* $1/p^m$-$\mathscr{S}$*-universal. If in addition* $\mathcal{M}$ *is a KV-DM, this PHF is* $(mn\varepsilon_{\mathsf{GenScalar}})$*-close to be* $1/p^m$-$\mathscr{S}$*-universal.*

The proof is a straightforward adaptation of the one for Theorem 4.2.15.

We could have introduced this notion of relaxed soundness from the beginning and defined everything using it. However, that would have made notation more complex. As in all our applications, we only use these notions of relaxed soundness or relaxed universality with disjunctions of DMs, we prefer to introduce these notions only here.

Another way to look at all these relaxed notions is to say that we are working with promise languages: we only consider words that are either in the language $\mathscr{L}$ or in the set $\mathscr{S}$, but we do not consider words that are neither in $\mathscr{L}$ nor in $\mathscr{S}$.

4.3.2.2 Disjunctions of GL-DVSs and Non-Degeneracy

Counter-example. Let us first show that the DM $\mathcal{M}$ in Example 3.2.5 is not sound, when the prime order p is just replaced by a composite order $M = pq$, with p and q being two distinct prime numbers. We recall that for any $x \in \mathbb{Z}_p$ and any $y \in \mathbb{Z}_q$, $\mathsf{crt}(x, y)$ is the unique element $z \in \mathbb{Z}_M$, such that $z \bmod p = x$ and $z \bmod q = y$ (see Section 4.2.2.3). We have $\mathsf{crt}(x, x) = x$, and in particular, $\mathsf{crt}(1, 1) = 1$ and $\mathsf{crt}(-1, -1) = -1$. Furthermore, crt is a ring homomorphism from $\mathbb{Z}_p \times \mathbb{Z}_q$ to $\mathbb{Z}_M$. Concretely, this implies that for any $x, x' \in \mathbb{Z}_p$ and any $y, y' \in \mathbb{Z}_q$, we have $\mathsf{crt}(x, y) + \mathsf{crt}(x', y') = \mathsf{crt}(x + x',\, y + y')$ and $\mathsf{crt}(x, y) \cdot \mathsf{crt}(x', y') = \mathsf{crt}(x \cdot x',\, y \cdot y')$.

In the DM $\mathcal{M}$, we have:

$$\Gamma(\chi) := \begin{pmatrix} 0 & g & 0 & g \\ g_1 & u_1 & 0 & 0 \\ h_1 & v_1 & 0 & 0 \\ 0 & 0 & g_2 & u_2 \\ 0 & 0 & h_2 & v_2 \end{pmatrix} \qquad \boldsymbol{\theta}(\chi) := \begin{pmatrix} g^{-1} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \\ 1_{\mathbb{G}} \end{pmatrix} .$$

We consider the word:

$$\chi = (u_1 := g_1^{\mathsf{crt}(1,1)},\ v_1 := h_1^{\mathsf{crt}(1,0)},\ u_2 := g_2^{\mathsf{crt}(1,1)},\ v_2 := h_2^{\mathsf{crt}(0,1)}) \in \mathcal{X} \setminus \mathscr{L} .$$

This word χ is not in $\mathscr{L}$, as $\chi_1 := (u_1, v_1)$ is not a DH tuple in basis (g_1, h_1) and $\chi_2 := (u_2, v_2)$ is not a DH in basis (g_2, h_2). But, we have $\boldsymbol{\theta} = \Gamma(\chi) \bullet \boldsymbol{\lambda}$ with:

$$\boldsymbol{\lambda} = \begin{pmatrix} \mathsf{crt}(1, 0) \\ \mathsf{crt}(-1, 0) \\ \mathsf{crt}(0, 1) \\ \mathsf{crt}(0, -1) \end{pmatrix} .$$

Therefore $\mathcal{M}$ is not sound.

We remark that $\chi_1 \bmod p$ and $\chi_2 \bmod q$ are DH tuples in bases (g_1, h_1) and (g_2, h_2) respectively (where mod works on the discrete logarithms of the elements, as already defined). In some sense, χ_1 and χ_2 are degenerated modulo p and q respectively.

Non-degeneracy. Let us now introduce formally this notion of non-degenerated words.

Definition 4.3.6. *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM. A word $\chi \in \mathcal{X}$ is* $(1 - \varepsilon)$-non-degenerated (in $\mathcal{M}$) *if for any prime factor p of the order M, when looking at the discrete logarithms of all the elements (see Remark 3.1.6), we have:*

$$\Pr\left[\boldsymbol{\theta}(\chi, \rho) \bmod p \notin \mathrm{ColSpan}\,(\Gamma \bmod p) \mid \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right] \leq \varepsilon .$$

It is non-degenerated (in $\mathcal{M}$) *if it is* $(1-\varepsilon)$*-non-degenerated with* ε *negligible in* $\mathfrak{K}$.

We remark that any non-degenerated word χ is necessarily in $\mathcal{X} \setminus \mathscr{L}$. Furthermore, when $M = p$ is a prime number (i.e., when $\mathcal{M}$ is a DVS), then non-degenerated words are exactly words in $\mathcal{X} \setminus \mathscr{L}$.

Let us look at three examples.

Example 4.3.7 (QR)**.** *Let us consider the DM* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *for the QR language. We use the notation of Example 4.2.17. In this DM, any word* $\chi = u = g^r$ *is degenerated, as* $r \bmod p'$ *is linearly dependent of the columns of* $\Gamma \bmod p'$ *(where the modulo is done on the discrete logarithms). But if we look at the reduction modulo* 2 *of this* $\mathcal{M}$*, any word in* $\mathcal{X} \setminus \mathscr{L}$ *is non-degenerated. Actually, modulo* 2*, this DM is even a DVS.*

Example 4.3.8 (DDH)**.** *Let us now consider the DM* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *for the DDH language for a cyclic group of composite order* M*. We use the notation of Example 4.2.16. In this DM, there are many words that are not in the language* $\mathscr{L}$ *but that are degenerated. For example, for any proper factor* $1 < M' < M$ *of* M*, the word* $\chi = (g^{M'}, h^{2 \cdot M'})$ *is degenerated but is not a DH tuple.*

However, a uniform word in $\mathcal{X} = \mathbb{G}^2$ *is non-degenerated with probability*

$$\prod_{p \text{ prime factor of } M} (1 - 1/p)\,,$$

as modulo p *(a prime factor of* M*), a random word* χ *is not a DH tuple with probability* $1 - 1/p$*. In particular, if the smallest prime factor* p *of* M *has at least* $\mathfrak{K}$ *bits, a uniform word in* $\mathcal{X} = \mathbb{G}^2$ *is non-degenerated with overwhelming probability. We use this property in our construction of NIZK from DMs for example.*

We can extend the previous example to any MDDH assumption.

Example 4.3.9 (MDDH)**.** *Let* $\mathfrak{G}$ *be a graded ring of order* M*,* $\tilde{\boldsymbol{v}}$ *be some index. Let* $\mathcal{D}$ *be a distribution of matrices in* $\mathbb{Z}_M^{n \times k}$*, with* $n > k$*. We can extend the* $\mathcal{D}$*-MDDH assumption recalled in Section 3.4.1 to this setting and construct a DM* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *associated to the* $\mathcal{D}$*-MDDH assumption.*

Let us show that when all the prime factors of M *have at least* $\mathfrak{K}$ *bits, a uniform word* $\chi = \boldsymbol{\theta} \xleftarrow{\$} \mathcal{X} = \mathfrak{G}_{\tilde{\boldsymbol{v}}}^n$ *is non-degenerated with overwhelming probability (for any matrix* $\Gamma = [A]$*). For any prime factor* p *of* M*, and any matrix* $\Gamma \in \mathfrak{G}_{\tilde{\boldsymbol{v}}}^{n \times k}$*, a uniform vector* $\boldsymbol{\theta} \xleftarrow{\$} \mathfrak{G}_{\tilde{\boldsymbol{v}}}^n$ *reduced modulo* p *is not in the column space of* $\Gamma \bmod p$*, except with probability at most* $1/p$ *(as modulo* p*, when looking at the discrete logarithms, the column space of* Γ *is a proper subspace of* $\mathbb{Z}_p^n$*, because* $n > k$*). As in Example 4.3.9, the probability that a uniform word* $\chi \xleftarrow{\$} \mathcal{X}$ *is non-degenerated is at most*

$$\prod_{p \text{ prime factor of } M} (1 - 1/p)\ .$$

This probability is negligible when the smallest prime factor p *of* M *has at least* $\mathfrak{K}$ *bits.*

Construction. We can define the GL disjunction of two GL-DMs as follows.

Construction 4.3.10 (GL disjunction)**.** *It is exactly the same as Construction 3.2.4 except that the order of the graded ring might now be a composite number* M *instead of necessarily being a prime number* p*.*

As already explained, the disjunction DM of two DMs as defined in Construction 4.3.1 is not sound, but is $\mathscr{S}$-sound for any set $\mathscr{S}$ of words $(\mathfrak{x}_1, \mathfrak{x}_2) \notin \mathscr{L}$, such as $\mathfrak{x}_1$ or $\mathfrak{x}_2$ is non-degenerated. More formally, we have the following proposition.

Proposition 4.3.11. *Let* $\mathcal{M}_1 = (M, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{M}_2 = (M, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DMs, that are* ε_1*-sound and* ε_2*-sound respectively. For* $i \in \{1, 2\}$*, let* $\mathscr{S}_i$ *be the set of* ε'_i*-non-degenerated words of* $\mathcal{M}_i$*, for some* ε'_i*. Let* $\varepsilon := \max(1 - \varepsilon_1 - \varepsilon'_2 + \varepsilon_1\varepsilon'_2,\ 1 - \varepsilon'_1 - \varepsilon_2 + \varepsilon'_1\varepsilon_2)$ *and*

$$\mathscr{S} := ((\mathcal{X}_1 \setminus \mathscr{L}_1) \times \mathscr{S}_2 \quad \cup \quad \mathscr{S}_1 \times (\mathcal{X}_2 \setminus \mathscr{L}_2))$$

Then, the GL disjunction DM $\mathcal{M}$ *of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *as defined in Construction 4.3.10 is* ε*-*$\mathscr{S}$*-sound.*

Proof. In this proof, we only look at the discrete logarithms of all the elements. Let us suppose that $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \in (\mathcal{X}_1 \setminus \mathscr{L}_1) \times \mathscr{S}_2$, and let us prove that

$$\Pr\left[\boldsymbol{\theta} \notin \mathrm{ColSpan}\left(\Gamma(\mathfrak{x}, \rho)\right) \mid \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right] \leq 1 - \varepsilon_1 - \varepsilon'_2 + \varepsilon_1\varepsilon'_2 \ .$$

The other case $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \in \mathscr{S}_1 \times (\mathcal{X}_2 \setminus \mathscr{L}_2)$ is similar.

We suppose that $\rho = (\rho_1, \rho_2)$ is such that, for all prime factor p of M:

$$\begin{cases} \boldsymbol{\theta}_1(\mathfrak{x}_1, \rho_1) \notin \mathrm{ColSpan}\left(\Gamma_1(\mathfrak{x}_1, \rho_1)\right) \ , \\ \boldsymbol{\theta}_2(\mathfrak{x}_2, \rho_2) \bmod p \notin \mathrm{ColSpan}\left(\Gamma_2(\mathfrak{x}_2, \rho_2) \bmod p\right) \ . \end{cases}$$

This happens with probability at least $1 - \varepsilon_1 - \varepsilon'_2 + \varepsilon_1\varepsilon'_2$.

By contradiction, let us suppose that there exists a vector $\boldsymbol{\lambda} \in \mathbb{Z}_p^k$ such that $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$. We necessarily have $\lambda_{k_1+1} + \lambda_{k_1+k_2+1} = -1$ (or both), due to the first row of the matrix Γ, as the (k_1+1)-th column of Γ is $(1_{\mathbb{Z}_p}, \boldsymbol{\theta}_1^\intercal, \mathbf{0}^\intercal)^\intercal$ and the (k_1+k_2+1)-th column of Γ is its last column $(1_{\mathbb{Z}_p}, \mathbf{0}^\intercal, \boldsymbol{\theta}_2^\intercal)^\intercal$. With DVS (i.e., prime order $M = p$), we could conclude directly, by saying that $\lambda_{k_1+1} \neq 0$ or $\lambda_{k_1+k_2+1} \neq 0$, thus one of these two scalars is invertible, and use this fact in an equation like Equation (3.7) (on page 72) or Equation (4.1). But here we cannot, as a non-zero scalar is not necessarily invertible.

Let us now show that for any prime factor p of M, we have $\lambda_{k_1+k_2+1} \bmod p = 0$. By contradiction, if there exists some prime p dividing M such that $\lambda_{k_1+k_2+1} \neq 0$, $\lambda_{k_1+k_2+1}$ is invertible modulo p, and we have:

$$\boldsymbol{\theta}_2 \bmod p = \Gamma_2 \bullet \boldsymbol{\lambda}' \bmod p \qquad \text{with } \boldsymbol{\lambda}' = -(\lambda_i)_{i=k_1+2,\ldots,k_1+k_2+1} \ / \ \lambda_{k_1+k_2+1} \bmod p \,,$$

which is impossible, as $\boldsymbol{\theta}_2 \bmod p \notin \mathrm{ColSpan}\left(\Gamma_2 \bmod p\right)$.

Thus for any prime factor p of M, $\lambda_{k_1+k_2+1} \bmod p = 0$ and therefore $\lambda_{k_1+1} \bmod p = -1$. Lemma 4.2.12 implies that λ_{k_1+1} is invertible modulo M. And this time, we can conclude as with DVSs:

$$\boldsymbol{\theta}_1 = \Gamma_1 \bullet \boldsymbol{\lambda}' \qquad \text{with } \boldsymbol{\lambda}' = -(\lambda_i)_{i=1,\ldots,k_1} \ / \ \lambda_{k_1+1} \ . \tag{4.1}$$

This is impossible, as $\boldsymbol{\theta}_1 \bmod p \notin \mathrm{ColSpan}\left(\Gamma_1 \bmod p\right)$. And that concludes the proof. □

4.3.2.3 Disjunctions of CS-DMs and Strongly Non-Degenerated Words

We might hope that disjunctions of CS-DMs work the same way as disjunctions of GL-DMs: we can use the same construction as disjunctions of CS-DVSs (Construction 3.2.8) and just "remove" the words which are degenerated to get relaxed soundness. Unfortunately, this is not sufficient and we need to introduce a stronger non-degeneracy notion.

Counter-examples. First, degenerated words create the same problem as for GL-DMs. If we take Example 3.2.9 and just change the prime order p to a composite order $M = pq$ with p and q two distinct primes, we have the same counterexample as before. We recall that, in this DM:

$$\Gamma(\mathfrak{x}) := \begin{pmatrix} g_1 & 0 & g_2 & 0 \\ 0 & g_1 & h_2 & 0 \\ h_1 & 0 & 0 & g_2 \\ 0 & h_1 & 0 & h_2 \end{pmatrix} \qquad \boldsymbol{\theta}(\mathfrak{x}) := \begin{pmatrix} u_1 \bullet u_2 \\ u_1 \bullet v_2 \\ v_1 \bullet u_2 \\ v_1 \bullet v_2 \end{pmatrix} = \begin{pmatrix} e(u_1, u_2) \\ e(u_1, v_2) \\ e(v_1, u_2) \\ e(v_1, v_2) \end{pmatrix} .$$

We can consider the word:

$$\mathfrak{x} = (u_1 := g_1^{\mathsf{crt}(1,1)},\ v_1 := h_1^{\mathsf{crt}(1,0)},\ u_2 := g_2^{\mathsf{crt}(1,1)},\ v_2 := h_2^{\mathsf{crt}(0,1)}) \in \mathcal{X} \setminus \mathscr{L} .$$

This word is not in $\mathscr{L}$, as $\mathfrak{x}_1 := (u_1, v_1)$ is not a DH tuple in basis (g_1, h_1) and $\mathfrak{x}_2 := (u_2, v_2)$ is not a DH tuple in basis (g_2, h_2). But, we have $\boldsymbol{\theta} = \Gamma(\mathfrak{x}) \bullet \boldsymbol{\lambda}$ with:

$$\boldsymbol{\lambda} = \begin{pmatrix} \mathsf{crt}(1,0) \bullet u_2 \\ \mathsf{crt}(1,0) \bullet v_2 \\ \mathsf{crt}(0,1) \bullet u_1 \\ \mathsf{crt}(0,1) \bullet v_1 \end{pmatrix} = \begin{pmatrix} \mathsf{crt}(1,0) \bullet g_2 \\ 0 \\ \mathsf{crt}(0,1) \bullet g_1 \\ 0 \end{pmatrix} \qquad \text{as } \boldsymbol{\theta} = \begin{pmatrix} \mathsf{crt}(1,1) \bullet g_1 \bullet g_2 \\ \mathsf{crt}(0,1) \bullet g_1 \bullet h_2 \\ \mathsf{crt}(1,0) \bullet h_1 \bullet g_2 \\ \mathsf{crt}(0,0) \bullet h_1 \bullet h_2 \end{pmatrix} .$$

Second, even non-degenerated words may create problems. Let us show this on an example. Let the global parameters consist of a bilinear group $(M, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of composite order $M = p^2$ with p being a prime number. Let $\mathcal{M}_1 = (M, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ be the KV-DM for the the DDH language in $\mathfrak{G}_1 = \mathbb{G}_1$. Let $\mathcal{M}_2 = (M, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be the following KV-DM in $\mathfrak{G}_2 = \mathbb{G}_2$:

$$\begin{aligned} \mathscr{L}_2 &:= \{u_2 \in \mathbb{G}_2 \mid \exists r_2 \in \mathbb{Z}_M,\ u_2 = g_2^{pr_2}\} \subseteq \mathbb{G}_2 \\ n_2 &:= 1 & k_2 &:= 1 \\ \Gamma_2(\mathfrak{x}_2) &:= \begin{pmatrix} g_2^p \end{pmatrix} \\ \boldsymbol{\theta}_2(\mathfrak{x}_2) &:= \begin{pmatrix} u_2 \end{pmatrix} & \boldsymbol{\lambda}_2(\mathfrak{x}_2, \mathfrak{w}_2) &:= \begin{pmatrix} r_2 \end{pmatrix} , \end{aligned}$$

where:

$$\mathfrak{x}_2 = (u_2) \qquad \mathfrak{w}_2 = (r_2) .$$

Following Construction 3.2.8, the CS/KV disjunction of these two DMs is the following DM

$\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$:

$$\mathscr{L} := \{(u_1, v_1, u_2) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \mid \exists r_1, r_2 \in \mathbb{Z}_M,\ (u_1, v_1) = (g_1^{r_1}, h_1^{r_1}) \text{ or } u_2 = g_2^{pr_2}\}$$

$$n := 2 \qquad\qquad k := 1$$

$$\Gamma(\mathfrak{x}) := \begin{pmatrix} g_1 & g_2^p & 0 \\ h_1 & 0 & g_2^p \end{pmatrix}$$

$$\boldsymbol{\theta}(\mathfrak{x}) := \begin{pmatrix} u_1 \bullet u_2 \\ v_1 \bullet u_2 \end{pmatrix} = \begin{pmatrix} e(u_1, u_2) \\ e(v_1, u_2) \end{pmatrix}$$

$$\boldsymbol{\lambda}(\mathfrak{x}, \mathfrak{w}) := \begin{cases} \begin{pmatrix} r_1 \bullet u_2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} u_2^{r_1} \\ 0 \\ 0 \end{pmatrix} & \text{if } (u_1, v_1) = (g^{r_1}, h^{r_1}) \\ \begin{pmatrix} 0 \\ r_2 \bullet u_1 \\ r_2 \bullet v_1 \end{pmatrix} = \begin{pmatrix} 0 \\ u_1^{r_2} \\ v_1^{r_2} \end{pmatrix} & \text{otherwise, if } u_2 = g^{pr_2} \end{cases}$$

where:

$$\mathfrak{x} = (u_1, v_1, u_2) \qquad\qquad \mathfrak{w} = (r_1, r_2)\ .$$

Then, we can consider the word

$$\mathfrak{x} = (u_1 := g_1^p,\ v_1 := h_1^{2p},\ u_2 := g_2)$$

which is not in $\mathscr{L}$. Furthermore, u_2 is non-degenerated. But we still have $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$ with

$$\boldsymbol{\lambda} = \begin{pmatrix} 0 \\ g_1 \\ h_1^2 \end{pmatrix}\ .$$

Strong non-degeneracy. Let us now introduce the notion of strong non-degeneracy.

Definition 4.3.12. *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM. A word $\mathfrak{x} \in \mathcal{X}$ is* $(1-\varepsilon)$-strongly-non-degenerated (in $\mathcal{M}$) *if when looking at the discrete logarithms of all the elements (see Remark 3.1.6):*

$$\Pr\left[\exists \boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1\times n},\ \begin{cases} \boldsymbol{x}^\intercal \bullet \Gamma = \boldsymbol{0}^\intercal \in \mathbb{Z}_M^{1\times k} \\ \boldsymbol{x}^\intercal \bullet \boldsymbol{\theta}(\mathfrak{x}) = 1_{\mathbb{Z}_M} \end{cases} \middle| \ \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right]\ .$$

It is strongly non-degenerated (in $\mathcal{M}$) *is it is $(1-\varepsilon)$-non-degenerated with ε negligible in $\mathfrak{K}$.*

We remark that any strongly non-degenerated word $\mathfrak{x}$ is in $\mathcal{X} \setminus \mathscr{L}$. Furthermore, when $M = p$ is a prime number (i.e., when $\mathcal{M}$ is a DVS), then strongly non-degenerated words are exactly words in $\mathcal{X} \setminus \mathscr{L}$. And when M is a product of distinct prime numbers, then the CRT ensures that non-degenerated words are exactly strongly non-degenerated words.

However, as seen in the above counter-example, this is not necessarily the case when M is divisible by p^2 for some prime p.

Let us give two examples.

Example 4.3.13 (QR)**.** *For the DM $\mathcal{M}$ for the QR language (Examples 4.2.17 and 4.3.7), the order is a product of three distinct primes 2, p', and q'. Therefore, non-degenerated words correspond to strongly non-degenerated words in this DM and any of its reductions. In particular, any word $\mathfrak{x} \in \mathcal{X} \setminus \mathscr{L}$ is strongly non-degenerated in $\mathcal{M}$* mod 2.

Example 4.3.14 (MDDH)**.** *Let us consider DMs for $\mathcal{D}$-MDDH language, over a graded ring of composite order M, as in Example 4.3.9. We suppose that the smallest prime factor p of M has at least $\mathfrak{K}$ bits. If M is a product of distinct prime numbers, everything easily works as before, since a non-degenerated word is also strongly non-degenerated: a uniform word $\mathfrak{x} \in \mathcal{X} = \mathfrak{G}_{\mathfrak{v}}^n$ is strongly non-degenerated with overwhelming probability.*

Let us now prove that a uniform word in $\mathfrak{G}_{\mathfrak{v}}^n$ is strongly non-degenerated with overwhelming probability in the general case, when $M \geq 2$ is any positive integer. This works for any distribution $\mathcal{D}$. Let A be any arbitrary matrix in $\mathbb{Z}_M^{n \times k}$. For this proof, we only look at the discrete logarithms of the elements and forget their indices (in $\mathfrak{G}$). Proposition 4.2.13 (applied on the transposed of A) shows that there exists a row vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^n$ such that $\boldsymbol{x}^\intercal \cdot A = \mathbf{0}^\intercal$ and one of the entry x_{j^*} of $\boldsymbol{x}^\intercal$ is invertible in $\mathbb{Z}_M$. We remark that if a word $\boldsymbol{\theta} \in \mathbb{Z}_M^n$ is such that $\boldsymbol{x}^\intercal \cdot \boldsymbol{\theta}$ is invertible, then this word is strongly non-degenerated. When $\boldsymbol{\theta}$ is a uniform vector in $\mathbb{Z}_M^n$, $\boldsymbol{x}^\intercal \cdot \boldsymbol{\theta}$ is uniformly random in $\mathbb{Z}_M$ thanks to the term $\theta_{i^*} \cdot x_{i^*}$ which is uniformly random in $\mathbb{Z}_M$. As a uniform scalar element in $\mathbb{Z}_M$ is invertible with probability:

$$\prod_{p \text{ prime factor of } M} (1 - 1/p)\,,$$

we conclude that a random vector $\boldsymbol{\theta} \xleftarrow{\$} \mathbb{Z}_M^n$ is strongly non-degenerated with at least this probability (which is overwhelming).

Construction. We can define the disjunction of two CS-DMs as follows.

Construction 4.3.15 (CS/KV disjunction)**.** *It is exactly the same as Construction 3.2.8 except that the order of the graded ring might now be a composite number M instead of necessarily being a prime number p.*

As already explained, the disjunction DM of two DMs as defined in Construction 4.3.15 is not sound, but is $\mathscr{S}$-sound for any set $\mathscr{S}$ of words $(\mathfrak{x}_1, \mathfrak{x}_2) \notin \mathscr{L}$, such as $\mathfrak{x}_1$ or $\mathfrak{x}_2$ is strongly non-degenerated. More formally, we have the following proposition.

Proposition 4.3.16. *Let $\mathcal{M}_1 = (M, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{M}_2 = (M, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two CS-DMs over two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. We suppose that $\mathcal{M}_1$ and $\mathcal{M}_2$ are ε_1-sound and ε_2-sound respectively. For $i \in \{1, 2\}$, let $\mathscr{S}_i$ be the set of ε_i'-non-degenerated words of $\mathcal{M}_i$, for some ε_i'. Let $\varepsilon := \max(1 - \varepsilon_1 - \varepsilon_2' + \varepsilon_1\varepsilon_2', 1 - \varepsilon_1' - \varepsilon_2 + \varepsilon_1'\varepsilon_2)$ and*

$$\mathscr{S} := ((\mathcal{X}_1 \setminus \mathscr{L}_1) \times \mathscr{S}_2) \quad \cup \quad (\mathscr{S}_1 \times (\mathcal{X}_2 \setminus \mathscr{L}_2)) \quad .$$

Then, the disjunction CS-DM $\mathcal{M}$ of $\mathcal{M}_1$ and $\mathcal{M}_2$ as defined in Construction 4.3.15 is an ε-$\mathscr{S}$-sound CS-DM. Furthermore, if $\mathcal{M}_1$ and $\mathcal{M}_2$ are KV-DMs and $\varepsilon_1' = \varepsilon_2' = 0$, then $\mathcal{M}$ is a $\mathscr{S}$-sound KV-DM.

We remark that this proposition directly implies Lemma 3.2.10, as when $M = p$ is prime, $\mathscr{S} = \mathcal{X} \setminus \mathscr{L}$. This gives a new proof of Lemma 3.2.10 using only Equation (3.8) on page 73 (and no more complex properties of tensors).

Proof. In this proof, we only look at the discrete logarithms of all the elements. Let us suppose that $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \in (\mathscr{X}_1 \setminus \mathscr{L}_1) \times \mathscr{S}_2$, and let us prove that

$$\Pr\left[\boldsymbol{\theta} \notin \mathrm{ColSpan}\left(\Gamma(\mathfrak{x}, \rho)\right) \mid \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})\right] \leq 1 - \varepsilon_1 - \varepsilon_2' + \varepsilon_1\varepsilon_2' \ .$$

The other case $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2) \in \mathscr{S}_1 \times (\mathscr{X}_2 \setminus \mathscr{L}_2)$ is similar.

We suppose that $\rho = (\rho_1, \rho_2)$ is such that, there exists a row vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1\times n_2}$ such that:

$$\begin{cases} \boldsymbol{\theta}_1(\mathfrak{x}_1, \rho_1) \notin \mathrm{ColSpan}\left(\Gamma_1(\mathfrak{x}_1, \rho_1)\right) \ , \\ \boldsymbol{x}^\intercal \bullet \Gamma_2 = \mathbf{0} \, , \\ \boldsymbol{x}^\intercal \bullet \boldsymbol{\theta}_2(\mathfrak{x}_2, \rho_2) = 1 \ . \end{cases}$$

This happens with probability at least $1 - \varepsilon_1 - \varepsilon_2' + \varepsilon_1\varepsilon_2'$.

By contradiction, let us suppose that there exists a vector $\boldsymbol{\lambda} \in \mathbb{Z}_M^k$ such that $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$. We then remark that:

$$\begin{aligned} (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet \boldsymbol{\theta} &= (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \\ &= (\mathsf{Id}_{n_1} \bullet \boldsymbol{\theta}_1) \otimes (\boldsymbol{x}^\intercal \bullet \boldsymbol{\theta}_2) \\ &= \boldsymbol{\theta}_1 \otimes (1_{\mathbb{Z}_M}) \\ &= \boldsymbol{\theta}_1 \end{aligned}$$

and

$$\begin{aligned} (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet \Gamma &= (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet \begin{pmatrix} \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2 \end{pmatrix} \\ &= \begin{pmatrix} (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) & (\mathsf{Id}_{n_1} \otimes \boldsymbol{x}^\intercal) \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \end{pmatrix} \\ &= \begin{pmatrix} (\mathsf{Id}_{n_1} \bullet \Gamma_1) \otimes (\boldsymbol{x}^\intercal \bullet \mathsf{Id}_{n_2}) & (\mathsf{Id}_{n_1} \bullet \mathsf{Id}_{n_1}) \otimes (\boldsymbol{x}^\intercal \bullet \Gamma_2) \end{pmatrix} \\ &= \begin{pmatrix} \Gamma_1 \otimes \boldsymbol{x}^\intercal & \mathsf{Id}_{n_1} \otimes \mathbf{0}_{k_2}^\intercal \end{pmatrix} \\ &= \begin{pmatrix} \Gamma_1 \otimes \boldsymbol{x}^\intercal & \mathbf{0}_{n_1 \times (n_1 k_2)} \end{pmatrix} \ . \end{aligned}$$

As $\boldsymbol{\theta} = \Gamma \bullet \boldsymbol{\lambda}$, we have:

$$\boldsymbol{\theta}_1 = \begin{pmatrix} \Gamma_1 \otimes \boldsymbol{x}^\intercal & \mathbf{0}_{n_1 \times (n_1 k_2)} \end{pmatrix} \bullet \boldsymbol{\lambda} \, ,$$

and thus:

$$\boldsymbol{\theta}_1 \in \mathrm{ColSpan}\left(\Gamma_1 \otimes \boldsymbol{x}^\intercal\right) \ .$$

If we write $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_{k_1}$ the columns of the matrix Γ_1, then the columns of the matrix $\Gamma_1 \otimes \boldsymbol{x}^\intercal$ are of the form $x_i \cdot \boldsymbol{C}_j$. Therefore $\mathrm{ColSpan}\left(\Gamma_1 \otimes \boldsymbol{x}^\intercal\right) \subseteq \mathrm{ColSpan}\left(\Gamma_1\right)$ and $\boldsymbol{\theta}_1 \in \mathrm{ColSpan}\left(\Gamma_1\right)$. This is impossible. This concludes the proof. □

4.4 t-Universality, t-Smoothness, and t-Soundness

Universality and smoothness ensure a certain randomness of hash values of words outside the language, when we are only given the projection key. This also holds when we are additionally given hash values of words inside the language, as these hash values can be computed using the projection key. But universality and smoothness do not guarantee anything when we are also given a hash value of a word outside the language.

That is why, we now introduce two stronger security notions for PHF: t-universality and t-smoothness, which deal with this case, or more precisely with the case where we are also given $t-1$ hash values of words potentially outside the language. We then show that if a CS-DVS or CS-DM satisfies a new property called t-soundness, its associated PHF satisfies t-smoothness and (weakly approximate) t-universality respectively. We conclude by constructing t-sound CS-DVS or CS-DM from any classical CS-DVS or CS-DM.

Historical note 4.4.1. *Cramer and Shoup defined a notion of 2-universality in [CS02]. Our notion is a slight variant of theirs with one main difference: the use of an explicit tag. This enables to work with statistical definitions while having the possibility to plug in a collision-resistant hash function to get efficiency improvements.*

In this whole section, for the sake of simplicity, we only consider PHFs for which the projection key hp does not depend on the word x. We could consider notions of t-universality and t-smoothness for PHFs in which the projection key hp depends on the word x: in this case, all the seen hash values would correspond to the same word but with different tags. Our definitions and constructions in this section can easily be extended to this case. This case was actually used in [BCPW15], but in this thesis, we improved the efficiency of the construction in the latter paper and we got rid of the part using t-smoothness (see Section 6.3).

4.4.1 t-Universality and t-Smoothness

We first need to extend the notion of PHF to add a tag, before defining t-universality and t-smoothness.

4.4.1.1 Tag-PHF

A *tag-PHF* is similar to a PHF, except that ProjHash and Hash takes an additional input, called a tag $\mathsf{tag} \in \mathsf{Tags}$. More formally, we have the following definition:

Definition 4.4.2. *A* tag-PHF *over* $(\mathscr{L}_{\mathsf{lpar}})$ *is defined by a tuple of four polynomial-time algorithms* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*, where:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *generates a hashing key* hk *for the language parameters* lpar*;*
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, x)$ *deterministically derives a projection key* hp *from the hashing key* hk*, the language parameters* lpar*, and possibly the word* $x \in \mathscr{X}_{\mathsf{lpar}}$*;*
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x, \mathsf{tag})$ *deterministically outputs a hash value* H *from the hashing key* hk*, for the word* $x \in \mathscr{X}_{\mathsf{lpar}}$*, the language parameters* lpar*, and the tag* $\mathsf{tag} \in \mathsf{Tags}$*; the set* Tags *is supposed to be efficiently recognizable;*
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, x, w, \mathsf{tag})$ *deterministically outputs a projected hash value* pH *from the projection key* hp*, and the witness* w*, for the word* $x \in \mathscr{L}_{\mathsf{lpar}}$ *(i.e.,* $\mathscr{R}_{\mathsf{lpar}}(x, w) = 1$*), the language parameters* lpar*, and the tag* $\mathsf{tag} \in \mathsf{Tags}$*.*

A PHF has to satisfy the following property:

- Perfect correctness. *For any* lpar*, for any word* $x \in \mathscr{L}_{\mathsf{lpar}}$ *with witness* w *(i.e., such that* $\mathscr{R}_{\mathsf{lpar}}(x, w) = 1$*), for any tag* $\mathsf{tag} \in \mathsf{Tags}$*, for any* $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$ *and for* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$*,*

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x, \mathsf{tag}) = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, x, w, \mathsf{tag}) \ .$$

Tag-PHFs are often just called PHFs in the sequel.

As usual with tags or labels, the tag is no useful for basic universality or smoothness: a smooth or universal PHF can be transformed in a tag-PHF for any tag set Tags just by ignoring the tag tag. But tags are necessary for t-universality and t-smoothness.

4.4.1.2 t-Universality and t-Smoothness

Definition 4.4.3. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε-t-$\mathscr{S}$-universal *(where t is a positive integer and $\mathscr{S}$ is a subset of $\mathcal{X}$) if the algorithm* ProjKG *does not use its input* $\mathfrak{x}$*, and if for any* lpar*, any words* $\mathfrak{x}_1, \ldots, \mathfrak{x}_{t-1} \in \mathcal{X}_{\mathsf{lpar}}$*, any word* $\mathfrak{x}_t \in \mathscr{S}$*, any tags* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1} \in \mathsf{Tags}$*, any tag* $\mathsf{tag}_t \in \mathsf{Tags}$ *distinct from* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$*, any projection key* hp*, and any hash values* $\mathsf{H}_1, \ldots, \mathsf{H}_t \in \Pi$*, we have*

$$\Pr\left[\begin{array}{l|l} \begin{cases} \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_t, \mathsf{tag}_t) = \mathsf{H}_t; \\ \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) = \mathsf{hp}; \\ \forall i \in \{1, \ldots, t-1\}, \\ \qquad \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_i, \mathsf{tag}_i) = \mathsf{H}_i \end{cases} & \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}) \end{array}\right]$$
$$\leq \varepsilon \cdot \Pr\left[\begin{array}{l|l} \begin{cases} \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}) = \mathsf{hp}; \\ \forall i \in \{1, \ldots, t-1\}, \\ \qquad \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_i, \mathsf{tag}_i) = \mathsf{H}_i \end{cases} & \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}) \end{array}\right] .$$

*It is t-$\mathscr{S}$-*universal *if it is ε-t-universal with ε is negligible in $\mathfrak{K}$. It is ε-t-*universal *(resp. t-*universal*), if it ε-t-$(\mathcal{X} \setminus \mathscr{L})$-universal (resp. t-$(\mathcal{X} \setminus \mathscr{L})$-universal).*

When $t = 2$, $\mathsf{Tags} = \mathcal{X}$ and $\mathsf{tag} = \mathfrak{x}$, we get back the notion of 2-universality of Cramer and Shoup.

Definition 4.4.4. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε-t-CS-smooth *if the algorithm* ProjKG *does not use its input* $\mathfrak{x}$*, and if for any* lpar*, any words* $\mathfrak{x}_1, \ldots, \mathfrak{x}_{t-1} \in \mathcal{X}_{\mathsf{lpar}}$*, any word* $\mathfrak{x}_t \in \mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$*, any tags* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1} \in \mathsf{Tags}$*, and any tag* $\mathsf{tag}_t \in \mathsf{Tags}$ *distinct from* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$*, then the following distributions are ε-close:*

$$\left\{ (\mathsf{hp}, \mathsf{H}_1, \ldots, \mathsf{H}_t) \;\middle|\; \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \\ \forall i \in \{1, \ldots, t-1\},\ \mathsf{H}_i \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_i, \mathsf{tag}_i); \\ \mathsf{H}_t \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_t, \mathsf{tag}_t) \end{array} \right\}$$
$$\left\{ (\mathsf{hp}, \mathsf{H}_1, \ldots, \mathsf{H}_t) \;\middle|\; \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \\ \forall i \in \{1, \ldots, t-1\},\ \mathsf{H}_i \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \mathfrak{x}_i, \mathsf{tag}_i); \\ \mathsf{H}_t \xleftarrow{\$} \Pi \end{array} \right\} .$$

*A PHF is t-*CS-smooth *if it is ε-t-CS-smooth with ε negligible in $\mathfrak{K}$.*

Definition 4.4.5. *A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is* ε-t-KV-smooth *if the algorithm* ProjKG *does not use its input* $\mathfrak{x}$*, and if for any* lpar*, any functions* $f_1, \ldots, f_t$ *from the set of possible projection keys* hp *to* $\mathcal{X}_{\mathsf{lpar}}$*, and any functions* $g_1, \ldots, g_t$ *from the set of possible projection keys* hp *to* Tags *such that the image of* f_t *is included in*

$\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$, *and for any projection key* hp, $g_t(\mathsf{hp})$ *is different from* $g_1(\mathsf{hp}), \ldots, g_{t-1}(\mathsf{hp})$, *then the following distributions are* ε*-close:*

$$\left\{ (\mathsf{hp}, \mathsf{H}_1, \ldots, \mathsf{H}_t) \;\middle|\; \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \\ \forall i \in \{1, \ldots, t-1\},\ \mathsf{H}_i \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f_i(\mathsf{hp}), g_i(\mathsf{hp})); \\ \mathsf{H}_t \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f_t(\mathsf{hp}), g_t(\mathsf{hp})) \end{array} \right\}$$

$$\left\{ (\mathsf{hp}, \mathsf{H}_1, \ldots, \mathsf{H}_t) \;\middle|\; \begin{array}{l} \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar}); \\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}); \\ \forall i \in \{1, \ldots, t-1\},\ \mathsf{H}_i \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f_i(\mathsf{hp}), g_i(\mathsf{hp})); \\ \mathsf{H}_t \xleftarrow{\$} \Pi \end{array} \right\}.$$

A PHF is t-KV-smooth *if it is* t*-*ε*-KV-smooth with* ε *negligible in* $\mathfrak{K}$*.*

4.4.2 t-Soundness

To construct t-universal and t-smooth PHFs, we first need to define the notion of t-sound (and t-$\mathscr{S}$-sound) tag-DMs and tag-DVSs. Then, we show that t-sound tag-DMs and tag-DVSs directly provide (weakly approximate) t-universal and t-smooth PHFs.

4.4.2.1 t-Sound Tag-DM and Tag-DVS

Tag-DMs and tag-DVSs are similar to DMs and DVSs except that $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ take an additional input: a tag $\mathsf{tag} \in \mathsf{Tags}$. Correctness is modified accordingly, as for tag-PHF. Similarly to tag-PHFs, tag-DMs and tag-DVSs are often just called DMs and DVSs.

Definition 4.4.6. *A DM* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *is* ε*-*t*-*$\mathscr{S}$-sound*, where* $\mathscr{S}$ *is a subset of* $\mathcal{X}$*, if for any words* $x_1, \ldots, x_{t-1} \in \mathcal{X}_{\mathsf{lpar}}$*, any word* $x_t \in \mathscr{S}$*, any tags* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1} \in \mathsf{Tags}$*, any tag* $\mathsf{tag}_t \in \mathsf{Tags}$ *distinct from* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$*:*

$$\Pr\left[\boldsymbol{\theta}_{\mathsf{lpar}}(x_t, \rho, \mathsf{tag}_t) \in \mathrm{ColSpan}\left(\left(\Gamma_{\mathsf{lpar}}(\rho) \quad \boldsymbol{\theta}_{\mathsf{lpar}}(x_1, \rho, \mathsf{tag}_1) \quad \ldots \quad \boldsymbol{\theta}_{\mathsf{lpar}}(x_{t-1}, \rho, \mathsf{tag}_{t-1}) \right) \right) \;\middle|\; \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar}) \right] \leq \varepsilon\,,$$

when looking at the discrete logarithms of all the elements (see Remark 3.1.6). It is t*-*$\mathscr{S}$*-*sound*, if it is* ε*-*t*-*$\mathscr{S}$*-sound, with* ε *negligible in* $\mathfrak{K}$*. It is* ε*-*t*-*sound *(resp.* t*-*sound*), if it* ε*-*t*-*$(\mathcal{X} \setminus \mathscr{L})$*-sound (resp.* t*-*$(\mathcal{X} \setminus \mathscr{L})$*-sound).*

4.4.2.2 From t-Sound Tag-CS-DMs and Tag-CS-DVSs to (Weakly Approximate) t-Universal and t-Smooth PHFs

We can construct a tag-PHF from a tag-CS-DM, in a straightforward way.

Construction 4.4.7 (tag-PHF from tag-CS-DM)**.** *Let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a tag-CS-DM. Let* m *be a positive integer. Then we construct a PHF as follows:*

- $\mathsf{HashKG}(\mathsf{lpar})$ *picks a random matrix* $\boldsymbol{\alpha}^{\intercal} \xleftarrow{\$} \mathsf{GenScalar}^{m \times n}(\mathsf{gpar})$*, generates a random element* $\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})$*, and outputs the hashing key* $\mathsf{hk} := (\boldsymbol{\alpha}^{\intercal}, \rho)$*;*

- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *outputs the projection key* $\mathsf{hp} := (\boldsymbol{\gamma}^\intercal, \rho)$, *where* $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$ *and*

$$\boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma_{\mathsf{lpar}}(\chi, \rho) \in \mathfrak{G}^{m \times k} ;$$

- $\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi, \mathsf{tag})$ *outputs the hash value*

$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}_{\mathsf{lpar}}(\chi, \rho, \mathsf{tag}) \in \mathfrak{G}^m ,$$

  where $\mathsf{hk} = (\boldsymbol{\alpha}^\intercal, \rho)$*;*

- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w, \mathsf{tag})$ *outputs the projected hash value*

$$\mathsf{pH} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\lambda}_{\mathsf{lpar}}(\chi, w, \rho, \mathsf{tag}) \in \mathfrak{G}^m \quad .$$

We can then extend Theorem 4.3.5 as follows.

Theorem 4.4.8 (tag-PHF from tag-CS-DM with t-soundness)**.** *Let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be a tag-CS-DM with an* $\varepsilon_{\mathsf{GenScalar}}$*-almost-uniform algorithm* $\mathsf{GenScalar}$*. Let* $\mathscr{S}$ *be a subset of* $\mathcal{X}$ *and* t *be a positive integer. Let* $M' \geq 2$ *be a factor of* M*, such that* $\mathcal{M} \bmod M'$ *is* ε*-*t*-*$\mathscr{S}$*-sound. Let* p *be the smallest prime factor of* M'*. Then the PHF described in Construction 4.4.7 is* $(mn\varepsilon_{\mathsf{GenScalar}} + 2\varepsilon)$*-weakly-close to be* $1/p^m$*-*t*-*$\mathscr{S}$*-universal. If in addition* $\mathcal{M}$ *is a KV-DM, this PHF is* $(mn\varepsilon_{\mathsf{GenScalar}})$*-close to be* $1/p^m$*-*t*-*$\mathscr{S}$*-universal.*

Similarly, in the case of a DVS, we have the following theorem, which is an extension of Theorem 3.1.11.

Theorem 4.4.9 (tag-PHF from tag-CS-DVS with t-soundness)**.** *Let* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be an* ε*-*t*-sound tag-CS-DVS. Then the PHF described in Construction 4.4.7 is an* ε*-*t*-smooth CS-SPHF. If in addition* $\mathcal{V}$ *is a KV-DVS, this PHF is a perfectly* t*-smooth KV-SPHF.*

Both these theorems can be proven by remarking that we can just consider the hash values of the words $\chi_1, \ldots, \chi_{t-1}$ with tags $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$ as part of the projection key.

4.4.3 Construction of t-Sound Tag-CS-DMs and Tag-CS-DVSs

We now show how to construct t-sound tag-CS-DM and tag-CS-DVS, from any classical CS-DM and CS-DVS. For that, we need introduce a useful tool: t-independent strongly non-degenerated functions.

4.4.3.1 t-Independent Strongly Non-Degenerated Function

Definition. For our construction of t-sound tag-CS-DM and tag-CS-DVS, we need the following tool.

Definition 4.4.10. *Let* $t \geq 1$*,* $s \geq 1$*, and* $M \geq 2$ *be integers. A* t-independent strongly non-degenerated function modulo M *is a function* $\boldsymbol{\phi}$ *from* Tags *to* $\mathbb{Z}_M^s$ *for some positive integer* s*, such that, for any elements* $\mathsf{tag}_1, \ldots, \mathsf{tag}_t \in \mathsf{Tags}$*, where* tag_t *is different from* $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$*, there exists a row vector* $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1 \times s}$*, such that:*

$$\boldsymbol{x}^\intercal \cdot \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \ldots & \boldsymbol{\phi}(\mathsf{tag}_{t-1}) & \boldsymbol{\phi}(\mathsf{tag}_t) \end{pmatrix} = \begin{pmatrix} 0 & \ldots & 0 & 1 \end{pmatrix} \quad .$$

We remark that a t-independent strongly non-degenerated function modulo M from Tags to $\mathbb{Z}_M^s$ is also a t-independent strongly non-degenerated function modulo M from any subset $\mathsf{Tags}' \subseteq \mathsf{Tags}$ to $\mathbb{Z}_M^s$.

We have the following straightforward lemma.

Lemma 4.4.11. *Let $t \geq 1$, $s \geq 1$ and $M \geq 2$ be integers. A function $\boldsymbol{\phi}$ from Tags to $\mathbb{Z}_M^s$ is a t-independent strongly non-degenerated function modulo M, if and only if, for any integer $t' \in \{1, \ldots, t\}$, for any* distinct *elements $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t'} \in \mathsf{Tags}$, there exists a row vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1\times s}$, such that:*

$$\boldsymbol{x}^\intercal \cdot \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \ldots & \boldsymbol{\phi}(\mathsf{tag}_{t'-1}) & \boldsymbol{\phi}(\mathsf{tag}_{t'}) \end{pmatrix} = \begin{pmatrix} 0 & \ldots & 0 & 1 \end{pmatrix} .$$

First construction. Let us show now a first construction.

Theorem 4.4.12. *Let $t \geq 1$ and $M \geq 2$ be integers. Let p be the smallest prime factor of M and let Tags be the set $\{0, \ldots, p-1\}$. Let $\boldsymbol{\phi}$ be the function defined as follows:*

$$\boldsymbol{\phi} : \left(\begin{array}{ccc} \mathsf{Tags} & \to & \mathbb{Z}_M^t \\ \mathsf{tag} & \mapsto & \begin{pmatrix} 1 \\ \mathsf{tag} \bmod M \\ \mathsf{tag}^2 \bmod M \\ \vdots \\ \mathsf{tag}^{t-1} \bmod M \end{pmatrix} \end{array} \right) .$$

Then $\boldsymbol{\phi}$ is a t-independent strongly non-degenerated function modulo M from Tags to $\mathbb{Z}_M^s$, with $s = t$.

Proof. Let $\mathsf{tag}_1, \ldots, \mathsf{tag}_t \in \mathsf{Tags}$ be tags, such that tag_t is different from $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$. Let $T \in \mathbb{Z}_M^{s\times t}$ be the matrix:

$$T \coloneqq \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \ldots & \boldsymbol{\phi}(\mathsf{tag}_t) \end{pmatrix} = \begin{pmatrix} 1 & \ldots & 1 \\ \mathsf{tag}_1 \bmod M & \ldots & \mathsf{tag}_t \bmod M \\ \vdots & & \vdots \\ \mathsf{tag}_1^{t-1} \bmod M & \ldots & \mathsf{tag}_t^{t-1} \bmod M \end{pmatrix} .$$

We want to construct $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1\times t}$ such that

$$\boldsymbol{x}^\intercal \cdot T = \begin{pmatrix} 0 & \ldots & 0 & 1 \end{pmatrix} .$$

If M was a prime number, this would be completely straightforward, as T is a Vandermonde matrix and therefore is full-rank, if we remove duplicated tags tag_i, with $i \in \{1, \ldots, t-1\}$. Here, M might be composite and we therefore provide a manual proof of the existence of $\boldsymbol{x}^\intercal$.

Let $P(X) \in \mathbb{Z}_M[X]$ be the polynomial defined as follows:

$$P(X) = \frac{(X - \mathsf{tag}_1) \cdots (X - \mathsf{tag}_{t-1})}{(\mathsf{tag}_t - \mathsf{tag}_1) \cdots (\mathsf{tag}_t - \mathsf{tag}_{t-1})} .$$

This polynomial is well-defined, as for any $i \in \{1, \ldots, t-1\}$, the integer $\mathsf{tag}_t - \mathsf{tag}_i$ is in $\{-p+1, \ldots, p-1\} \setminus \{0\}$, so it is coprime with M, and thus invertible in $\mathbb{Z}_M$. Furthermore, $P(X)$ has degree $t-1$ and is such that:

$$P(\mathsf{tag}_i) = \begin{cases} 0 & \text{if } i \in \{1, \ldots, t-1\}\,, \\ 1 & \text{if } i = t\ . \end{cases}$$

Let $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1\times t}$ be the vector such that $P(X) = x_1 + x_2 \cdot X + \cdots + x_t \cdot X^{t-1}$. We have:

$$\boldsymbol{x}^\intercal \cdot T = \begin{pmatrix} P(\mathsf{tag}_1) & \ldots & P(\mathsf{tag}_{t-1}) & P(\mathsf{tag}_t) \end{pmatrix} = \begin{pmatrix} 0 & \ldots & 0 & 1 \end{pmatrix}\ .$$

□

Second construction. The first construction is limited to subsets of $\mathsf{Tags} = \{0, \ldots, p-1\}$ with p the smallest prime factor of M. When this prime p is small (e.g., when it is 2), this is often insufficient.

We point out that this limitation is not a limitation of the proof, but really a limitation of the construction. In any set of at least $p+1$ integers Tags, there exist two distinct integers tag_1 and tag_2 such that $\mathsf{tag}_1 \bmod p = \mathsf{tag}_2 \bmod p$ because of the pigeonhole principal. The matrix

$$T := \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \boldsymbol{\phi}(\mathsf{tag}_2) \end{pmatrix}$$

is therefore equal to $\begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \boldsymbol{\phi}(\mathsf{tag}_1) \end{pmatrix}$ modulo p and there cannot exists a vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^2$ such that $\boldsymbol{x}^\intercal \cdot T = (0, 1)$. Therefore, $\boldsymbol{\phi}$ is not even 2-independent strongly non-degenerated, when Tags contains at least $p+1$ integers.

The idea of the new construction is to work in field extensions. We start with the case $M = p^e$, where p is a prime number.

For this section, for any prime number p and any integer $\nu \geq 1$, we consider an arbitrary linear isomorphism $\boldsymbol{\psi}_{p,\nu}$ from from the finite field $\mathbb{F}_{p^\nu}$ to the vector space $\mathbb{Z}_p^\nu$. Both $\boldsymbol{\psi}_{p,\nu}$ and its inverse $\boldsymbol{\psi}_{p,\nu}^{-1}$ are efficiently computable. Furthermore, we implicitly use the canonical embeddings from $\mathbb{Z}_p$ to $\mathbb{Z}_{p^e}$ and to $\mathbb{F}_{p^\nu}$.

Theorem 4.4.13. *Let t, e, and ν be positive integers. Let $M = p^e$, where p is a prime number. Let $\mathsf{Tags} := \{0, \ldots, p-1\}^\nu = \mathbb{Z}_p^\nu$. Let $\boldsymbol{\phi}$ be the function defined as follows:*

$$\boldsymbol{\phi} : \begin{pmatrix} \mathsf{Tags} & \to & \mathbb{Z}_M^{1+\nu\cdot(t-1)} \\ \mathsf{tag} & \mapsto & \begin{pmatrix} 1 \\ \boldsymbol{\psi}_{p,\nu}(\xi) \\ \boldsymbol{\psi}_{p,\nu}(\xi^2) \\ \vdots \\ \boldsymbol{\psi}_{p,\nu}(\xi^{t-1}) \end{pmatrix} \end{pmatrix},$$

where $\xi := \boldsymbol{\psi}_{p,\nu}^{-1}(\mathsf{tag}) \in \mathbb{F}_{p^\nu}$. Then $\boldsymbol{\phi}$ is a t-independent strongly non-degenerated function modulo M from Tags to $\mathbb{Z}_M^s$, with $s = 1 + \nu \cdot (t-1)$.

Proof. We do the proof using Lemma 4.4.11 and by induction over e. More precisely, let $t' \in \{1, \dots, t\}$ and let $\mathsf{tag}_1, \dots, \mathsf{tag}_{t'}$ be distinct tags in Tags. For all $i \in \{1, \dots, t'\}$, we define $\xi_i \coloneqq \psi_{p,\nu}^{-1}(\mathsf{tag}_i)$. Let $T \in \mathbb{Z}_M^{s \times t}$ be the matrix:

$$T \coloneqq \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \dots & \boldsymbol{\phi}(\mathsf{tag}_{t'}) \end{pmatrix} = \begin{pmatrix} 1 & \dots & 1 \\ \boldsymbol{\psi}_{p,\nu}(\xi_1) & \dots & \boldsymbol{\psi}_{p,\nu}(\xi_{t'}) \\ \boldsymbol{\psi}_{p,\nu}(\xi_1^2) & \dots & \boldsymbol{\psi}_{p,\nu}(\xi_{t'}^2) \\ \vdots & & \vdots \\ \boldsymbol{\psi}_{p,\nu}(\xi_1^{t-1}) & \dots & \boldsymbol{\psi}_{p,\nu}(\xi_{t'}^{t-1}) \end{pmatrix} .$$

We construct $\boldsymbol{x}^\intercal \in \mathbb{Z}_{p^e}^{1 \times s}$ such that

$$\boldsymbol{x}^\intercal \cdot T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{Z}_{p^e}^{1 \times t'} .$$

We do this by induction over e.

- *Base case:* $e = 1$. This case is a non-trivial extension to the proof of Theorem 4.4.12. Basically, T can be seen as a Vandermonde matrix with coefficients in the extension $\mathbb{F}_{p^\nu}$ of $\mathbb{F}_p$ and extended as vectors over the base field $\mathbb{F}_p$. Formally, let $P(X) \in \mathbb{F}_{p^\nu}[X]$ be the polynomial defined as follows:

  $$P(X) = \frac{(X - \xi_1) \cdots (X - \xi_{t'-1})}{(\xi_{t'} - \xi_1) \cdots (\xi_{t'} - \xi_{t'-1})} .$$

  The polynomial $P(X)$ has degree $t - 1$ and is such that:

  $$P(\xi_i) = \begin{cases} 0 & \text{if } i \in \{1, \dots, t' - 1\} , \\ 1 & \text{if } i = t' . \end{cases}$$

  Let $\boldsymbol{x'}^\intercal \in \mathbb{F}_{p^\nu}^{1 \times t}$ be the vector such that $P(X) = x'_1 + x'_2 \cdot X + \dots + x'_{t'} \cdot X^{t'-1}$. For any $i \in \{1, \dots, t'\}$, let $A_i \in \mathbb{Z}_p^{\nu \times \nu}$ be the matrix corresponding to the multiplication by x'_i in $\mathbb{F}_{p^\nu}$, i.e., such that for any $y \in \mathbb{F}_{p^\nu}$:

  $$\boldsymbol{\psi}_{p,\nu}(x'_i \cdot y) = A_i \cdot \boldsymbol{\psi}_{p,\nu}(y) .$$

  Finally, let $\boldsymbol{a}_i^\intercal \in \mathbb{Z}_p^{1 \times \nu}$ be the first row of A_i and let $\boldsymbol{x}^\intercal \in \mathbb{Z}_p^s$ be defined as follows:

  $$\boldsymbol{x}^\intercal = \begin{pmatrix} a_{1,1} & \boldsymbol{a}_2^\intercal & \boldsymbol{a}_3^\intercal & \dots & \boldsymbol{a}_{t'-1}^\intercal \end{pmatrix} .$$

  We remark that for any $\xi \in \mathbb{F}_{p^\nu}$ such that $P(\xi)$ is in the base field $\mathbb{F}_p$, we have:

  $$\boldsymbol{x}^\intercal \cdot \begin{pmatrix} 1 \\ \boldsymbol{\psi}_{p,\nu}(\xi) \\ \vdots \\ \boldsymbol{\psi}_{p,\nu}(\xi^{t-1}) \end{pmatrix} = P(\xi) .$$

  Therefore, we have:

  $$\boldsymbol{x}^\intercal \cdot T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} .$$

  That concludes the case $e = 1$.

- *Induction case:* $e \geq 2$. We use the induction hypothesis for $e-1$ and get a vector $\boldsymbol{x}'^{\intercal} \in \mathbb{Z}_{p^e}^{1\times s}$ such that:

$$\boldsymbol{x}'^{\intercal} \cdot T \bmod p^{e-1} = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix}$$

In other words, there exists a vector $\boldsymbol{y}' \in \mathbb{Z}^{1\times t'}$ such that:

$$\boldsymbol{x}'^{\intercal} \cdot T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} + p^{e-1} \cdot \boldsymbol{y}' \ .$$

Using a proof similar at what we did in the base case, using P the Lagrange polynomial of degree $t'-1$ satisfying $P(\xi_i) = -y_i$ modulo p for $i \in \{1, \dots, t'\}$, we get that there exists a vector $\boldsymbol{x}''^{\intercal}$ such that:

$$\boldsymbol{x}''^{\intercal} \cdot T \bmod p = -\boldsymbol{y}' \ .$$

We conclude by setting $\boldsymbol{x}^{\intercal} = \boldsymbol{x}'^{\intercal} + p^{e-1} \cdot \boldsymbol{x}''^{\intercal}$ and remarking that:

$$\boldsymbol{x}^{\intercal} \cdot T = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} + p^{e-1} \cdot \boldsymbol{y}' - p^{e-1} \cdot \boldsymbol{y}' = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} \ .$$

This concludes the proof. □

Let us now show a construction for a generic composite number $M = p_1^{e_1} \cdots p_r^{e_r}$. This is basically just a CRT combination of the functions $\boldsymbol{\phi}$ from Theorem 4.4.13 modulo $p_i^{e_i}$.

Theorem 4.4.14. *Let t and ν be positive integers. Let $M \geq 2$ be an integer. Let $M = p_1^{e_1} \cdots p_r^{e_r}$ be its prime decomposition. Let p be the smallest prime factor of M. Let $\mathsf{Tags} := \{0, \dots, p-1\}^{\nu}$. Let $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_r$ be the t-independent strongly non-degenerated functions defined in Theorem 4.4.13 modulo $p_1^{e_1}, \dots, p_r^{e_r}$ respectively. Let $\boldsymbol{\phi}$ be the function defined as follows:*

$$\boldsymbol{\phi} : \left(\begin{array}{ccc} \mathsf{Tags} & \to & \mathbb{Z}_M^{1+\nu\cdot(t-1)} \\ \mathsf{tag} & \mapsto & \mathsf{crt}(\boldsymbol{\phi}_1(\mathsf{tag}), \dots, \boldsymbol{\phi}_r(\mathsf{tag})) \end{array} \right) \ .$$

Then $\boldsymbol{\phi}$ is a t-independent strongly non-degenerated function modulo M from Tags to $\mathbb{Z}_M^s$, with $s = 1 + \nu \cdot (t-1)$.

Proof. The proof is straightforward using the CRT and actually works for as long as $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_r$ are t-independent strongly non-degenerated functions modulo $p_1^{e_1}, \dots, p_r^{e_r}$ respectively. □

4.4.3.2 Construction of t-Sound Tag-CS-DM and Tag-CS-DVS

From any CS-DM, we can now construct a t-sound tag-CS-DM corresponding to the same language. The resulting tag-CSDM is called a *t-sound extension* of the former CS-DM.

Construction 4.4.15 (t-sound extension)**.** *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a CS-DM. Let t be a positive integer. Let $\boldsymbol{\phi}$ be a t-independent strongly non-degenerated function from some set Tags to $\mathbb{Z}_M^s$. Let us construct a t-sound tag-CS-DM $\mathcal{M}' = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n', k', \mathsf{RGen}, \Gamma', \boldsymbol{\theta}', \boldsymbol{\lambda}')$, with p the smallest prime factor of M, as follows:*

$$n' := n \cdot s \qquad\qquad k' := k \cdot s$$

$$\Gamma'(\rho) := \mathsf{Id}_s \otimes \Gamma(\rho) = \begin{pmatrix} \Gamma & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \Gamma & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \Gamma \end{pmatrix}$$

$$\boldsymbol{\theta}'(\mathcal{x}, \rho, \mathsf{tag}) \coloneqq \boldsymbol{\phi}(\mathsf{tag}) \otimes \boldsymbol{\theta}(\mathcal{x}, \rho) = \begin{pmatrix} \phi_1(\mathsf{tag}) \bullet \boldsymbol{\theta}(\mathcal{x}, \rho) \\ \vdots \\ \phi_s(\mathsf{tag}) \bullet \boldsymbol{\theta}(\mathcal{x}, \rho) \end{pmatrix}$$

$$\boldsymbol{\lambda}'(\mathcal{x}, \mathcal{w}, \rho, \mathsf{tag}) \coloneqq \boldsymbol{\phi}(\mathsf{tag}) \otimes \boldsymbol{\lambda}(\mathcal{x}, \mathcal{w}, \rho) = \begin{pmatrix} \phi_1(\mathsf{tag}) \bullet \boldsymbol{\lambda}(\mathcal{x}, \mathcal{w}, \rho) \\ \vdots \\ \phi_s(\mathsf{tag}) \bullet \boldsymbol{\lambda}(\mathcal{x}, \mathcal{w}, \rho) \end{pmatrix} ,$$

where:

$$\boldsymbol{\phi}(\mathsf{tag}) = \begin{pmatrix} \phi_1(\mathsf{tag}) \\ \vdots \\ \phi_s(\mathsf{tag}) \end{pmatrix} \in \mathbb{Z}_M^s \ .$$

Let us now prove the security of the construction.

Proposition 4.4.16. *If $\mathcal{M}$ is an ε-$\mathscr{S}$-sound CS-DM (resp. 0-$\mathscr{S}$-sound KV-DM), then the DM $\mathcal{M}'$ defined in Construction 4.4.15 is an ε-t-$\mathscr{S}$-sound CS-DM (resp. 0-t-$\mathscr{S}$-sound KV-DM).*

Proof. As expected with the tensor products and the use of strong non-degeneracy, we will re-use the security proof of the disjunction of CS-DMs (Proposition 4.3.16).

Let $\mathcal{x}_1, \ldots, \mathcal{x}_{t-1} \in \mathcal{X}_{\mathsf{lpar}}$, $\mathcal{x}_t \in \mathscr{S}$, $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1} \in \mathsf{Tags}$, and $\mathsf{tag}_t \in \mathsf{Tags}$ distinct from $\mathsf{tag}_1, \ldots, \mathsf{tag}_{t-1}$. We suppose that ρ is such that $\boldsymbol{\theta}(\mathcal{x}, \rho) \notin \mathrm{ColSpan}\,(\Gamma(\rho))$. This happens with probability at least $1 - \varepsilon$.

We want to prove that:

$$\boldsymbol{\theta}'(\mathcal{x}_t, \rho, \mathsf{tag}_t) \notin \mathrm{ColSpan}\left(\begin{pmatrix} \Gamma'(\mathcal{x}, \rho) & \boldsymbol{\theta}'(\mathcal{x}_1, \rho, \mathsf{tag}_1) & \ldots & \boldsymbol{\theta}'(\mathcal{x}_{t-1}, \rho, \mathsf{tag}_{t-1}) \end{pmatrix}\right) \ .$$

Let $T \in \mathbb{Z}_M^{s \times (t-1)}$ be the matrix defined as:

$$T \coloneqq \begin{pmatrix} \boldsymbol{\phi}(\mathsf{tag}_1) & \ldots & \boldsymbol{\phi}(\mathsf{tag}_{t-1}) \end{pmatrix} \ .$$

We remark that for all $i \in \{1, \ldots, t-1\}$, if $\boldsymbol{e_i} \in \mathbb{Z}_M^{t-1}$ the i-th vector of the canonical basis of $\mathbb{Z}_M^{t-1}$:

$$\boldsymbol{\theta}'(\mathcal{x}_i, \rho, \mathsf{tag}_i) = \boldsymbol{\phi}(\mathcal{x}_i) \otimes \boldsymbol{\theta}(\mathcal{x}_i, \rho) = (T \otimes \mathsf{Id}_n) \bullet (\boldsymbol{e_i} \otimes \boldsymbol{\theta}(\mathcal{x}_i, \rho)) \in \mathrm{ColSpan}\,(T \otimes \mathsf{Id}_n) \ .$$

We therefore just need to prove that:

$$\boldsymbol{\theta}'(\mathcal{x}_t, \rho, \mathsf{tag}_t) \notin \mathrm{ColSpan}\left(\begin{pmatrix} \Gamma'(\mathcal{x}, \rho) & T \otimes \mathsf{Id}_n \end{pmatrix}\right) ,$$

or in other words:

$$\boldsymbol{\phi}(\mathsf{tag}_t) \otimes \boldsymbol{\theta}(\mathcal{x}_t) \notin \mathrm{ColSpan}\left(\begin{pmatrix} T \otimes \mathsf{Id}_n & \mathsf{Id}_s \otimes \Gamma(\mathcal{x}, \rho) \end{pmatrix}\right) \ .$$

This is similar to what we prove for the disjunction of two CS-DMs: one with $\Gamma_1 \coloneqq T$ and one with $\Gamma_2 = \Gamma$. Furthermore, by t-independent strong non-degeneracy of $\boldsymbol{\phi}$, we have that there exists a row vector $\boldsymbol{x}^\intercal \in \mathbb{Z}_M^{1 \times s}$ such that $\boldsymbol{x}^\intercal \cdot T = \mathbf{0}$ and $\boldsymbol{x}^\intercal \cdot \boldsymbol{\phi}(\mathsf{tag}_t) = 1$. In other words, informally, $\boldsymbol{\phi}(\mathsf{tag}_t)$ is strongly non-degenerated. We conclude exactly as in Proposition 4.3.16. $\square$

Remark 4.4.17. *For some applications, as NIZK, we use Construction 4.4.15 (to get t-soundness) in conjunction with Construction 4.3.15 (CS/KV disjunction). Let us show that these two operations "commute", up to a permutation of the columns of* Γ *and the coefficients of* $\boldsymbol{\theta}$*. This equivalence essentially comes from the associativity of tensor products. More precisely, let* $\mathcal{M}_1 = (M, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{M}_2 = (M, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs. The DVS corresponding to the disjunction of a t-sound extension of the DVS* $\mathcal{V}_1$ *and of the (unmodified) DVS* $\mathcal{V}_2$ *corresponds to the t-sound extension of the disjunction of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*: the dimensions* n *and* k *together with the vector* $\boldsymbol{\theta}$ *are the same, while the matrix* Γ *and the vector* $\boldsymbol{\lambda}$ *are equal up to permutation of the columns for* Γ *and of the entries for* $\boldsymbol{\lambda}$*. With notation Construction 4.4.15, the former DVS is indeed defined as follows:*

$$
\begin{aligned}
n &\coloneqq (s \cdot n_1) \cdot n_2 \qquad\qquad k \coloneqq (s \cdot k_1) \cdot n_2 + (s \cdot n_1) \cdot k_2 \\
\Gamma &\coloneqq \begin{pmatrix} (\mathsf{Id}_s \otimes \Gamma_1) \otimes \mathsf{Id}_{n_2} & (\mathsf{Id}_s \otimes \mathsf{Id}_{n_1}) \otimes \Gamma_2 \end{pmatrix} \\
&= \begin{pmatrix}
\Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathbf{0} & \dots & \mathbf{0} & \mathsf{Id}_{n_1} \otimes \Gamma_2 & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \Gamma_1 \otimes \mathsf{Id}_{n_2} & & \vdots & \mathbf{0} & \mathsf{Id}_{n_1} \otimes \Gamma_2 & & \vdots \\
\dots & & \ddots & \mathbf{0} & \dots & & \ddots & \mathbf{0} \\
\mathbf{0} & \dots & \mathbf{0} & \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathbf{0} & \dots & \mathbf{0} & \mathsf{Id}_{n_1} \otimes \Gamma_2
\end{pmatrix} \\
\boldsymbol{\theta} &\coloneqq (\phi(\mathsf{tag}) \otimes \boldsymbol{\theta}_1) \otimes \boldsymbol{\theta}_2 \\
\boldsymbol{\lambda} &\coloneqq \begin{cases} \begin{pmatrix} (\phi(\mathsf{tag}) \otimes \boldsymbol{\lambda}_1) \otimes \boldsymbol{\theta}_2 \\ \mathbf{0}_{sn_1k_2} \end{pmatrix} & \text{if } \mathscr{R}(\mathfrak{x}_1, \mathfrak{w}_1) = 1 \\ \begin{pmatrix} \mathbf{0}_{sk_1n_2} \\ (\phi(\mathsf{tag}) \otimes \boldsymbol{\theta}_1) \otimes \boldsymbol{\lambda}_2 \end{pmatrix} & \text{otherwise, if } \mathscr{R}(\mathfrak{x}_2, \mathfrak{w}_2) = 1 \end{cases},
\end{aligned}
$$

and the latter DVS is defined as follows:

$$
\begin{aligned}
n &\coloneqq s \cdot (n_1 \cdot n_2) \qquad\qquad k \coloneqq s \cdot (k_1 \cdot n_2 + n_1 \cdot k_2) \\
\Gamma &\coloneqq \mathsf{Id}_s \otimes \begin{pmatrix} \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2 \end{pmatrix} \\
&= \begin{pmatrix}
\Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2 & & \vdots \\
\vdots & \vdots & & \ddots & \ddots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \Gamma_1 \otimes \mathsf{Id}_{n_2} & \mathsf{Id}_{n_1} \otimes \Gamma_2
\end{pmatrix} \\
\boldsymbol{\theta} &\coloneqq (\phi(\mathsf{tag}) \otimes \boldsymbol{\theta}_1) \otimes \boldsymbol{\theta}_2 \\
\boldsymbol{\lambda} &\coloneqq \begin{cases} \phi(\mathsf{tag}) \otimes \begin{pmatrix} \boldsymbol{\lambda}_1 \otimes \boldsymbol{\theta}_2 \\ \mathbf{0}_{sn_1k_2} \end{pmatrix} & \text{if } \mathscr{R}(\mathfrak{x}_1, \mathfrak{w}_1) = 1 \\ \phi(\mathsf{tag}) \otimes \begin{pmatrix} \mathbf{0}_{sk_1n_2} \\ \boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2 \end{pmatrix} & \text{otherwise, if } \mathscr{R}(\mathfrak{x}_2, \mathfrak{w}_2) = 1 \end{cases}.
\end{aligned}
$$

Chapter 5

Pseudorandomness

This chapter introduces the notion of *pseudorandom projective hash functions* (PrPHFs) and *pseudorandom diverse vector spaces* (Pr-DVSs), together with the notion of mixed pseudorandomness, used to improve the efficiency of some of our applications. We only work with prime-order *diverse modules* (DMs), or in other words, with *diverse vector spaces* (DVSs), as this is quite technical and already gives many interesting results.

A PrPHF is a PHF for which the hash value of a random word inside the language looks uniform for an adversary who only knows the projection key (and but not a witness for this word). A Pr-DVS is just a KV-DVS for which the associated PHF is pseudorandom. Any SPHF or DVS for a hard-subset-membership language is a PrPHF or Pr-DVS. But there exist more efficient constructions of PrPHFs and Pr-DVSs, in particular constructions for which the associated language is trivial and the dimensions of the corresponding vector spaces ($\hat{\mathscr{L}}$ and $\hat{\mathscr{X}}$) are equal to 1. For a hard-subset-membership language, this is impossible as $\hat{\mathscr{L}}$ has to be a proper non-zero subspace of $\hat{\mathscr{X}}$.

In our applications like *non-interactive zero-knowledge arguments* (NIZK) and *implicit zero-knowledge arguments* (iZK) (Sections 3.3, 6.2 and 6.3), we do not use directly a DVS over a hard-subset-membership language, but we use the disjunction of such a DVS $\mathcal{V}_2$ with a DVS $\mathcal{V}_1$ corresponding to the language we are interested in. When $\mathcal{V}_2$ is replaced a Pr-DVS for a potentially trivial language, mixed pseudorandomness corresponds to the property we need to prove the security of the resulting iZK or NIZK.

Contents

5.1 Pseudorandom Projective Hash Functions and Diverse Vector Spaces 124

5.1.1 Definition 124

5.1.2 Construction from Hard-Subset-Membership Languages 124

5.1.3 Construction from MDDH 126

5.2 Mixed Pseudorandomness 129

5.2.1 Definition 129

5.2.2 GL Disjunctions of a GL-DVS and a Pr-DVS 130

5.2.3 CS/KV Disjunctions of a DVS and a Pr-DVS 133

5.1 Pseudorandom Projective Hash Functions (PrPHFs) and Pseudorandom Diverse Vector Spaces (Pr-DVSs)

Let us start by defining and constructing PrPHFs and Pr-DVSs. We provide two constructions: one completely generic from any KV-DVS associated to a hard-subset-membership language, and one based on *matrix decisional Diffie-Hellman* (MDDH) assumptions (recalled in Section 3.4.1) which is not generic but is more efficient.

5.1.1 Definition

Definition 5.1.1. *A PHF* (HashKG, ProjKG, Hash, ProjHash) *for a language* $\mathscr{L} \subseteq \mathcal{X}$ *is a* pseudorandom projective hash function *(PrPHF) if it satisfies the following properties:*

- $\mathscr{R}$-samplability of $\mathscr{L}$. *As in Definition 2.3.3;*
- KV/CS type. *Projections keys do not depend on the word to be hashed, i.e.,* ProjKG *does not use its input* χ*;*
- Pseudorandomness. *The hash value of a random word* $\chi \xleftarrow{\$} \mathscr{L}$ *is computationally indistinguishable from a random value from the range* Π *of the PHF. More formally, the advantage of an adversary* $\mathcal{A}$ *against pseudorandomness is defined by the experiments* $\mathsf{Exp}^{psrnd\text{-}b}$ *depicted in Figure 5.1. The PHF is pseudorandom if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*.*

We remark that in the pseudorandomness experiment, contrary to most of the experiments in this thesis but similarly to the experiments $\mathsf{Exp}^{\texttt{sub-memb-}b}$ for hard subset membership in Figure 2.3, the adversary is not given access the trapdoor ltrap for the language.

A *pseudorandom diverse vector space* (Pr-DVS) is a KV-DVS for which the corresponding PHF is a PrPHF.

The restrictions to KV-DVSs and to CS-type PHFs make everything simpler and we do not know of any application for which lifting these restrictions would provide any benefit. That is why we consider only this restricted setting.

5.1.2 Construction from Hard-Subset-Membership Languages

A KV-SPHF for a hard-subset-membership language is actually a Pr-DVS. More formally, we have the following proposition.

Proposition 5.1.2. *Let PHF* (HashKG, ProjKG, Hash, ProjHash) *be an* ε*-smooth KV-SPHF over a hard-subset-membership language* $\mathscr{L}$*. Then this PHF is a PrPHF. More precisely, for any adversary* $\mathcal{A}$ *against pseudorandomness of the associated PHF, there exists an adversary* $\mathcal{B}$ *against subset membership of* $\mathscr{L}$ *with similar running time*[1] *such that:*

$$\mathsf{Adv}^{psrnd}(\mathcal{A}, \mathfrak{K}) \leq 2 \cdot \mathsf{Adv}^{sub\text{-}memb}(\mathcal{B}, \mathfrak{K}) + \varepsilon \ .$$

In particular, a KV-DVS for a hard-subset-membership language $\mathscr{L}$ is a Pr-DVS.

Proof. We do a proof by games:

[1] See Footnote 7 on page 32.

Pseudorandomness

$\mathsf{Exp}^{\mathtt{psrnd}\text{-}b}(\mathcal{A}, \mathfrak{K})$
  $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
  $(\mathsf{lpar}, \mathsf{ltrap}) \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$
  $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$
  $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$
  $\chi \xleftarrow{\$} \mathscr{L}_{\mathsf{lpar}}$
  if $b = 1$ **then**
    $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$
  else
    $\mathsf{H} \xleftarrow{\$} \Pi$
  return $\mathcal{A}(\mathsf{lpar}, \chi, \mathsf{hp}, \mathsf{H})$

Figure 5.1: Experiments for Definition 5.1.1 (pseudorandomness)

Game $\mathbf{G}_0$: This game corresponds to $\mathsf{Exp}^{\mathtt{psrnd}\text{-}1}$.

Game $\mathbf{G}_1$: In this game, we sample χ from $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ instead of from $\mathscr{L}_{\mathsf{lpar}}$. This game is computationally indistinguishable from the previous one under hard-subset-membership. More formally, we can construct an adversary $\mathcal{B}_1$ with running time similar to $\mathcal{A}$ such that:

$$|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_1 = 1]| \leq \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_1, \mathfrak{K}) \ .$$

Game $\mathbf{G}_2$: In this game, we sample H at random from Π instead of computing it as the hash value of χ under hk. This game is statistically indistinguishable from the previous one thanks to smoothness:

$$|\Pr[\mathbf{G}_1 = 1] - \Pr[\mathbf{G}_2 = 1]| \leq \varepsilon \ .$$

Game $\mathbf{G}_3$: In this game, we sample χ from $\mathscr{L}_{\mathsf{lpar}}$ again instead of from $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$. This game is computationally indistinguishable from the previous one under hard-subset-membership. More formally, we can construct an adversary $\mathcal{B}_2$ with running time similar to $\mathcal{A}$ such that:

$$|\Pr[\mathbf{G}_2 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_2, \mathfrak{K}) \ .$$

Furthermore, this game exactly corresponds to $\mathsf{Exp}^{\mathtt{psrnd}\text{-}0}$.

We therefore get:

$$\mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{A}, \mathfrak{K}) = |\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_1, \mathfrak{K}) + \varepsilon + \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_2, \mathfrak{K}) \ .$$

We get the statement of the proposition by constructing an adversary $\mathcal{B}$ that draws a uniform random integer $i \xleftarrow{\$} \{1, 2\}$ and runs $\mathcal{B}_i$. □

5.1.3 Construction from MDDH

As already explained, the previous construction works fine but is not optimal as a KV-DVS $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ for a hard-subset-membership language $\mathscr{L}$ has to be such that $\hat{\mathscr{L}}$ is a proper non-zero subspace of $\hat{\mathcal{X}}$. This implies that $n > k \geq 1$ and $n \geq 2$.

Let us show a construction of Pr-DVS which does not have this limitation.

Construction 5.1.3. *Let $\mathcal{D}$ be a distribution of matrices in $\mathbb{Z}_p^{(k+1)\times k}$ samplable in polynomial time. Let $\mathfrak{G}$ be a graded ring of prime order p. Let $\tilde{\boldsymbol{v}}$ be some index for this graded ring. For pseudorandomness, we suppose that the $\mathcal{D}$-MDDH assumption (recalled in Section 3.4.1) holds in $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$.*

We define a Pr-DVS as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ *generates the graded ring* $\mathsf{gpar} := \mathfrak{G}$ *for the security parameter* $\mathfrak{K}$*;*
- $\mathsf{Setup.lpar}(\mathsf{gpar})$ *generates and random matrix* $A \xleftarrow{\$} \mathcal{D}$*, splits it in two matrices* $\bar{A} \in \mathbb{Z}_p^{k\times k}$ *and* $\underline{A} \in \mathbb{Z}_p^{1\times k}$*, such that:*
$$A = \begin{pmatrix} \bar{A} \\ \underline{A} \end{pmatrix} \ .$$
and outputs $\mathsf{lpar} := [\bar{A}] := [\tilde{\boldsymbol{v}}, \bar{A}] \in \mathfrak{G}_{\tilde{\boldsymbol{v}}}^{n\times k}$ *and* $\mathsf{ltrap} = A$*;*
- *the matrix* Γ_{lpar} *is exactly the matrix* $[\bar{A}] = \mathsf{lpar}$*;*
- *the functions* θ *and* $\boldsymbol{\lambda}$ *are the identity functions;*
- *the language is defined by:*
$$\mathscr{L} := \hat{\mathscr{L}} := \{[\boldsymbol{u}] \mid \exists \boldsymbol{r} \in \mathbb{Z}_p^k,\ \boldsymbol{u} = \bar{A} \cdot \boldsymbol{r}\} \quad \subseteq \quad \mathbb{G}^n =: \hat{\mathcal{X}} =: \mathcal{X} \ .$$

Concretely, the graded ring $\mathfrak{G}$ is often a cyclic group $(p, \mathbb{G}, g)$ and the set $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$ is the group $\mathbb{G}$. In this case, we recall that for any matrix $B \in \mathbb{Z}_p^{\ell\times m}$, the matrix $C = [B] \in \mathbb{G}^{\ell\times m}$ is defined by $C_{i,j} = g^{B_{i,j}}$.

We remark that when the matrices $A \xleftarrow{\$} \mathcal{D}$ are full-rank, then $\mathscr{L} = \mathcal{X}$ and the language is trivial. In this case, the above KV-DVS would be perfectly smooth, as $\mathcal{X} \setminus \mathscr{L}$ is empty. But smoothness is completely vacuous, in this case. Furthermore, the language is clearly not hard-subset-membership. Let us still show that the above KV-DVS is a Pr-DVS.

Proposition 5.1.4. *Let $\mathcal{D}$ be a distribution of matrices in $\mathbb{Z}_p^{(k+1)\times k}$ samplable in polynomial time. Let $\mathfrak{G}$ be a graded ring of prime order p. Let $\tilde{\boldsymbol{v}}$ be some index for this graded ring. We suppose that for any matrix $A \xleftarrow{\$} \mathcal{D}$, $\bar{A}$ is invertible, except with negligible probability ε. If $\mathcal{D}$-MDDH holds in $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$, then the KV-DVS defined in Construction 5.1.3 is a Pr-DVS. More precisely, for any adversary $\mathcal{A}$ against pseudorandomness of the associated PHF, there exists an adversary $\mathcal{B}$ against the $\mathcal{D}$-MDDH assumption with similar running time[2] such that:*

$$\mathsf{Adv}^{psrnd}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{mddh}(\mathcal{B}, \mathfrak{K}) + 4\varepsilon + 2/p \ .$$

We remark that the condition that $\bar{A}$ is invertible with overwhelming probability is satisfied by all the standard MDDH assumptions, including DDH, DLin, the κ-linear assumption [HK07; Sha07], the cascade assumption, and the symmetric cascade assumption [EHK+13].

[2] See Footnote 7 on page 32.

Proof. We do a proof by games.

Game $\mathbf{G}_0$: This game corresponds to $\mathsf{Exp}^{\mathtt{psrnd\text{-}1}}$, except that we compute the hash value H as pH using a witness w (which does not change anything by perfect correctness of the DVS). We have:

$$w := \boldsymbol{\lambda} \xleftarrow{\$} \mathbb{Z}_p^k \qquad\qquad \chi := \boldsymbol{\theta} := [\bar{A}] \bullet \boldsymbol{\lambda}$$
$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times k} \qquad \mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet [\bar{A}] \qquad \mathsf{H} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda} \ .$$

Game $\mathbf{G}_1$: In this game, we abort (and return 0) if $\bar{A}$ is not invertible. This game is statistically indistinguishable from the previous one:

$$|\Pr[\,\mathbf{G}_0 = 1\,] - \Pr[\,\mathbf{G}_1 = 1\,]| \le \varepsilon \ .$$

Game $\mathbf{G}_2$: In this game, we generate an additional uniform random value $\alpha_{k+1} \xleftarrow{\$} \mathbb{Z}_p$ and compute hp as follows:

$$\mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet [\bar{A}] \ \boldsymbol{+} \ \alpha_{k+1} \bullet [\underline{A}] = \boldsymbol{\alpha}'^\intercal \bullet [A] \quad \text{where } \boldsymbol{\alpha}'^\intercal := \begin{pmatrix} \boldsymbol{\alpha}^\intercal & \alpha_{k+1} \end{pmatrix} \ .$$

As $\bar{A}$ is full-rank, in this game and the previous game, hp has the same distribution: it is uniform over $\mathfrak{G}_{\tilde{\boldsymbol{v}}}^{1\times k}$. Therefore, this game is perfectly indistinguishable from the previous one:

$$\Pr[\,\mathbf{G}_1 = 1\,] = \Pr[\,\mathbf{G}_2 = 1\,] \ .$$

Game $\mathbf{G}_3$: In this game, we do not abort anymore when $\bar{A}$ is not invertible. This game is statistically indistinguishable from the previous one:

$$|\Pr[\,\mathbf{G}_2 = 1\,] - \Pr[\,\mathbf{G}_3 = 1\,]| \le \varepsilon \ .$$

Game $\mathbf{G}_4$: In this game, we change the way H and $\boldsymbol{\theta}$ are computed and defined:

$$\boldsymbol{\theta}' := [A] \bullet \boldsymbol{\lambda} \in \mathfrak{G}_{\tilde{\boldsymbol{v}}}^{k+1} \qquad \boldsymbol{\theta} := (\theta'_i)_{i=1,\dots,k} \qquad \mathsf{H} := \boldsymbol{\alpha}'^\intercal \bullet \boldsymbol{\theta}' \ .$$

Concretely, this means that:

$$\boldsymbol{\theta}' = \begin{pmatrix} \boldsymbol{\theta} \\ \theta'_{k+1} \end{pmatrix} \ .$$

Furthermore $w = \boldsymbol{\lambda}$ is no more used in this game. This game is perfectly indistinguishable from the previous game:

$$\Pr[\,\mathbf{G}_3 = 1\,] = \Pr[\,\mathbf{G}_4 = 1\,] \ .$$

Game $\mathbf{G}_5$: Let $\mathscr{L}' \subseteq \mathcal{X}'$ be the canonical language corresponding to the $\mathcal{D}$-MDDH assumption, as in Section 3.4.1:

$$\mathscr{L}' := \{[\boldsymbol{u}] \mid \exists \boldsymbol{r} \in \mathbb{Z}_p^k,\ \boldsymbol{u} = A \cdot \boldsymbol{r}\} \quad \subseteq \quad \mathfrak{G}_{\tilde{\boldsymbol{v}}}^{k+1} =: \mathcal{X}' \ .$$

In this game, we now generate $\boldsymbol{\theta}'$ as a uniform vector in $\mathcal{X} \setminus \mathscr{L}$. We have:

$$\boldsymbol{\theta}' \xleftarrow{\$} \mathcal{X}' \setminus \mathscr{L}' \qquad \chi := \boldsymbol{\theta} := (\theta'_i)_{i=1,\dots,k} \qquad \boldsymbol{\alpha}'^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times(k+1)}$$
$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal := (\alpha'_i)_{i=1,\dots,k}^\intercal \qquad \mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}'^\intercal \bullet [A] \qquad \mathsf{H} := \boldsymbol{\alpha}'^\intercal \bullet \boldsymbol{\theta}' \ .$$

This game is computationally indistinguishable from the previous one under the $\mathcal{D}$-MDDH assumption. More formally, we can construct an adversary $\mathcal{B}$ with running time similar to $\mathcal{A}$ such that:

$$|\Pr[\mathbf{G}_4 = 1] - \Pr[\mathbf{G}_5 = 1]| \leq \mathsf{Adv}^{\mathsf{mddh}}(\mathcal{B}, \mathfrak{K}) + 1/p \ .$$

The additive term $1/p$ comes from the fact that the probability that a uniform vector $[\boldsymbol{u}] \in \mathfrak{G}^{k+1}$ is in $\mathscr{L}'$ is at most $1/p$.

Game $\mathbf{G}_6$: In this game, we replace H by a random value in $\mathfrak{G}_{\tilde{\boldsymbol{v}}}$. This game is statistically indistinguishable from the previous one thanks to the smoothness of the SPHF with hashing key $\boldsymbol{\alpha}'$, projection key $\boldsymbol{\gamma}$, and hash value H, for the MDDH language $\mathscr{L}' \subseteq \mathcal{X}'$ as defined in Section 3.4.1:

$$|\Pr[\mathbf{G}_5 = 1] - \Pr[\mathbf{G}_6 = 1]| \leq 1/p \ .$$

Game $\mathbf{G}_7$: We remark that in the previous game θ'_{k+1} is not used anymore (as it was only used in the computation of H) and $\boldsymbol{\theta}$ is just a random vector, so we compute everything as follows:

$$\begin{array}{lll} \chi := \boldsymbol{\theta} \xleftarrow{\$} \mathfrak{G}_{\tilde{\boldsymbol{v}}}^k & & \boldsymbol{\alpha}'^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times(k+1)} \\ \mathsf{hk} := \boldsymbol{\alpha}^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times k} & \mathsf{hp} := \boldsymbol{\gamma}^{\mathsf{T}} := \boldsymbol{\alpha}'^{\mathsf{T}} \bullet [A] & \mathsf{H} \xleftarrow{\$} \mathfrak{G}_{\tilde{\boldsymbol{v}}} \ . \end{array}$$

This game is perfectly indistinguishable from the previous game and:

$$\Pr[\mathbf{G}_6 = 1] = \Pr[\mathbf{G}_7 = 1] \ .$$

Game $\mathbf{G}_8$: This game exactly is $\mathsf{Exp}^{\mathtt{psrnd}\text{-}0}$. Compared to the previous game, we just changed

$$\begin{cases} \chi := \boldsymbol{\theta} \xleftarrow{\$} \mathfrak{G}_{\tilde{\boldsymbol{v}}}^k \\ \boldsymbol{\alpha}'^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times(k+1)} \\ \mathsf{hp} := \boldsymbol{\gamma}^{\mathsf{T}} := \boldsymbol{\alpha}'^{\mathsf{T}} \bullet [A] \end{cases} \quad \text{to} \quad \begin{cases} w := \boldsymbol{\lambda} \xleftarrow{\$} \mathbb{Z}_p^k \\ \chi := \boldsymbol{\theta} := [\bar{A}] \bullet \boldsymbol{\lambda} \\ \boldsymbol{\alpha}^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times k} \\ \mathsf{hp} := \boldsymbol{\gamma}^{\mathsf{T}} := \boldsymbol{\alpha}^{\mathsf{T}} \bullet [\bar{A}] \end{cases} .$$

When $\bar{A}$ is invertible, the distribution of (χ, hp) is exactly the same in both cases. Therefore, we can show that this game is indistinguishable from the previous one:

$$|\Pr[\mathbf{G}_7 = 1] - \Pr[\mathbf{G}_8 = 1]| \leq \varepsilon \ .$$

We therefore get:

$$\mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{A}, \mathfrak{K}) = |\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_3 = 1]| \leq \mathsf{Adv}^{\mathsf{mddh}}(\mathcal{B}, \mathfrak{K}) + 4\varepsilon + 2/p \ .$$

That concludes the proof. □

Let us show a simple concrete example

Example 5.1.5 (Pr-DVS under DDH)**.** *The global parameters are* $\mathsf{gpar} := (p, \mathbb{G}, g)$*, with* $\mathbb{G}$ *being a cyclic group of prime order* p *generated by* g*. There are no language parameters (*$\mathsf{lpar} = \perp$*). The trivial language is defined as follows:*

$$\mathscr{L} = \{u \in \mathbb{G} \mid \exists r \in \mathbb{Z}_p,\ u = g^r\} \quad \subsetneq \quad \mathbb{G} = \mathcal{X} \ .$$

The DVS is defined as follows:

$$n := 1 \qquad\qquad k := 1$$
$$\Gamma := \begin{pmatrix} g \end{pmatrix}$$
$$\boldsymbol{\theta}(\chi) := \begin{pmatrix} u \end{pmatrix} \qquad\qquad \boldsymbol{\lambda}(\chi, w) := \begin{pmatrix} r \end{pmatrix} \ ,$$

where

$$\chi = u \qquad\qquad w = r \ .$$

Historical note 5.1.6. *While pseudorandomness was introduced by Gennaro and Lindell in [GL03; GL06], it was always a consequence of the hard-subset-membership property. The first constructions with trivial language were proposed in [ABP15c] and were based on the* κ*-linear assumption [HK07; Sha07] which is a generalization of DDH and DLin. Construction 5.1.3 is even more general and is inspired from [KW15]. When the distribution* $\mathcal{D}$ *corresponds to the* κ*-linear assumption, we do not exactly get the construction of [ABP15c] though, as in the latter construction, there was only one possible word in the trivial language. But this difference has no impact in our applications.*

5.2 Mixed Pseudorandomness

5.2.1 Definition

Definition 5.2.1. *Let* $\mathscr{L}$ *be a language corresponding to the disjunction of two languages* $\mathscr{L}_1$ *and* $\mathscr{L}_2$ *as defined in Section 3.2.2. A PHF* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *is* mixed-pseudorandom *if the hash value of a word* (χ_1, χ_2) *is computationally indistinguishable from a random value, when* χ_1 *is chosen by the adversary in* $\mathcal{X}_1 \setminus \mathscr{L}_1$*, while* χ_2 *is chosen at random in* $\mathscr{L}_2$*. When* χ_1 *is chosen before the projection key, the PHF is said to be* GL/CS-mixed-pseudorandom *(CS when* ProjKG *does not use its input* χ *and GL otherwise). Otherwise, the PHF is said to be* KV-mixed-pseudorandom *(in which case* ProjKG *should not use its input* χ*). The advantages of an adversary* $\mathcal{A}$ *against GL/CS and KV pseudorandomness are defined by the experiments* $\mathsf{Exp}^{\mathtt{gl/cs\text{-}m\text{-}psrnd\text{-}b}}$ *and* $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd\text{-}b}}$ *respectively, both depicted in Figure 5.2. The PHF is* mixed-pseudorandom *if the corresponding advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*.*

A GL-DVS (resp. CS-DVS and KV-DVS) for a disjunction $\mathscr{L}$ of two languages $\mathscr{L}_1$ and $\mathscr{L}_2$ is said to be *GL (resp. CS and KV)-mixed-pseudorandom* if its associated PHF is *GL (resp. CS and KV)-mixed-pseudorandom.*

We remark that in the experiments we explicitly defined $\mathsf{ltrap} = (\mathsf{ltrap}_1, \mathsf{ltrap}_2)$, as our proof of mixed pseudorandomness for CS/KV disjunctions actually requires ltrap_1 to contain enough information so that for any ρ, it is possible to compute the discrete logarithms of the entries of $\Gamma_1(\rho)$. We use a strong security notion in which the adversary is also given access to ltrap_1. This is convenient, as often ltrap_1 contains a decryption key. However, as in pseudorandomness, we do not give access to ltrap_2 to the adversary.

Mixed Pseudorandomness

$\mathsf{Exp}^{\texttt{gl/cs-m-psrnd-}b}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Exp}^{\texttt{kv-m-psrnd-}b}(\mathcal{A}, \mathfrak{K})$

$\quad \mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$

$\quad (\mathsf{lpar} = (\mathsf{lpar}_1, \mathsf{lpar}_2), \mathsf{ltrap} = (\mathsf{ltrap}_1, \mathsf{ltrap}_2)) \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$

$\quad \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{lpar})$

$\quad \mathsf{hp} \leftarrow \perp$ ▷ only in $\mathsf{Exp}^{\texttt{gl/cs-m-psrnd-}b}$

$\quad \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$ ▷ only in $\mathsf{Exp}^{\texttt{kv-m-psrnd-}b}$

$\quad x_2 \xleftarrow{\$} \mathscr{L}_{2,\mathsf{lpar}_2}$

$\quad (x_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{lpar}, \mathsf{ltrap}_1, \mathsf{hp}, x_2)$

$\quad x \leftarrow (x_1, x_2)$

$\quad \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}, x)$ ▷ only in $\mathsf{Exp}^{\texttt{gl/cs-m-psrnd-}b}$

$\quad$ **if** $b = 1$ or $x_1 \in \mathscr{L}_{1,\mathsf{lpar}_1}$ **then**

$\quad\quad \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, x)$

$\quad$ **else**

$\quad\quad \mathsf{H} \xleftarrow{\$} \Pi$

$\quad$ **return** $\mathcal{A}(\mathsf{st}, \mathsf{hp}, \mathsf{H})$

Figure 5.2: Experiments for Definition 5.2.1 (GL/CS and KV mixed pseudorandomness)

5.2.2 GL Disjunctions of a GL-DVS and a Pr-DVS

Let us show that the GL disjunction of a GL-DVS $\mathcal{V}_1$ and a Pr-DVS $\mathcal{V}_2$ is (GL) mixed-pseudorandom, when the trapdoor ltrap_1 output by $\mathsf{Setup.lpar}_1$ enables to check whether a word $x_1 \in \mathcal{X}_1$ is in $\mathscr{L}_1$ in polynomial time.

Theorem 5.2.2. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs. Let* $\mathcal{V}$ *be the disjunction GL-DVS of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$ *as defined in Construction 3.2.4. We suppose that the trapdoor* ltrap_1 *output by* $\mathsf{Setup.lpar}_1$ *enables to check whether a word* $x_1 \in \mathcal{X}_1$ *is in* $\mathscr{L}_1$ *in polynomial time.*

If $\mathcal{V}_2$ *is a Pr-DVS, then* $\mathcal{V}$ *is mixed-pseudorandom. More precisely, for any adversary* $\mathcal{A}$ *against mixed pseudorandomness of the PHF associated to* $\mathcal{V}$*, there exists any adversary* $\mathcal{B}$ *against pseudorandomness of the PHF associated to* $\mathcal{V}_2$ *with similar running time, such that:*

$$\mathsf{Adv}^{\mathit{gl/cs\text{-}m\text{-}psrnd}}(\mathcal{A}, \mathfrak{K}) \leq 2 \cdot \mathsf{Adv}^{\mathit{psrnd}}(\mathcal{B}, \mathfrak{K}) + \varepsilon\,,$$

assuming $\mathcal{V}_1$ *is* ε*-sound.*

Proof. We first remark that the condition on the trapdoor ltrap_1 makes the experiment polynomial time.

We recall the definition of Γ and $\boldsymbol{\theta}$ in Construction 3.2.4:

$$\Gamma(x, \rho) := \begin{pmatrix} \mathbf{0}_{k_1}^\intercal & 1_{\mathbb{Z}_p} & \mathbf{0}_{k_2}^\intercal & 1_{\mathbb{Z}_p} \\ \Gamma_1 & \boldsymbol{\theta}_1 & 0_{n_1 \times k_2} & \mathbf{0}_{n_1} \\ 0_{n_2 \times k_1} & \mathbf{0}_{n_2} & \Gamma_2 & \boldsymbol{\theta}_2 \end{pmatrix} \qquad \boldsymbol{\theta}(x, \rho) := \begin{pmatrix} [\top, -1] \\ [\top, 0] \\ \vdots \\ [\top, 0] \end{pmatrix} \in \mathfrak{G}_\top^n\ .$$

We do a proof by games.

Game $\mathbf{G}_0$: This game corresponds to $\mathsf{Exp}^{\texttt{gl/cs-m-psrnd-1}}$. We recall that:

$$\begin{aligned} \boldsymbol{\alpha}^\intercal &\xleftarrow{\$} \mathbb{Z}_p^{1\times n} & \mathsf{hk} &= (\boldsymbol{\alpha}^\intercal, \rho) \\ \boldsymbol{\gamma}^\intercal &:= \boldsymbol{\alpha}^\intercal \bullet \Gamma & \mathsf{hp} &= (\boldsymbol{\gamma}^\intercal, \rho) \\ \mathsf{H} &= \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} = \alpha_1 \ . \end{aligned}$$

In the last equality as in the whole proof, we are quite liberal regarding the indexes of the graded ring elements.

Game $\mathbf{G}_1$: This game is the same as the previous one, except that γ_k is replaced by a uniform random value (independent of everything else). This game is computationally indistinguishable form the previous one, under pseudorandomness of $\mathcal{V}_2$, thanks to Lemma 5.2.3 (below). We remark that we can also define this game as follows:

$$\begin{aligned} \Gamma' &:= \begin{pmatrix} \mathbf{0}_{k_1}^\intercal & 1_{\mathbb{Z}_p} & \mathbf{0}_{k_2}^\intercal & 1_{\mathbb{Z}_p} \\ \Gamma_1 & \boldsymbol{\theta}_1 & 0_{n_1\times k_2} & \mathbf{0}_{n_1} \\ 0_{n_2\times k_1} & \mathbf{0}_{n_2} & \Gamma_2 & \mathbf{0}_{n_2} \\ \mathbf{0}_{k_1}^\intercal & 0 & \mathbf{0}_{k_2}^\intercal & 1_{\mathbb{Z}_p} \end{pmatrix} \\ \boldsymbol{\alpha}^\intercal &\xleftarrow{\$} \mathbb{Z}_p^{1\times n} & \alpha'_n &\xleftarrow{\$} \mathbb{Z}_p \\ \boldsymbol{\alpha}'^\intercal &:= \begin{pmatrix} \boldsymbol{\alpha}^\intercal & \alpha'_n \end{pmatrix} \\ \boldsymbol{\theta}' &:= \begin{pmatrix} [\top, -1] \\ [\top, 0] \\ \vdots \\ [\top, 0] \end{pmatrix} \in \mathfrak{G}_\top^{n+1} \\ \boldsymbol{\gamma}^\intercal &:= \boldsymbol{\alpha}'^\intercal \bullet \Gamma' \\ \mathsf{H} &:= \boldsymbol{\alpha}'^\intercal \bullet \boldsymbol{\theta}' \ . \end{aligned}$$

Compared to the matrix Γ, the matrix Γ' has an additional row and its last column is slightly different. The other entries are the same.

Game $\mathbf{G}_2$: In this game, we replace H by a uniform random element in $\mathbb{G}$ independent of everything else. Thanks to the above description of the previous game, when $\boldsymbol{\theta}_1$ is linearly independent from the columns of Γ_1, $\boldsymbol{\theta}'$ is linearly independent of the columns of Γ'. As this happens with probability as least $1-\varepsilon$ by ε-soundness of $\mathcal{V}_1$, we get that:

$$|\Pr[\,\mathbf{G}_1 = 1\,] - \Pr[\,\mathbf{G}_2 = 1\,]| \leq \varepsilon\,,$$

by a proof similar to the smoothness proof of a PHF associated to a DVS (Theorem 3.1.11).

Game $\mathbf{G}_3$: In this game, we compute again γ_k as in the first game. This game is computationally indistinguishable form the previous one, under pseudorandomness of $\mathcal{V}_2$, thanks to Lemma 5.2.4 (below). Furthermore, it corresponds to $\mathsf{Exp}^{\texttt{gl/cs-m-psrnd-1}}$.

Lemma 5.2.3. *For any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}_1$ against pseudorandomness of $\mathcal{V}_2$ with a similar running time, such that:*

$$|\Pr[\,\boldsymbol{G}_0 = 1\,] - \Pr[\,\boldsymbol{G}_1 = 1\,]| \leq \mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{B}_1, \mathfrak{K}) \ .$$

Proof. Let us show how to construct an adversary $\mathcal{B}_1$ against pseudorandomness of the PHF associated to $\mathcal{V}_2$ with similar running time such that

$$|\Pr[\,\mathbf{G}_0 = 1\,] - \Pr[\,\mathbf{G}_{G:glpsrnd-2} = 1\,]| \leq \mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{B}_1, \mathfrak{K}) \ .$$

The adversary $\mathcal{B}_1$ receives gpar, lpar_2, $\mathfrak{x}_2$, $\mathsf{hp}_2 = \boldsymbol{\gamma}_2^\intercal = \boldsymbol{\alpha}_2^\intercal \bullet \Gamma_2$ (where $\boldsymbol{\alpha}_2^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n_2}$), and $\mathsf{H}_2 = \boldsymbol{\alpha}_2^\intercal \bullet \boldsymbol{\theta}_2(\mathfrak{x}_2)$ in the case $b = 1$ or $\mathsf{H}_2 \xleftarrow{\$} \mathfrak{G}$ in the case $b = 0$. Next, it generates $(\mathsf{lpar}_1, \mathsf{ltrap}_1) \xleftarrow{\$} \mathsf{Setup.lpar}(\mathsf{gpar})$ and runs $\mathcal{A}(\mathsf{lpar}, \mathsf{ltrap}_1, \perp, \mathfrak{x}_2)$ and gets back $(\mathfrak{x}_1, \mathsf{st})$. It then computes the following elements:

$$\begin{aligned}
&\rho_1 \xleftarrow{\$} \mathsf{RGen}_1(\mathsf{lpar}_1)\\
&\boldsymbol{\alpha}_1^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times(k_1+1)}\\
&\boldsymbol{\gamma}^\intercal := \begin{pmatrix} \boldsymbol{\alpha}_1^\intercal \bullet \Gamma' & \boldsymbol{\gamma}_2^\intercal & (\alpha_{1,1} + \mathsf{H}_2) \end{pmatrix} \in \mathfrak{G}^{1\times(2+k_1+k+2)}\\
&\qquad \text{where } \Gamma' := \begin{pmatrix} \mathbf{0}_{k_1}^\intercal & 1_{\mathbb{Z}_p} \\ \Gamma_1(\mathfrak{x}_1, \rho_1) & \boldsymbol{\theta}_1(\mathfrak{x}_1, \rho_1) \end{pmatrix} \in \mathfrak{G}^{(k_1+1)\times(n_1+1)} \text{ is the upper-left part of } \Gamma,\\
&\mathsf{H} := \alpha_{1,1} \ .
\end{aligned}$$

Then $\mathcal{B}_1$ runs $\mathcal{A}$ on inputs $\mathcal{A}(\mathsf{st}, \mathsf{hp}, \mathfrak{x})$ (where $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2)$) and outputs what $\mathcal{A}$ outputs. Let us prove that $\mathcal{B}_1$ exactly simulates Game $\mathbf{G}_0$ for $\mathcal{A}$ when $\mathcal{B}_1$ is run in $\mathsf{Exp}^{\mathtt{psrnd}\text{-}b}$ with $b = 1$, and otherwise exactly simulates Game $\mathbf{G}_1$ for $\mathcal{A}$ when $b = 0$.

Case $b = 1$. In this case, we have:

$$\mathsf{H}_2 = \boldsymbol{\alpha}_2^\intercal \bullet \boldsymbol{\theta}_2 \ .$$

Let us define:

$$\boldsymbol{\alpha}^\intercal = \begin{pmatrix} \boldsymbol{\alpha}_1^\intercal & \boldsymbol{\alpha}_2^\intercal \end{pmatrix} \ .$$

We remark that we have:

$$\boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma \qquad\qquad \mathsf{H} = \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta} \ .$$

This concludes this case, as $\boldsymbol{\alpha}^\intercal$ is clearly uniformly random in $\mathbb{Z}_p^{1\times n}$.

Case $b = 0$. In this case, we have $\mathsf{H}_2 \xleftarrow{\$} \mathfrak{G}$. This case is similar to the previous one, except $\gamma_k = \alpha_{1,1} + \mathsf{H}_2$ is now a uniform random value (independent of $\boldsymbol{\alpha}$). This concludes this case and the proof of the lemma. □

Lemma 5.2.4. *For any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}_2$ against pseudorandomness of $\mathcal{V}_2$ with a similar running time, such that:*

$$|\Pr[\,\boldsymbol{G}_2 = 1\,] - \Pr[\,\boldsymbol{G}_3 = 1\,]| \leq \mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{B}_2, \mathfrak{K}) \ .$$

Proof. The proof is the same as for Lemma 5.2.3, except that H is chosen uniformly at random instead to be defined as $\alpha_{1,1}$. □

We therefore get:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{gl/cs\text{-}m\text{-}psrnd}}(\mathcal{A},\mathfrak{K}) &= |\Pr[\,\mathbf{G}_0 = 1\,] - \Pr[\,\mathbf{G}_3 = 1\,]| \\ &\leq \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_1,\mathfrak{K}) + \varepsilon + \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}_2,\mathfrak{K}) \ .\end{aligned}$$

We get the statement of the proposition by constructing an adversary $\mathcal{B}$ that draws a uniform random integer $i \xleftarrow{\$} \{1,2\}$ and runs $\mathcal{B}_i$. □

Remark 5.2.5. *When $\mathcal{V}_2$ is a KV-DVS for a hard-subset-membership language (Section 5.1.2), there is a much easier proof of mixed pseudorandomness of $\mathcal{V}$: we start by sampling x_2 from $\mathcal{X}_2 \setminus \mathscr{L}_2$ instead of from $\mathscr{L}_2$, which is indistinguishable under the hard-subset-membership property, and then we remark that $(x_1, x_2) \notin \mathscr{L}$ when $x_1 \notin \mathscr{L}_1$ and so smoothness directly implies mixed pseudorandomness.*

5.2.3 CS/KV Disjunctions of a DVS and a Pr-DVS

Let us now show that the DVS corresponding to the CS/KV disjunction of a DVS $\mathcal{V}_1$ and a Pr-DVS $\mathcal{V}_2$ is mixed-pseudorandom, when the DVS $\mathcal{V}_1$ is *witness samplable* [JR14]. By witness samplable, we mean that $\mathsf{Setup.lpar}_1$ generates a trapdoor ltrap_1 in addition to the language parameters lpar_1, such that ltrap_1 enables to efficiently compute the discrete logarithms of the entries of $\Gamma_1(\mathsf{lpar}_1)$.

Theorem 5.2.6. *Let $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two DVSs over two multiplicatively sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. We suppose that $\mathscr{L}_1$ is* witness samplable*. Let $\mathcal{V}$ be the CS/KV disjunction of $\mathcal{V}_1$ and $\mathcal{V}_2$ as defined in Construction 3.2.8.*

If $\mathcal{V}_2$ is a Pr-DVS, then $\mathcal{V}$ is GL/CS-mixed-pseudorandom. More precisely, if $\mathcal{V}_1$ is ε-sound, for any adversary $\mathcal{A}$ against mixed pseudorandomness of the PHF associated to $\mathcal{V}$, there exists any adversary $\mathcal{B}$ against pseudorandomness of the PHF associated to $\mathcal{V}_2$ with similar running time, such that:

$$\mathsf{Adv}^{\mathtt{gl/cs\text{-}m\text{-}psrnd}}(\mathcal{A},\mathfrak{K}) \leq m_1 \cdot \mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{B},\mathfrak{K}) + 2\varepsilon\,,$$

where $m_1 := n_1 - \max_{\mathsf{lpar}_1}(\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1})$, where $\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$ is the dimension of the vector space $\hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$, or in other words the rank of $\Gamma_{1,\mathsf{lpar}_1}$.

Moreover, if $\mathcal{V}_1$ is a KV-DVS, then $\mathcal{V}$ is KV-mixed-pseudorandom with the same bound on the advantage than above (with $\varepsilon = 0$).

Proof. Let us first prove the theorem when $\mathcal{V}_1$ is a KV-DVS (and $\varepsilon = 0$).

From a high level point of view, we show how to simulate $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}b}$ using hash values of x_2 (and the associated projection keys), so that if these hash values are valid, we are exactly in the case $b = 1$, while if these hash values are uniformly random, we are exactly in the case $b = 0$. Contrary to the proof of Theorem 5.2.2, we directly do the reduction to $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}b}$ (without using an intermediate game) and we do not just need one hash value and one projection key, but many of them.

That is why, we start by remarking that using a classical hybrid argument, the following

two distributions are computationally indistinguishable:

$$\mathcal{D}_1 := \left\{ (\boldsymbol{\gamma}_2^\intercal, \mathbf{H}_2) \;\middle|\; \begin{array}{l} \mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar}); \mathfrak{x}_2 \xleftarrow{\$} \mathscr{L}_2; \\ \boldsymbol{\alpha}_2^\intercal \xleftarrow{\$} \mathbb{Z}_p^{m_1 \times n_2};\ \boldsymbol{\gamma}_2^\intercal \leftarrow \boldsymbol{\alpha}_2^\intercal \bullet \Gamma_2 \in \mathfrak{G}_2^{m_1 \times k_2}; \\ \mathbf{H}_2 \leftarrow \boldsymbol{\alpha}_2^\intercal \bullet \boldsymbol{\theta}_2(\mathfrak{x}_2) \in \mathfrak{G}_2^{m_1} \end{array} \right\}$$

$$\mathcal{D}_0 := \left\{ (\boldsymbol{\gamma}_2^\intercal, \mathbf{H}_2) \;\middle|\; \begin{array}{l} \mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar}); \mathfrak{x}_2 \xleftarrow{\$} \mathscr{L}_2; \\ \boldsymbol{\alpha}_2^\intercal \xleftarrow{\$} \mathbb{Z}_p^{m_1 \times n_2};\ \boldsymbol{\gamma}_2^\intercal \leftarrow \boldsymbol{\alpha}_2^\intercal \bullet \Gamma_2 \in \mathfrak{G}_2^{m_1 \times k_2}; \\ \mathbf{H}_2 \xleftarrow{\$} \mathfrak{G}_2^{m_1} \end{array} \right\} .$$

More formally, for any adversary $\mathcal{B}'$ distinguishing these two distributions, we can construct an adversary $\mathcal{B}$ against pseudorandomness, such that:

$$\mathsf{Adv}(\mathcal{B}', \mathfrak{K}) \leq m_1 \cdot \mathsf{Adv}^{\mathtt{psrnd}}(\mathcal{B}, \mathfrak{K}) \ .$$

Let us now simulate $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}b}$ using one sample $(\boldsymbol{\gamma}_2^\intercal, \mathbf{H}_2)$ from $\mathcal{D}_0$ or $\mathcal{D}_1$, such that if this sample comes from $\mathcal{D}_0$, we simulate $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}0}$, while if this sample comes from $\mathcal{D}_1$, we simulate $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}1}$. This will conclude the proof.

We use notation in Figure 5.2. First, using Itrap_1, we compute the discrete logarithms of the entries of Γ_1 and we derive from them a matrix $\Delta \in \mathbb{Z}_p^{m_1 \times n_1}$ such that $\ker \Gamma_1^\intercal = \mathrm{ColSpan}\,(\Delta)$, or in other words, the solutions $\boldsymbol{x}^\intercal \in \mathbb{Z}_p^{1 \times n_2}$ of the linear equation $\boldsymbol{x}^\intercal \bullet \Gamma_1 = \mathbf{0}^\intercal$ are exactly the row vectors $\boldsymbol{x}^\intercal = \boldsymbol{\delta}^\intercal \bullet \Delta$, with $\boldsymbol{\delta}^\intercal \in \mathbb{Z}_p^{1 \times m_1}$. The matrix Δ can be computed by Gaussian elimination.

We then define hp and H as follows:

$$\begin{aligned}
\boldsymbol{\gamma}_2'^\intercal &:= \begin{pmatrix} \gamma_{2,1,1} & \cdots & \gamma_{2,1,k_2} & \cdots & \gamma_{2,m_1,1} & \cdots & \gamma_{2,m_1,k_2} \end{pmatrix} \in \mathfrak{G}_2^{1 \times (mk_2)} \\
\boldsymbol{\alpha}'^\intercal &\xleftarrow{\$} \mathbb{Z}_p^{1 \times n} \\
\boldsymbol{\gamma}^{(1)\intercal} &:= \boldsymbol{\alpha}'^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) && (5.1) \\
\boldsymbol{\gamma}^{(2)\intercal} &:= \boldsymbol{\gamma}_2'^\intercal \bullet (\Delta \otimes \mathsf{Id}_{k_2}) \ \boldsymbol{+}\ \boldsymbol{\alpha}'^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) && (5.2) \\
\boldsymbol{\gamma}^\intercal &:= \begin{pmatrix} \boldsymbol{\gamma}^{(1)\intercal} & \boldsymbol{\gamma}^{(2)\intercal} \end{pmatrix} \\
\mathsf{hp} &:= \boldsymbol{\gamma} \\
\mathsf{H} &:= \mathbf{H}_2^\intercal \bullet \Delta \bullet \boldsymbol{\theta}_1 \ \boldsymbol{+}\ \boldsymbol{\alpha}'^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \ . && (5.3)
\end{aligned}$$

Case $b = 1$. Let us now suppose that $(\boldsymbol{\gamma}_2^\intercal, \mathbf{H}_2) \xleftarrow{\$} \mathcal{D}_1$ and let us prove that hp and H are distributed exactly as in $\mathsf{Exp}^{\mathtt{kv\text{-}m\text{-}psrnd}\text{-}1}$. For that purpose, let $\boldsymbol{\alpha}_2^\intercal \in \mathbb{Z}_p^{m_1 \times n_2}$ be the matrix defined in $\mathcal{D}_1$ and let us set:

$$\begin{aligned}
\boldsymbol{\alpha}_2'^\intercal &:= \begin{pmatrix} \alpha_{2,1,1} & \cdots & \alpha_{2,1,n_2} & \cdots & \alpha_{2,m_1,1} & \cdots & \alpha_{2,m_1,n_2} \end{pmatrix} \in \mathbb{Z}_p^{1 \times (m_1 n_2)} \\
\boldsymbol{\alpha}^\intercal &:= \boldsymbol{\alpha}_2'^\intercal \bullet (\Delta \otimes \mathsf{Id}_{n_2}) \ +\ \boldsymbol{\alpha}'^\intercal \in \mathbb{Z}_p^{1 \times n} \ . && (5.4)
\end{aligned}$$

We remark that $\boldsymbol{\alpha}^\intercal$ is uniformly random, thanks to $\boldsymbol{\alpha}'^\intercal$, and therefore is distributed as a valid hashing key. Furthermore, looking at the definition of $\boldsymbol{\gamma}_2^\intercal$, $\boldsymbol{\gamma}_2'^\intercal$, $\boldsymbol{\alpha}_2^\intercal$, $\boldsymbol{\alpha}_2'^\intercal$, and H_2, we get:

$$\boldsymbol{\gamma}_2'^\intercal = \boldsymbol{\alpha}_2'^\intercal \bullet (\mathsf{Id}_{m_1} \otimes \Gamma_2) \qquad (5.5)$$

$$\mathbf{H}_2 = \boldsymbol{\alpha}_2'^\intercal \bullet (\mathsf{Id}_{m_1} \otimes \boldsymbol{\theta}_2)\,, \qquad (5.6)$$

and thus we have from Equations (5.1) and (5.4):

$$\begin{aligned}\boldsymbol{\alpha}^{\mathsf{T}} \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) &= \boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet (\Delta \otimes \mathsf{Id}_{n_2}) \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \\ &= \boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet ((\Delta \bullet \Gamma_1) \otimes \mathsf{Id}_{n_2}) \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \\ &= \boldsymbol{\gamma}^{(1)\mathsf{T}},\end{aligned}$$

since $\Delta \bullet \Gamma_1 = 0$ by definition of Δ. So $\boldsymbol{\gamma}^{(1)\mathsf{T}}$ is the correct first part of the projection key associated to $\boldsymbol{\alpha}$. And, from Equations (5.2) and (5.5), we get:

$$\begin{aligned}\boldsymbol{\alpha}^{\mathsf{T}} \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) &= \boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet (\Delta \otimes \mathsf{Id}_{n_2}) \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \\ &= \boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet (\Delta \otimes \Gamma_2) + \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \\ &= \boldsymbol{\gamma}^{(2)\mathsf{T}},\end{aligned}$$

because

$$\boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet (\Delta \otimes \Gamma^{(2)}) = \boldsymbol{\alpha}_2'^{\mathsf{T}} \bullet (\mathsf{Id}_{m_1} \otimes \Gamma_2) \bullet (\Delta \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\gamma}_2'^{\mathsf{T}} \bullet (\Delta \otimes \mathsf{Id}_{n_2}),$$

so that $\boldsymbol{\gamma}^{(2)\mathsf{T}}$ is the correct second part of the projection key associated to $\boldsymbol{\alpha}$, and hp is the projection key associated to $\boldsymbol{\alpha}$.

From Equations (5.6) and (5.3), we get:

$$\begin{aligned}\mathsf{H} &= \boldsymbol{\alpha}_{\mathbf{2}}'^{\mathsf{T}} \bullet (\mathsf{Id}_{m_1} \otimes \boldsymbol{\theta}_2) \bullet \Delta \bullet \boldsymbol{\theta}_1 \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \\ &= \boldsymbol{\alpha}_{\mathbf{2}}'^{\mathsf{T}} \bullet ((\Delta \bullet \boldsymbol{\theta}_1) \otimes \boldsymbol{\theta}_2) \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \\ &= \boldsymbol{\alpha}_{\mathbf{2}}'^{\mathsf{T}} \bullet (\Delta \otimes \mathsf{Id}_{n_2}) \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2) \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2),\end{aligned}$$

and by definition of $\boldsymbol{\alpha}^{\mathsf{T}}$ (Equation (5.4)), we have:

$$\mathsf{H} = \boldsymbol{\alpha} \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2), \tag{5.7}$$

hence H is the hash value of (x_1, x_2) under the hashing key $\boldsymbol{\alpha}^{\mathsf{T}}$. In this case, everything has been generated as in the experiment $\mathsf{Exp}^{\texttt{kv-m-psrnd-1}}$.

Case $b = 0$. Let us now suppose that $(\boldsymbol{\gamma}_2^{\mathsf{T}}, \mathbf{H}_2) \xleftarrow{\$} \mathcal{D}_0$ and let us prove that hp and H are distributed exactly as in $\mathsf{Exp}^{\texttt{kv-m-psrnd-0}}$. This case is similar to the previous, except for Equation (5.6), as $\mathbf{H}_2$ is now uniformly random, or in other words, is defined as follows:

$$\boldsymbol{H_2} = \boldsymbol{\alpha}_2''^{\mathsf{T}} \bullet (\mathsf{Id}_{m_1} \otimes \boldsymbol{\theta}_2'), \tag{5.8}$$

with $\boldsymbol{\theta}_2'$ being an arbitrary non-zero vector in $\mathbb{Z}_p^{k_2}$ and $\boldsymbol{\alpha}_2''^{\mathsf{T}}$ being a uniform row vector in $\mathbb{Z}_p^{1 \times m_1 n_2}$ (independent of $\boldsymbol{\alpha}_2'^{\mathsf{T}}$). Except for the above definition, in this case $b = 1$, we use the same notation and definitions (in particular for $\boldsymbol{\alpha}^{\mathsf{T}}$) as for the case $b = 1$, We remark that the proof that hp is the projection key for the hashing key $\boldsymbol{\alpha}$ is still valid. Let us now look at H. From Equations (5.8) and (5.3), using a similar reasoning as the one used to derive Equation (5.7), we get that:

$$\mathsf{H} = \boldsymbol{\alpha}''^{\mathsf{T}} \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2'),$$

where

$$\boldsymbol{\alpha}''^{\mathsf{T}} = \boldsymbol{\alpha}_2''^{\mathsf{T}} \bullet \Delta \ + \ \boldsymbol{\alpha}'^{\mathsf{T}} \ .$$

Since $\boldsymbol{\alpha}_2''^{\mathsf{T}}$ is uniform and independent of everything else, and by definition of Δ, the row vector $\boldsymbol{\alpha}''^{\mathsf{T}}$ can be seen as an independent hashing key chosen uniformly at random among the hashing keys satisfying:

$$\boldsymbol{\alpha}''^{\mathsf{T}} \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) = \boldsymbol{\alpha}'^{\mathsf{T}} \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}).$$

Since $\chi_1 \notin \mathscr{L}_1$, $\boldsymbol{\theta}_1$ is linearly independent from the columns of Γ_1, and $\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2'$ is linearly independent from the columns of $\Gamma_1 \otimes \mathsf{Id}_{n_2}$, hence $\mathsf{H} = \boldsymbol{\alpha}'' \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\theta}_2')$ looks uniformly random (with the same proof as for the smoothness of a SPHF associated to a DVS in Theorem 3.1.11). Therefore, in this case, everything has been generated as in the experiment $\mathsf{Exp}^{\texttt{kv-m-psrnd-0}}$. This concludes the proof when $\mathcal{V}_1$ is a KV-DVS.

When $\mathcal{V}_1$ is a GL-DVS or a CS-DVS, we just abort when $\boldsymbol{\theta}_1 \in \mathrm{ColSpan}\,(\Gamma_1)$, which happens with probability at most ε, by ε-soundness. We remark that checking this condition can be done in polynomial time, as we know the discrete logarithms of the entries of Γ_1 thanks to Itrap_1. This concludes the proof. □

Historical note 5.2.7. *The above proof is a rephrasing of the proof in [ABP15c]. The latter proof was inspired by the proof of computational soundness of* trapdoor smooth projective hash functions *(TSPHFs) in [BBC+13c].*

Remark 5.2.5 also applies.

Chapter 6

Applications of Diverse Modules

This chapter shows four applications of DMs and DVSs related to zero-knowledge: honest-verifier zero-knowledge arguments, *non-interactive zero-knowledge arguments* (NIZK), *trapdoor smooth projective hash functions* (TSPHFs), and *implicit zero-knowledge arguments* (iZK).

On the one hand, the first two applications are classical primitives in cryptography. DMs and DVSs offer a different perspective on honest-verifier zero-knowledge arguments and NIZK, together with more efficient constructions in some cases. In particular, they enable us to construct the most efficient threshold and structure-preserving IND-CCA encryption scheme, to the best of our knowledge.

On the other hand, TSPHFs and iZK are new primitives which can be seen as zero-knowledge versions of SPHFs. Not only are they lightweight alternatives to NIZK, but they also have an implicitness flavor similar to SPHFs, which makes them suitable for applications like *password authenticated key exchange* (PAKE) [BBC+13c] and our protocol for secret agents in Section 2.5.3.3.

All these applications, except honest-verifier zero-knowledge arguments, heavily rely on disjunctions of DMs and DVSs, and for the most efficient variants, on mixed pseudorandomness.

Contents

6.1 Honest-Verifier Zero-Knowledge Arguments **138**
- 6.1.1 Two Dual Constructions From DMs 138
- 6.1.2 Extensions and Comparisons 141

6.2 Non-Interactive Zero-Knowledge Arguments (NIZK) **145**
- 6.2.1 First Constructions 145
- 6.2.2 t-Time Simulation-Soundness 147
- 6.2.3 Concrete Instantiation and Comparison 152
- 6.2.4 Application: Threshold Cramer-Shoup-like Encryption Scheme 153

6.3 Trapdoor Smooth Projective Hashing and Implicit Zero-Knowledge **160**
- 6.3.1 Overview 160
- 6.3.2 Trapdoor Smooth Projective Hash Functions (TSPHFs) 164
- 6.3.3 Implicit Zero-Knowledge Arguments (iZK) 170

6.1 Honest-Verifier Zero-Knowledge Arguments

We have already seen in Section 2.5.3.1 that we can construct an honest-verifier zero-knowledge argument for a language $\ddot{\mathscr{L}}$, from any SPHF for the same language $\mathscr{L} = \ddot{\mathscr{L}}$.

In this section, we go further and show that from a DM there are two dual constructions of honest-verifier zero-knowledge arguments: one via PHFs and another one via Sigma-protocols, a.k.a, proofs "à la Schnorr" [Sch90; Cra97; CDS94]. We then extend these protocols to support (partial) extractability and also any NP languages. Both these extensions work by using a language $\mathscr{L}$ for the DM different from the language $\ddot{\mathscr{L}}$ for the argument.

6.1.1 Two Dual Constructions From DMs

In this section, we want to construct an honest-verifier zero-knowledge proof for some language $\ddot{\mathscr{L}}$, supposing that we have a DM $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ for the same language $\mathscr{L} = \ddot{\mathscr{L}}$. The first construction works for all the languages supported by DMs, in particular for all the examples of DMs and DVSs in Chapters 3 and 4. The second construction has some minor limitations on the representation of scalars, but it works at least in all the examples where the order of the graded ring is public, and in particular for all the examples of DVSs in Chapter 3. One of the most interesting example of DVS for both constructions is the one from Section 3.4.3, which yields an honest-verifier zero-knowledge argument for the language of ElGamal ciphertexts whose corresponding plaintexts satisfy a system of quadratic equations.

As usual, let M' be a factor of M such that $\mathcal{M}$ mod M' is sound (M' can be M, but it is better to choose a proper factor of M which is not divisible by a small prime, if possible). Let p the smallest prime factor of M' and let m be a positive integer such that $1/p^m$ is negligible. In case of a DVS (i.e., $M = p$ is a prime number), we choose $M' = M = p$ and $m = 1$.

6.1.1.1 Via PHFs

We first construct a (weakly approximate) universal PHF from the DM $\mathcal{M}$ using Construction 4.2.5. Then, we use the PHF as in Section 2.5.3.1 (see Figure 2.8 on page 51). For the sake of completeness, Figure 6.1 depicts the resulting 2-round protocol.

We have the following security theorem.

Theorem 6.1.1. *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM with an $\varepsilon_{\mathsf{GenScalar}}$-almost-uniform algorithm* GenScalar*. Let $M' \geq 2$ be a factor of M, such that $\mathcal{M}$ mod M' is ε-sound. Let p be the smallest prime factor of M'. We suppose that $1/p^m$ is negligible in $\mathfrak{K}$. Then the protocol described in Figure 6.1 is a perfectly complete, statistically sound, and perfectly honest-verifier zero-knowledge proof.*[1] *More precisely, the advantage of any (potentially unbounded) adversary against soundness is at most $mn\varepsilon_{\mathsf{GenScalar}} + \varepsilon + 1/p^m$.*

Proof. Perfect completeness directly comes from the perfect correctness property of the DM $\mathcal{M}$.

Perfect honest-verifier zero-knowledge. Simulating a prover against an honest verifier is straightforward: we generate the second flow using the hashing key $(\boldsymbol{\alpha}^\intercal, \rho)$ as $\boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\chi, \rho)$.

[1] In all the schemes of this section, the corresponding simulator is described in the proof.

Prover		Verifier
Input: $(\ddot{\chi} = \chi,\ \ddot{w} = w)$	CRS: lpar	Input: $\ddot{\chi} = \chi$
		$\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})$ $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathsf{GenScalar}^{m\times n}(\mathsf{gpar})$ $\boldsymbol{\gamma}^\intercal \leftarrow \boldsymbol{\alpha}^\intercal \bullet \Gamma(\chi, \rho) \in \mathfrak{G}^{m\times k}$
	$\xleftarrow{\quad \mathsf{hp} \quad}$	$\mathsf{hp} \leftarrow (\boldsymbol{\gamma}^\intercal, \rho)$
$(\boldsymbol{\gamma}^\intercal, \rho) \leftarrow \mathsf{hp}$ $\mathsf{pH} \leftarrow \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(\chi, w, \rho) \in \mathfrak{G}^m$	$\xrightarrow{\quad \mathsf{pH} \quad}$	
		$\mathsf{H} \leftarrow \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\chi, \rho)$ **if** $\mathsf{pH} = \mathsf{H}$ **then** **accept** **else reject**

Figure 6.1: Honest-verifier zero-knowledge proof from a DM via a PHF

Statistical soundness directly follows from Theorem 4.2.15 which shows that the PHF associated to $\mathcal{M}$ is $(mn\varepsilon_{\mathsf{GenScalar}} + \varepsilon)$-weakly-close to be $1/p^m$-universal. □

Remark 6.1.2. *We can improve the communication complexity by making the prover extract a bit string of $\mathfrak{K}$ bits from the projected hash value* pH *using a randomness extractor, and sending this bit string instead of* pH*. As explained in Section 2.2.3, in some cases, this can be done deterministically, as when $\mathfrak{G}$ is a cyclic group defined over some elliptic curve of prime order. But in general, we can always do it by making the verifier send a seed* seed *for the randomness extractor together with the projection key* hp*. Since the size of the seed* seed *might be even larger than the size of* pH*, in many cases, it is better to generate it using a* pseudorandom number generator *(PRG), at the expense of making the soundness only computational. A PRG is a cryptographic primitive which takes as input a (uniform) seed* PRG.seed $\in \{0,1\}^{\mathfrak{K}}$ *and output a longer bit string which is computationally indistinguishable from random. The verifier just needs to send a seed* PRG.seed *for the PRG, and the seed* seed *for the randomness extractor can be deduced from it.*

6.1.1.2 Via Sigma-Protocols

Let us now show another construction of honest-verifier zero-knowledge proofs from the same DM $\mathcal{M}$, via Sigma-protocols [Cra97; CDS94]. Sigma-protocols are specific 3-round honest-verifier zero-knowledge proofs, where the verifier only send some (public) random coins, called a challenge. We do not give a precise definition here, as we do not need it.

We restrict ourselves to the case where there is a unique representation of scalars. If this is not the case, we can use various techniques such as randomization of scalars (as in [GGH13]) or we can use larger representations for Λ (notation from Figure 6.2) to completely mask $\boldsymbol{\lambda}$. However, this goes beyond the scope of this thesis.

The protocol is depicted in Figure 6.2, where the integer q is such that $1/q$ is negligible in the security parameter $\mathfrak{K}$ and $q \leq p$. If the DM $\mathcal{M}$ does not use ρ, then the resulting protocol is a classical 3-round Sigma-protocol. Otherwise, it is a 4-round protocol.

We have the following security theorem.

Prover		Verifier
Input: $(\ddot{\chi} = \chi,\ \ddot{w} = w)$	CRS: lpar	Input: $\ddot{\chi} = \chi$
	$\xleftarrow{\quad \rho \quad}$	$\rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar})$
$\Lambda \xleftarrow{\$} \mathsf{GenScalar}^{k\times m}(\mathsf{gpar})$ $\Theta \leftarrow \Gamma(\chi,\rho) \bullet \Lambda \in \mathfrak{G}^{n\times m}$	$\xrightarrow{\quad \Theta \quad}$	
	$\xleftarrow{\quad \boldsymbol{\beta} \quad}$	$\boldsymbol{\beta}^{\mathsf{T}} \xleftarrow{\$} \{0,\ldots,q-1\}^{1\times m} \subseteq \mathbb{Z}_M^{1\times m}$
$\Lambda' \leftarrow \Lambda + \boldsymbol{\lambda}(\chi,w,\rho) \bullet \boldsymbol{\beta}^{\mathsf{T}} \in \mathbb{Z}_M^{k\times m}$	$\xrightarrow{\quad \Lambda' \quad}$	
		if $\Gamma(\chi,\rho) \bullet \Lambda' = \Theta + \boldsymbol{\theta}(\chi,\rho) \bullet \boldsymbol{\beta}^{\mathsf{T}}$ **then** **accept** **else reject**

Figure 6.2: Honest-verifier zero-knowledge proof from a DM via a Sigma-Protocol

Theorem 6.1.3. *Let $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ be a DM with an $\varepsilon_{\mathsf{GenScalar}}$-almost-uniform algorithm* GenScalar *and such that scalars in $\mathfrak{G}$ have a unique representation. Let $M' \geq 2$ be a factor of M, such that $\mathcal{M}$* mod *M' is ε-sound. Let p be the smallest prime factor of M' and $q \leq p$ be a positive integer. We suppose that $1/q^m$ is negligible in $\mathfrak{K}$. Then the protocol described in Figure 6.1 is a perfectly complete, statistically sound, and perfectly honest-verifier zero-knowledge proof. More precisely, the advantage of any (unbounded) adversary against soundness is at most $\varepsilon + 1/q^m$, while the advantage of any (unbounded) adversary against honest-verifier zero-knowledge is at most $2km\varepsilon_{\mathsf{GenScalar}}$.*

Proof. Perfect completeness is straightforward.

Statistical honest-verifier zero-knowledge. We can simulate a transcript with an honest verifier on a word $\chi \in \mathscr{L}$ as follows:

$$\begin{aligned}
&\boldsymbol{\beta}^{\mathsf{T}} \xleftarrow{\$} \{0,\ldots,q-1\}^{1\times m} \subseteq \mathbb{Z}_M^{1\times m}\\
&\Lambda' \xleftarrow{\$} \mathsf{GenScalar}^{k\times m}(\mathsf{gpar})\\
&\Theta \leftarrow \Gamma(\chi,\rho) \bullet \Lambda' - \boldsymbol{\theta}(\chi,\rho) \bullet \boldsymbol{\beta}^{\mathsf{T}} \ .
\end{aligned}$$

If GenScalar outputs a perfectly uniform distribution, then both in the real world and in the simulated world Λ' are uniformly random (in the real world, this is because Λ is uniformly random). Furthermore, Θ is also distributed the same way in the both worlds, as it is uniquely defined by the equation $\Gamma \bullet \Lambda' = \Theta + \boldsymbol{\theta} \bullet \boldsymbol{\beta}^{\mathsf{T}}$. The simulation is therefore perfect when GenScalar is perfect. When it is $\varepsilon_{\mathsf{GenScalar}}$-almost-uniform, we just lose an additive term of at most $2km\varepsilon_{\mathsf{GenScalar}}$.

Statistical soundness. Let $\chi \notin \mathscr{L}$ and let us suppose that ρ is such that $\boldsymbol{\theta}(\chi,\rho)$ mod M' is linearly independent of the columns of $\Gamma(\chi,\rho)$ mod M'. This happens with probability $1-\varepsilon$. Let us prove that for a given matrix Θ, there exists a unique challenge $\boldsymbol{\beta}^{\mathsf{T}} \in \{0,\ldots,q-1\}^{1\times m}$ such that there exists a valid response $\Lambda' \in \mathbb{Z}_M^{k\times m}$ accepted by the verifier, i.e., such that:

$$\Gamma \bullet \Lambda' = \Theta + \boldsymbol{\theta} \bullet \boldsymbol{\beta}^{\mathsf{T}} \ .$$

This will prove that the protocol is ε'-statistically sound, where

$$\varepsilon' = \varepsilon + 1/|\{0, \ldots, q-1\}^{1\times m}| = \varepsilon + 1/q^m \ .$$

Let us now suppose by contradiction that there exist two distinct vectors $\boldsymbol{\beta}_1^\intercal$ and $\boldsymbol{\beta}_2^\intercal$ in $\{0, \ldots, q-1\}^{1\times m}$, such that there exists two corresponding valid responses $\Lambda_1', \Lambda_2' \in \mathbb{Z}_M^{k\times m}$:

$$\begin{aligned} \Gamma \bullet \Lambda_1' &= \Theta \ \boldsymbol{+} \ \boldsymbol{\theta} \bullet \boldsymbol{\beta}_1^\intercal \\ \Gamma \bullet \Lambda_2' &= \Theta \ \boldsymbol{+} \ \boldsymbol{\theta} \bullet \boldsymbol{\beta}_2^\intercal \ . \end{aligned}$$

We can subtract these two equations and get:

$$\Gamma \bullet (\Lambda_1' - \Lambda_2') = \boldsymbol{\theta} \bullet (\boldsymbol{\beta}_1^\intercal - \boldsymbol{\beta}_2^\intercal) \ .$$

As $\boldsymbol{\beta}_1^\intercal \neq \boldsymbol{\beta}_2^\intercal$, there exists an index i^* such that $\beta_{1,i^*} \neq \beta_{2,i^*}$. In the previous equation, we look at the i^*-th column, and we get:

$$\Gamma \bullet \boldsymbol{\lambda}' = \boldsymbol{\theta} \bullet (\beta_{1,i^*} - \beta_{2,i^*}) \, ,$$

where $\boldsymbol{\lambda}'$ is the i^*-th column of the matrix $\Lambda_1' - \Lambda_2'$. As the integer $\beta_{1,i^*} - \beta_{2,i^*}$ is non-zero and is between $-p+1$ and $p-1$ (as $q \leq p$), it is invertible modulo any prime dividing M and so modulo M (according to Lemma 4.2.12). We get that $\boldsymbol{\theta}$ is in the column space of Γ which is impossible. This concludes the proof. □

Remark 6.1.4. *We can improve the communication complexity at the expense on relying on computational soundness by using a PRG (see Remark 6.1.2 for a definition of PRG) to generate the second flow of the verifier: in this case the verifier just needs to send a seed* PRG.seed *which is a bit string in* $\{0,1\}^{\mathfrak{K}}$.

Historical note 6.1.5. *The first construction of honest-verifier zero-knowledge argument via PHFs was implicit in many work on SPHF and was explicitly introduced in [BBC+13c]. The second construction via Sigma-protocols was also implicit in many works on Sigma-protocols.*

6.1.2 Extensions and Comparisons

6.1.2.1 Partial Extractability

Generic construction. In the CRS model, we can actually add partial extractability in a blackbox way: the prover encrypts the extractable part of the witness $\ddot{w}_{\gimel}$ under a public key for an IND-CPA encryption scheme (in the CRS) and then proves (using one of our previous protocols) that the resulting ciphertext encrypts a valid partial witness $\ddot{w}_{\gimel}$ for the word we are interested in. In the case of our second construction (Sigma-protocol), the resulting protocol is actually an Omega-protocol [GMY06].

More formally, let us show how to construct partially extractable honest-verifier zero-knowledge proofs for the following language:

$$\ddot{\mathscr{L}} = \{\ddot{\chi} \mid \gimel \ddot{w}_{\gimel}, \exists \ddot{w}_{\beth}, \ddot{\mathscr{R}}(\ddot{\chi}, (\ddot{w}_{\gimel}, \ddot{w}_{\beth})) = 1\} \subseteq \ddot{\mathcal{X}} \, ,$$

with the help of an IND-CPA encryption scheme (Setup.gpar, KeyGen, Enc, Dec), with a setup Setup.gpar corresponding to the setup Setup.gpar for the DM $\mathcal{M}$ and with a message space corresponding to the space of the extractable part of the witnesses.

To generate the CRS crs and its trapdoor trap, we generate an encryption/decryption key pair $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{gpar})$ and output $\mathsf{crs} := (\mathsf{lpar}, \mathsf{ek})$ and $\mathsf{trap} := \mathsf{dk}$. The prover then first sends a ciphertext c generated as follows:

$$\mathsf{c} \xleftarrow{\$} \mathsf{Enc}(\mathsf{ek}, \ddot{w}_{\text{Ꙗ}}; r) \quad \text{with fresh random coins } r \ .$$

Then he runs any honest-verifier zero-knowledge argument ZK' for the following language:

$$\mathscr{L} := \{(\ddot{\chi}, \mathsf{c}) \mid \exists r,\ \exists \ddot{w}_{\exists},\ \exists \ddot{w}_{\text{Ꙗ}},\ \mathsf{c} = \mathsf{Enc}(\mathsf{ek}, \ddot{w}_{\text{Ꙗ}}; r) \text{ and } \ddot{\mathscr{R}}(\ddot{\chi}, (\ddot{w}_{\text{Ꙗ}}, \ddot{w}_{\exists})) = 1\} \ . \tag{6.1}$$

Instantiation. We can instantiate ZK' with any of the construction in Section 6.1.1. In the Sigma-protocol version, we can send c with ρ and this does not change the number of rounds, when ρ is really used ($\rho \neq \perp$). In the PHF version, we can send c with H, when the DM is a KV-DM. Otherwise, we need to add an initial flow to send the ciphertext c.

We should however point out that in the 2-round PHF version with a KV-DM, we cannot use a non-deterministic randomness extractor to compress the last flow H (as in Remark 6.1.2).

Security. If the underlying protocol ZK' for $\mathscr{L}$ is perfectly complete, sound, and honest-verifier zero-knowledge, the resulting protocol for $\ddot{\mathscr{L}}$ is perfectly complete, partially extractable, and honest-verifier zero-knowledge.

The simulator just encrypts an arbitrary bit string $\ddot{w}_{\text{Ꙗ}}$ and then simulates the protocol ZK'. This is indistinguishable from an honest execution, under the IND-CPA property of the encryption scheme and the honest-verifier zero-knowledge property of ZK'.

The extractor just acts as an honest verifier except he decrypts the ciphertext c at the end to get the partial witness $\ddot{w}_{\text{Ꙗ}}$. Extraction reference string indistinguishability and extractor indistinguishability are perfect and straightforward as the extractor behaves exactly as an honest verifier from the prover's point of view. Perfect correctness of the encryption scheme and soundness of ZK' ensures that $\ddot{w}_{\text{Ꙗ}}$ is a valid partial witness, and so implies partial extractibility. We point out that if we did no require the encryption scheme to be perfectly correct, we would need to explicitly require that the encryption scheme is committing.

Concrete example: system quadratic pairing equations. Using a conjunction of the DVSs introduced in Section 3.4.3, we can construct a 2-round zero-knowledge argument for languages of the form:

$$\begin{aligned}\ddot{\mathscr{L}} := \Bigg\{ \Big((A_{t,1,i})_{\substack{t=1,\dots,s\\ i=1,\dots,\nu_1}},\ (A_{t,2,j})_{\substack{t=1,\dots,s\\ j=1,\dots,\nu_2}},\ (a_{t,i,j})_{\substack{t=1,\dots,s\\ i=1,\dots,\nu_1\\ j=1,\dots,\nu_2}},\ (a_{t,T,k})_{\substack{t=1,\dots,s\\ k=1,\dots,\nu_T}},\ (B_t)_{t=1,\dots,s} \Big) \ \Bigg| \\ \text{Ꙗ}(\mathsf{M}_{1,i})_{i=1,\dots,\nu_1},\ \text{Ꙗ}(\mathsf{M}_{2,i})_{i=1,\dots,\nu_2},\ \text{Ꙗ}(\mathsf{M}_{T,i})_{i=1,\dots,\nu_T},\ \forall t \in \{1,\dots,s\} \\ \prod_{i=1}^{\nu_1} e(\mathsf{M}_{1,i}, A_{t,2,i}) \cdot \prod_{j=1}^{\nu_2} e(A_{t,1,j}, \mathsf{M}_{2,j}) \cdot \prod_{i=1}^{\nu_1}\prod_{j=1}^{\nu_2} e(\mathsf{M}_{1,i}, \mathsf{M}_{2,j})^{a_{t,i,j}} \cdot \prod_{k=1}^{\nu_T} \mathsf{M}_{T,k}^{a_{t,T,k}} = B_t \Bigg\} \\ \subseteq \mathbb{G}_1^{s\nu_1} \times \mathbb{G}_2^{s\nu_2} \times \mathbb{Z}_p^{s\nu_1\nu_2} \times \mathbb{Z}_p^{s\nu_T} \times \mathbb{G}_T^{s} =: \ddot{\mathcal{X}} \ ,\end{aligned}$$

where s, ν_1, ν_2, ν_T are positive integers, and where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ is a prime order bilinear group. These kinds of languages encompass the languages handled by Groth-Sahai NIZK [GS08] and are actually more powerful, as they support indeterminates $\mathsf{M}_{T,k}$ in $\mathbb{G}_T$.

Table 6.1: Comparison of our constructions of (partially extractable) honest-verifier zero-knowledge arguments from DMs

Construction	Rounds	Communication complexity			
		$\mathfrak{G}$	$\mathbb{Z}_M$	$\{0,1\}^{\mathfrak{K}}$	other
Without extractability ($\ddot{w}_{\text{Я}} = \perp$ and $\mathscr{L} = \ddot{\mathscr{L}}$)					
PHF	2	mk	0	2	$\lvert\rho\rvert$
Sigma-protocol when $\rho = \perp$	3	mn	mk	1	0
Sigma-protocol when $\rho \neq \perp$	4	mn	mk	1	$\lvert\rho\rvert$
With extractability ($\ddot{w}_{\text{Я}} \neq \perp$ and $\mathscr{L}$ is defined as in Equation (6.1))					
PHF with KV-DM	2	$mk + m$	0	0	$\lvert\mathsf{c}\rvert$
PHF	3	mk	0	2	$\lvert\rho\rvert + \lvert\mathsf{c}\rvert$
Sigma-protocol	4	mn	mk	1	$\lvert\rho\rvert + \lvert\mathsf{c}\rvert$

Basically, in the resulting argument, the prover proves that he knows a solution $(\mathsf{M}_{\cdot,\cdot})$ to a system of s quadratic pairing equations defined by the word $\ddot{\chi}$ (which contains the coefficients $A_{\cdot,\cdot}$, $a_{\cdot,\cdot,\cdot}$, and $B_{\cdot}$ of the quadratic equations).

If we allow for an additional round, in many cases, we can improve the communication complexity by compressing the matrix Γ, using in particular batching methods such as the ones in Example 3.1.5. We can also often improve the communication complexity by using randomness reuse in the encryption, see Section 2.2.2.4.

Finally, for a composite order M, the same construction works as soon as m is such that $1/p^m$ is negligible in $\mathfrak{K}$, where p is the smallest prime factor of M.

6.1.2.2 Comparison

Table 6.1 gives a comparison of the communication complexity of the two constructions (using the improvement of Remarks 6.1.2 and 6.1.4). In general, $k < n$ and the construction via PHFs is more efficient. However, there might be DMs where this is not the case (in which case, some columns of the matrix Γ are linearly dependent) and where group elements have a much larger representation than scalars.

We recall that the dimensions k and n of the DM depends on the language $\mathscr{L}$ and of the type of DM: GL-DM can often have much smaller dimensions than KV-DM. In particular, these dimensions are different with and without partial extractability, because the language $\mathscr{L}$ is not the same in these two cases.

6.1.2.3 Support of any NP Language

All the previous constructions require that the language we consider, or the language of ciphertexts of partial witnesses can be handled by a DM. This already includes a large variety of algebraic languages, but not all languages.

Let us show that we can extend the construction to any NP language $\ddot{\mathscr{L}} \subseteq \ddot{\mathscr{X}}$, defined by a circuit C (with gates of fan-in at most 2) taking as input the word $\ddot{\chi}$ and the witness $\ddot{w} = \ddot{w}_{\text{Я}}$ (we suppose that the whole witness is extractable without loss of generality) and

outputting $\mathscr{R}(\ddot{\chi}, \ddot{w})$:

$$\ddot{\mathscr{L}} := \{\ddot{\chi} \mid \exists \ddot{w},\ C(\ddot{\chi}, \ddot{w}) = 1\} \ .$$

We use the same idea as for partial extractability: the prover encrypts additional information in some ciphertexts and then uses an honest-verifier zero-knowledge argument for an extended language $\mathscr{L}$ checking the validity of these ciphertexts with regards to the word $\ddot{\chi}$. Concretely, the prover encrypts using ElGamal the value of all the wires of the circuit C when evaluated on $(\ddot{\chi}, \ddot{w})$ and proves (using one of the previous constructions in Section 6.1.1) that:

- the ciphertexts of corresponding to the input wires of $\ddot{\chi}$ really encrypts the bits of $\ddot{\chi}$;
- the ciphertext corresponding to the output wire encrypts 1;
- each gate is evaluated correctly: if $\mathsf{c}_1, \mathsf{c}_2$ corresponds to two input wires of some gate f, and c_3 corresponds to the output wire the same gate, the plaintexts $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}3$ of $\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3$ respectively have to satisfy $f(x_1, x_2) = x_3$. This can be done naively using GL disjunctions of conjunctions of basic DVS for ElGamal plaintexts (using Example 3.1.9): $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$ is valid if "$\mathsf{m}_1 = 0$ and $\mathsf{m}_2 = 0$ and $\mathsf{m}_3 = f(0, 0)$" or "$\mathsf{m}_1 = 0$ and $\mathsf{m}_2 = 1$ and $\mathsf{m}_3 = f(0, 1)$" or...

This can obviously be optimized, e.g., by not encrypting $\ddot{\chi}$ nor the output wire, by using more efficient DVSs, and by using randomness reuse. A slightly optimized version can be found in [BCPW15]. Anyway, this construction is more a proof of concept of the power of DVSs than a really practical construction.

If we restrict ourselves to languages defined by arithmetic branching programs, we also provide a more efficient direct construction in [BCPW15].

In general, however, we believe that this thesis and these constructions should be seen as a toolbox rather than as ready-to-use solutions to any problem, when it comes to practical instantiations. For any concrete language, it is often better to directly design an optimized DVS inspired by all the examples we give in this thesis, rather than using generic constructions.

6.1.2.4 Toward Zero-Knowledge

None of the previous constructions are proven to be zero-knowledge. For the construction via a PHF, the obstacles to zero-knowledge have already been explained in Section 2.5.3.1, while Sigma-protocols are known not to satisfy zero-knowledge in general.

For Sigma-protocols, zero-knowledge is usually added using the "or" trick: instead of just proving that $\ddot{\chi} \in \ddot{\mathscr{L}}$, we prove that $\ddot{\chi} \in \ddot{\mathscr{L}}$ or some word χ' in the CRS crs is in some hard-subset-membership language $\mathscr{L}'$. When $\chi' \notin \mathscr{L}'$, we get soundness as before, while when $\chi' \in \mathscr{L}'$, we can use a witness for χ' to simulate a proof against any malicious verifier. As the two settings $\chi' \in \mathscr{L}$ and $\chi' \notin \mathscr{L}$ are indistinguishable by hard subset membership, we get a secure zero-knowledge proof. This directly works with Sigma-protocols because these protocols actually satisfy an additional property, namely witness indistinguishability: a prover that uses a witness for χ' is indistinguishable from a prover that uses a witness for $\ddot{\chi}$.

For the protocols based on PHFs, we can try to use the same idea but this would not work directly as these protocols are not witness indistinguishable: the projection key hp might be maliciously generated in such a way that the projected hash value pH depends on the witness used to compute it. For example, in Example 3.2.5, let us suppose that the malicious verifier generates correctly $\mathsf{hk} = \boldsymbol{\alpha}^\intercal$ together with γ_1 and γ_2, but picks γ_3 and γ_4 uniformly

at random. Then, for a word $\mathfrak{x} = (\mathfrak{x}_1 = (u_1, v_1), \mathfrak{x}_2 = (u_2, v_2))$ such that $\mathfrak{x}_1$ and $\mathfrak{x}_2$ are both DH tuples, the projected hash value pH computed using the witness $\mathfrak{w}_1$ for $\mathfrak{x}_1$ is equal to its hash value H (that the malicious verifier can compute using hk), but the projected value pH computed using the witness $\mathfrak{w}_2$ for $\mathfrak{x}_2$ is uniformly random (and it not equal to H with overwhelming probability). That way, the malicious adversary can know whether the prover used $\mathfrak{w}_1$ or $\mathfrak{w}_2$ to compute pH. This makes the "or" trick fails.

To get zero-knowledge, we need to ensure that the projection key hp is correctly generated. That is exactly what we are doing with TSPHFs and iZK (see Section 6.3).

6.2 Non-Interactive Zero-Knowledge Arguments (NIZK)

We have already seen in Section 3.3 how to construct a (constant-size) NIZK using a disjunction of two DVSs: one corresponding to the language $\mathscr{L}_1 = \ddot{\mathscr{L}}$ of the NIZK and one for a hard-subset-membership language $\mathscr{L}_2$. We first extend this construction to DMs and improve it using mixed pseudorandomness at the expense of requiring $\ddot{\mathscr{L}}$ to be witness samplable. We then show how to construct stronger forms of NIZK, namely t-time simulation-sound NIZK using t-sound extensions. We conclude by comparing our constructions with existing ones and by showing a concrete application: the construction of a threshold Cramer-Shoup-like encryption scheme.

6.2.1 First Constructions

6.2.1.1 Construction Based on Disjunctions with a Hard-Subset-Membership Language

We can extend the construction of NIZK based on KV disjunctions of DVSs (Construction 3.3.1) to DMs.

Construction 6.2.1. *Let* $\mathcal{M}_1 = (M, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{M}_2 = (M, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DMs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$. *Let* m *be a positive integer. Finally, let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be the KV disjunction of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$. *We recall that* $n = n_1 n_2$ *and* $k = k_1 n_2 + n_1 k_2$.

We construct a NIZK for the language $\ddot{\mathscr{L}} := \mathscr{L}_1$ *as follows:*

- $\mathsf{NIZK.Setup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar_2}(\mathsf{gpar})$ *for* $\mathscr{L}_2$, *together with a hashing key* hk *and the associated projection key* hp *for* $\mathcal{V}$ *as follows:*

$$\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathsf{GenScalar}^{m\times n}(\mathsf{gpar}),$$
$$\mathsf{hp} := \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma \quad \in \mathfrak{G}^{m\times k},$$

  and outputs $\mathsf{crs} := (\mathsf{lpar}_2, \mathsf{hp})$. *In the sequel, we split* $\boldsymbol{\gamma}^\intercal$ *in two parts:*

$$\boldsymbol{\gamma}_1^\intercal := (\gamma_{j,i})_{\substack{i=1,\ldots,m\\ j=1,\ldots,k_1n_2}} = \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m\times(k_1n_2)},$$
$$\boldsymbol{\gamma}_2^\intercal := (\gamma_{j,i})_{\substack{i=1,\ldots,m\\ j=k_1n_2+1,\ldots,n}} = \boldsymbol{\alpha}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \quad \in \mathfrak{G}_2^{m\times(n_1k_2)};$$

- $\mathsf{NIZK.Sim_1}(\mathsf{lpar}_1)$ *works as* NIZK.Setup *except it also outputs the following trapdoor:*

$$\mathsf{trap} := \mathsf{hk} := \boldsymbol{\alpha}^\intercal \quad \in \mathbb{Z}_p^{m\times n};$$

- $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, x_1, w_1)$ *outputs:*

$$\boldsymbol{\pi}^\mathsf{T} := \boldsymbol{\gamma}_1^\mathsf{T} \bullet (\boldsymbol{\lambda}_1(x_1, w_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m \times n_2} ;$$

- $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, x_1, \boldsymbol{\pi}^\mathsf{T})$ *checks the following equation:*

$$\boldsymbol{\pi}^\mathsf{T} \bullet \Gamma_2 \stackrel{?}{=} \boldsymbol{\gamma}_2^\mathsf{T} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{k_2}) ;$$

- $\mathsf{NIZK.Sim}_2(\mathsf{trap}, \mathsf{tag}, x_1)$ *outputs:*

$$\boldsymbol{\pi}^\mathsf{T} := \boldsymbol{\alpha}^\mathsf{T} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m \times n_2} \ .$$

Tags tag *are not used in this construction.*

When the order $M = p$ is prime, we can choose $m = 1$ and get Construction 3.3.1.

We have the following security theorem, which extends Theorem 3.3.2. The assumptions on $\mathcal{M}_1$ and $\mathcal{M}_2$ are more complex due to the problems with degenerated words and small factors of M.

Theorem 6.2.2. *Let* $\mathcal{M}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{M}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DVSs over two multiplicatively compatible subgraded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*, with an* $\varepsilon_{\mathsf{GenScalar}}$*-almost-uniform algorithm* GenScalar*. Let* $M' \geq 2$ *be a factor of* M *such that* $\mathcal{M}_1 \bmod M'$ *and* $\mathcal{M}_2 \bmod M'$ *are sound. Let* p *be the smallest prime factor of* M'*. We suppose that* $\mathscr{L}_2$ *is a hard-subset-membership language, and that a random word* $x_2 \xleftarrow{\$} \mathcal{X}_2 \setminus \mathscr{L}_2$ *is strongly non-degenerated in* $\mathcal{M}_2 \bmod M'$ *with overwhelming probability* $1 - \varepsilon$*. We finally suppose that* $1/p^m$ *is negligible in* $\mathfrak{K}$*.*

Then the NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.2.1 is perfectly complete, perfectly zero-knowledge and sound. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against soundness of the NIZK, we can construct an adversary* $\mathcal{B}$ *against subset-membership for* $\mathscr{L}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{sound}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{sub\text{-}memb}(\mathcal{B}, \mathfrak{K}) + mn\varepsilon_{\mathsf{GenScalar}} + \frac{1}{p^m} + \varepsilon \ .$$

We remark that being strongly non-degenerated in $\mathcal{M}_2 \bmod M'$ is weaker than being strongly generated (in $\mathcal{M}_2$).

Proof. Perfect completeness and *perfect zero-knowledge* are proven exactly as for Theorem 3.3.2, as the proof is just syntactical.

For *soundness*, everything is similar to the latter proof except for the end, when we want to prove that if $x_2 \notin \mathscr{L}_2$ and $x_1 \notin \mathscr{L}_1$, then the hash value H looks uniformly random. This is no more true. Instead, we look at $\mathcal{M} \bmod M'$ which can be seen as the KV disjunction of $\mathcal{M}_1 \bmod M'$ and $\mathcal{M}_2 \bmod M'$. We suppose that x_2 is strongly non-degenerated in $\mathcal{M}_2 \bmod M'$. This happens with overwhelming probability $1 - \varepsilon$. The reduced DM $\mathcal{M} \bmod M'$ is then sound for the set $\mathscr{S} = \{x_1, x_2\}$ (thanks to Proposition 4.3.16). We conclude by using Theorem 4.4.8 which ensures that the resulting PHF is $mn\varepsilon_{\mathsf{GenScalar}}$-close to be $1/p^m$-$\mathscr{S}$-universal, and so that the hash value H of (x_1, x_2) cannot be guessed by an adversary with probability more than $1/p^m$ (at least when $\varepsilon_{\mathsf{GenScalar}} = 0$). □

6.2.1.2 Construction Based on Mixed Pseudorandomness

Let us now show how to improve the size of the proof, by using a Pr-DVS as second DVS $\mathcal{V}_2$ and mixed pseudorandomness, in the prime order case (i.e., when DMs are actually DVSs). We need to add a slight restriction on the language of the NIZK $\ddot{\mathscr{L}} = \mathscr{L}_1$: it has to be witness samplable (see Section 5.2.3).

Construction 6.2.3. *The construction is exactly the same as Construction 3.3.1, except that we do not require that $\mathscr{L}_2$ is a hard-subset-membership language but only that $\mathcal{V}_2$ is a Pr-DVS. We also suppose that $\mathscr{L}_1$ is witness samplable.*

We point out that we never use the trapdoor ltrap_1 of $\mathscr{L}_1$ for witness samplability in the NIZK scheme. It is only used for the proof.

Theorem 6.2.4. *Let $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two KV-DVSs over two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. We suppose that $\mathcal{V}_2$ is a Pr-DVS and that $\mathscr{L}_1$ is witness samplable.*

Then the NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ in Construction 6.2.3 is perfectly complete, perfectly zero-knowledge, and sound. More precisely, if $\mathcal{A}$ is a polynomial-time adversary against soundness of the NIZK, we can construct an adversary $\mathcal{B}$ against pseudorandomness of $\mathcal{V}_2$ with similar running time such that:

$$\mathsf{Adv}^{sound}(\mathcal{A}, \mathfrak{K}) \leq m_1 \cdot \mathsf{Adv}^{psrnd}(\mathcal{B}, \mathfrak{K}) + \frac{1}{p} \ .$$

where $m_1 := n_1 - \max_{\mathsf{lpar}_1}(\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1})$, where $\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$ is the dimension of the vector space $\hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$, or in other words the rank of $\Gamma_{1,\mathsf{lpar}_1}$.

Proof. Perfect completeness and *perfect zero-knowledge* are proven exactly as for Theorem 3.3.2, as the proof is just syntactical.

For *soundness* as for the proof of Theorem 6.2.2, everything is similar to the proof of Theorem 3.3.2 except for the end, when we want to prove that if $\mathfrak{x}_2 \notin \mathscr{L}_2$ and $\mathfrak{x}_1 \notin \mathscr{L}_1$, then the hash value H looks uniformly random. This is no more true statistically, but this directly comes from mixed pseudorandomness. We conclude using Theorem 5.2.6.

There is a slight hidden subtlety: since the proof is no more statistical, we should be careful that the games can be simulated in polynomial time. This was not the case before. But now, the trapdoor ltrap_1 enables to compute the discrete logarithms of the entries of Γ_1 and so enable to efficiently check whether a word $\mathfrak{x}_1$ is in $\mathscr{L}_1$ or not. We remark that this trapdoor is available to the adversary $\mathcal{B}$ in the experiments $\mathsf{Exp}^{\texttt{kv-m-psrnd-}b}$ for mixed pseudorandomness. □

6.2.2 t-Time Simulation-Soundness

Let us now extend the previous constructions to get t-time simulation-soundness.

6.2.2.1 Construction Based on Disjunctions with a Hard-Subset-Membership Language

If we try to do a proof of t-time simulation-soundness for the NIZK in Construction 3.3.1 for example, we have the following problem: we cannot rely on smoothness to prove that the

hash value H looks uniform at the end of the proof, as we may have used the hashing key $\mathsf{trap} = \mathsf{hk}$ to simulate t other proofs.

That is exactly why $(t+1)$-universality, $(t+1)$-smoothness, and $(t+1)$-soundness have been invented: we just need to consider the $(t+1)$-sound extension of the KV disjunction $\mathcal{V}$ or $\mathcal{M}$. To make notation easier to follow however, we consider the KV disjunction of a $(t+1)$-sound extension of $\mathcal{M}_1$, and of $\mathcal{M}_2$. But we recall that this is equivalent to the $(t+1)$-sound extension of the KV disjunction of $\mathcal{M}_1$ and $\mathcal{M}_2$ (see Remark 4.4.17).

Construction 6.2.5. *Let* $\mathcal{M}_1 = (M, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{M}_2 = (M, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DMs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*, with an* $\varepsilon_{\mathsf{GenScalar}}$*-almost-uniform algorithm* $\mathsf{GenScalar}$*. We suppose that* $\mathcal{M}_1$ *is a tag-KV-DM with tag set* Tags*, and for the security proof, that it is* $(t+1)$*-sound (possibly modulo some integer* M' *such that* $\mathcal{M}_2 \bmod M'$ *is sound). We recall that such a KV-DM can be constructed in a blackbox way from any KV-DM using Construction 4.4.15. Let* m *be a positive integer. Finally, let* $\mathcal{M} = (M, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *be the KV disjunction of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*.*

We construct a NIZK for the language $\ddot{\mathscr{L}} := \mathscr{L}_1$ *with tag set* Tags *as follows:*

- $\mathsf{NIZK.Setup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with a hashing key* hk *and the associated projection key* hp *for* $\mathcal{V}$ *as follows:*

$$\begin{aligned} \mathsf{hk} &:= \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathsf{GenScalar}^{m\times n}(\mathsf{gpar}) \,, \\ \mathsf{hp} &:= \boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma \quad \in \mathfrak{G}^{m\times k} \,, \end{aligned}$$

  and outputs $\mathsf{crs} := (\mathsf{lpar}_2, \mathsf{hp})$*. In the sequel, we split* $\boldsymbol{\gamma}^\intercal$ *in two parts:*

$$\begin{aligned} \boldsymbol{\gamma}_1^\intercal &:= (\gamma_{j,i})_{\substack{i=1,\dots,m \\ j=1,\dots,k_1 n_2}} = \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m\times(k_1 n_2)} \,, \\ \boldsymbol{\gamma}_2^\intercal &:= (\gamma_{j,i})_{\substack{i=1,\dots,m \\ j=k_1 n_2+1,\dots,n}} = \boldsymbol{\alpha}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \quad \in \mathfrak{G}_2^{m\times(n_1 k_2)} \,; \end{aligned}$$

- $\mathsf{NIZK.Sim}_1(\mathsf{lpar}_1)$ *works as* $\mathsf{NIZK.Setup}$ *except it also outputs the following trapdoor:*

$$\mathsf{trap} := \mathsf{hk} = \boldsymbol{\alpha}^\intercal \quad \in \mathbb{Z}_p^{m\times n} \,;$$

- $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, x_1, w_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1(x_1, w_1, \mathsf{tag}) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m\times n_2} \,;$$

- $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, x_1, \boldsymbol{\pi}^\intercal)$ *checks the following equation:*

$$\boldsymbol{\pi}^\intercal \bullet \Gamma_2 \stackrel{?}{=} \boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1(x_1, \mathsf{tag}) \otimes \mathsf{Id}_{k_2}) \,;$$

- $\mathsf{NIZK.Sim}_2(\mathsf{trap}, \mathsf{tag}, x_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \boldsymbol{\alpha}^\intercal \bullet (\boldsymbol{\theta}_1(x_1, \mathsf{tag}) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{m\times n_2} \,.$$

We have the following security theorem.

Theorem 6.2.6. *Let t be a positive integer. Let $\mathcal{M}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ and $\mathcal{M}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ be two KV-DVSs over two multiplicatively compatible sub-graded rings $\mathfrak{G}_1$ and $\mathfrak{G}_2$ of some graded ring $\mathfrak{G}$. Let $M' \geq 2$ be a factor of M such that $\mathcal{M}_1 \bmod M'$ is $(t+1)$-sound and $\mathcal{M}_2 \bmod M'$ is sound. Let p be the smallest prime factor of M'. We suppose that $\mathscr{L}_2$ is a hard-subset-membership language, and that a random word $x_2 \xleftarrow{\$} \mathcal{X}_2 \setminus \mathscr{L}_2$ is strongly non-degenerated in $\mathcal{M}_2 \bmod M'$ with overwhelming probability $1 - \varepsilon$. We finally suppose that $1/p^m$ is negligible in $\mathfrak{K}$.*

Then the NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ in Construction 6.2.1 is perfectly complete, perfectly zero-knowledge, and t-time simulation-sound. More precisely, if $\mathcal{A}$ is a polynomial-time adversary against t-time simulation-soundness of the NIZK, we can construct an adversary $\mathcal{B}$ against subset-membership for $\mathscr{L}_2$ with similar running time such that:

$$\mathsf{Adv}^{t\text{-}sim\text{-}sound}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{sub\text{-}memb}(\mathcal{B}, \mathfrak{K}) + mn\varepsilon_{\mathsf{GenScalar}} + \frac{1}{p^m} + \varepsilon \ .$$

We recall t-time simulation-soundness directly implies soundness.

Construction 6.2.5 enables to build NIZK for the same languages as for normal NIZK in Construction 6.2.1, as we can extend any KV-DM $\mathcal{M}_1$ to a $(t+1)$-sound KV-DM, using Construction 4.4.15. We remark that, as we only require $\mathcal{M}_1 \bmod M'$ to be $(t+1)$-sound, this previous construction can be slightly optimized: more precisely, we just need to use t-independent strongly non-degenerated functions modulo M' instead of modulo M. In any case, the dimensions n_1 and k_1 get multiplied by some integer $s \geq t+1$. The exact value of s depends on the tag set and the smallest prime factor p of M'. Compared to Construction 6.2.1, the size of the proof is exactly the same, and the size of the CRS is multiplied by s.

When M is a prime number, any word in $\mathcal{X}_2 \setminus \mathscr{L}_2$ is strongly non-degenerated, and we can choose $M' = M = p$, $m = 1$, and $s = t+1$ for the tag set $\mathsf{Tags} = \mathbb{Z}_M$.

Proof. Perfect completeness and *perfect zero-knowledge* are proven exactly as for Theorem 6.2.2, as the proof is just syntactical.

We need to prove the *t-time simulation-soundness* property. The proof is similar to the proof of soundness for Theorem 6.2.2: instead of using universality we use $(t+1)$-universality which follows from the fact that $\mathcal{M}$ is t-sound thanks to Remark 4.4.17. □

6.2.2.2 Construction Based on Mixed Pseudorandomness

Let us now try to improve the previous construction using pseudorandomness in the prime order case, as we did with normal NIZK in Construction 6.2.3. Unfortunately, this is not as easy: Construction 6.2.5 seems difficult (if at all possible) to prove t-time simulation-sound when the second DM $\mathcal{M}_2$ is just a Pr-DVS. The main problem is that the security proof of mixed pseudo-randomness is not statistical, and an adversary for mixed pseudorandomness has not access to $\mathsf{hk} = \boldsymbol{\alpha}$, but has only access to some representation of $\boldsymbol{\alpha}$, which does not allow to simulate proofs for words and tags chosen by the adversary.

We could actually do a completely manual proof, in the specific case where $\mathcal{M}_1$ is constructed via Construction 4.4.15 and if the t tags for the simulated proofs are known before the generation of the CRS crs of the NIZK. This is already useful, as we can then bootstrap this into a real t-time simulation-sound NIZK by using a one-time signature: the tag tag becomes a public key for a one-time signature (and so can be chosen in advance before the generation of the CRS in the security proof) and the whole NIZK proof is signed using this public key.

However, the previous construction requires to use a one-time signature and therefore increases the proof size. This increase is likely to be larger than the gain obtained using a Pr-DVS instead of a KV-DVS for a hard-subset-membership language. Thus, let us instead show how to construct a one-time simulation-sound NIZK with the same proof size, as the normal NIZK from Construction 6.2.3. We focus on one-time simulation-soundness, as the construction is already not straightforward and one-time simulation-soundness is already very useful. In particular, it is sufficient to construct a threshold Cramer-Shoup-like encryption scheme in Section 6.2.4.

Our solution is to use the tag bit by bit. So we just need to guess which bit is different between the tag tag of the proof generated by the adversary and the tag tag' for the simulated proof. This idea is inspired from [CW13]. Concretely, we replace the first DVS $\mathcal{V}_1$ in Construction 6.2.3 by its 2-sound extension (Construction 4.4.15) using the tag set $\mathsf{Tags} = \{0,1\}^\nu$ (for any positive integer ν) and the following 2-independent strongly non-degenerated function modulo $M = p$:

$$\phi : \begin{pmatrix} \mathsf{Tags} & \to & \mathbb{Z}_p^{2\nu} \\ \mathsf{tag} & \mapsto & \begin{pmatrix} \mathsf{tag}_1 \\ 1-\mathsf{tag}_1 \\ \mathsf{tag}_2 \\ 1-\mathsf{tag}_2 \\ \vdots \\ \mathsf{tag}_\nu \\ 1-\mathsf{tag}_\nu \end{pmatrix} \end{pmatrix} .$$

Let us now show the construction in a more concrete way.

Construction 6.2.7. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DVSs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. For the security proof, we suppose that* $\mathcal{V}_2$ *is a Pr-DVS. Furthermore, we consider the KV disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*. We recall that* $n = n_1 n_2$ *and* $k = k_1 n_2 + n_1 k_2$*. Finally, let* ν *be some positive integer, and let* Tags *be the tag set* $\mathsf{Tags} = \{0,1\}^\nu$*.*

We construct a NIZK for $\ddot{\mathscr{L}} := \mathscr{L}_1$ *as follows:*

- $\mathsf{NIZK.Setup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with* 2ν *hashing key* $\mathsf{hk}_{k,b}$ *and the associated projection keys* $\mathsf{hp}_{k,b}$*, for* $k \in \{1, \dots, \nu\}$ *and* $b \in \{0,1\}$*, as follows:*

$$\begin{aligned} \mathsf{hk}_{k,b} &:= \boldsymbol{\alpha}_{k,b}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n} , \\ \mathsf{hp}_{k,b} &:= \boldsymbol{\gamma}_{k,b}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma \quad \in \mathfrak{G}^{1\times k} , \end{aligned}$$

and outputs $\mathsf{crs} := (\mathsf{lpar}_2, (\mathsf{hp}_{k,b})_{\substack{k=1,\dots,\nu \\ b=0,1}})$*. In the sequel, we split* $\boldsymbol{\gamma}_{k,b}^\intercal$ *in two parts:*

$$\begin{aligned} \boldsymbol{\gamma}_{k,b,1}^\intercal &:= (\gamma_{k,b,i})_{i=1,\dots,k_1 n_2}^\intercal = \boldsymbol{\alpha}_{k,b}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times(k_1 n_2)} , \\ \boldsymbol{\gamma}_{k,b,2}^\intercal &:= (\gamma_{k,b,i})_{i=k_1 n_2+1,\dots,n}^\intercal = \boldsymbol{\alpha}_{k,b}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \quad \in \mathfrak{G}_2^{1\times(n_1 k_2)} ; \end{aligned}$$

- $\mathsf{NIZK.Sim}_1(\mathsf{lpar}_1)$ *works as* $\mathsf{NIZK.Setup}$ *except it also outputs the following trapdoor:*

$$\mathsf{trap} := (\mathsf{hk}_{k,b})_{\substack{k=1,\dots,\nu \\ b=0,1}} ;$$

- $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, x_1, w_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \sum_{k=1}^{\nu} \boldsymbol{\gamma}_{k,\mathsf{tag}_k,1}^\intercal \bullet (\boldsymbol{\lambda}_1(x_1, w_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times n_2} ;$$

- $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, x_1, \boldsymbol{\pi}^\intercal)$ *checks the following equation:*

$$\boldsymbol{\pi}^\intercal \bullet \Gamma_2 \stackrel{?}{=} \sum_{k=1}^{\nu} \boldsymbol{\gamma}_{k,\mathsf{tag}_k,2}^\intercal \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{k_2}) ; \tag{6.2}$$

- $\mathsf{NIZK.Sim}_2(\mathsf{trap}, \mathsf{tag}, x_1)$ *outputs:*

$$\boldsymbol{\pi}^\intercal := \sum_{k=1}^{\nu} \boldsymbol{\alpha}_{k,\mathsf{tag}_k}^\intercal \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times n_2} \quad .$$

We can see the CRS and its associated trapdoor as 2ν CRSs and associated trapdoors for Construction 6.2.3. A proof $\boldsymbol{\pi}^\intercal$ is the sum of the ν proofs in Construction 6.2.3 for the projection keys or CRSs $\mathsf{hp}_{k,\mathsf{tag}}$. Everything works nicely thanks to the linearity of the proofs.

We insist on the fact that in this construction, contrary to Construction 6.2.5, $\mathcal{V}_1$ (or $\mathcal{M}_1$) is a normal DVS (and not necessarily $(t+1)$-sound). The size of the proof is the same as the one in Construction 6.2.3, while the CRS is 2ν times larger.

Theorem 6.2.8. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two KV-DVSs over two multiplicatively compatible subgraded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. We suppose that* $\mathcal{V}_2$ *is a Pr-DVS and that* $\mathscr{L}_1$ *is witness samplable.*

Then the NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.2.7 is perfectly complete, perfectly zero-knowledge, and one-time simulation-sound. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against one-time simulation-soundness of the NIZK, we can construct an adversary* $\mathcal{B}$ *against pseudorandomness of* $\mathcal{V}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{sound}(\mathcal{A}, \mathfrak{K}) \leq 2\nu \left(m_1 \cdot \mathsf{Adv}^{psrnd}(\mathcal{B}, \mathfrak{K}) + \frac{1}{p} \right) \quad .$$

where $m_1 := n_1 - \max_{\mathsf{lpar}_1}(\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1})$*, where* $\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$ *is the dimension of the vector space* $\hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$*, or in other words the rank of* $\Gamma_{1,\mathsf{lpar}_1}$*.*

Proof. Perfect correctness and perfect zero-knowledge can be proven as in Theorem 3.3.2.

Let us prove one-time simulation-soundness, with a reduction to soundness of Construction 3.3.1 (for the same DVSs $\mathcal{V}_1$ and $\mathcal{V}_2$). Let us consider a polynomial-time adversary $\mathcal{A}$ that generates a valid proof $\boldsymbol{\pi}^\intercal$ for a word $x_1 \notin \mathscr{L}_1$ and a tag tag, after having asked for a proof $\boldsymbol{\pi}'^\intercal$ for a word $x_1' \in \mathcal{X}_1$ and a tag $\mathsf{tag}' \neq \mathsf{tag}$. We construct a polynomial-time adversary $\mathcal{B}$ against soundness of Construction 3.3.1.

The adversary $\mathcal{B}$ guesses an index $k^* \in \{1, \ldots, \nu\}$ and a bit $b^* \in \{0, 1\}$ such that $\mathsf{tag}' = 1 - b^*$ and $\mathsf{tag} = b^*$. If the guess is incorrect, it returns a random bit at the end of the experiment. There is at least one such pair and the adversary $\mathcal{B}$ guesses correctly with probability at least $1/(2\nu)$.

The adversary $\mathcal{B}$ then gets a CRS hp_{k^*,b^*} from the soundness experiment. It also generates honestly the hashing keys and projections keys $\mathsf{hk}_{k,b}$ and $\mathsf{hp}_{k,b}$ for $(k, b) \neq (k^*, b^*)$. In particular, this means that it can easily honestly simulate $\boldsymbol{\pi}'^{\mathsf{T}}$. Furthermore, it computes:

$$\boldsymbol{\pi}^{*\mathsf{T}} := \boldsymbol{\pi} - \sum_{\substack{k=1 \\ k \neq k^*}}^{\nu} \boldsymbol{\alpha}^{\mathsf{T}}_{k,\mathsf{tag}_k} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{n_2}) \ .$$

Using a proof like with Equation (3.10) on page 79, for $k \neq k^*$, we have:

$$\boldsymbol{\alpha}^{\mathsf{T}}_{k,\mathsf{tag}_k} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{n_2}) \bullet \Gamma_2 = \boldsymbol{\gamma}^{\mathsf{T}}_{k,\mathsf{tag}_k,2} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{k_2}) \ .$$

From Equation (6.2), we get:

$$\boldsymbol{\pi}^{*\mathsf{T}} \bullet \Gamma_2 = \boldsymbol{\gamma}^{\mathsf{T}}_{k^*,\mathsf{tag}_{k^*},2} \bullet (\boldsymbol{\theta}_1(x_1) \otimes \mathsf{Id}_{k_2}) \ .$$

In other words, the adversary $\mathcal{B}$ has constructed a valid proof $\boldsymbol{\pi}^{*\mathsf{T}}$ for x_1 under the CRS hp_{k^*,b^*}. This concludes the proof. □

6.2.3 Concrete Instantiation and Comparison

Prime order. Let us first focus on the prime order case, where $\mathfrak{G}$ is a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, $\mathfrak{G}_1$ corresponds to $\mathbb{G}_1$ and $\mathfrak{G}_2$ corresponds to $\mathbb{G}_2$. The languages $\mathscr{L}$ we can handle are languages arising from KV-DVSs over $\mathbb{G}_1$, and so are basically *linear subspaces over* $\mathbb{G}_1$. We can instantiate all our constructions under any $\mathcal{D}$-MDDH assumption on $\mathbb{G}_2$. For the ones using mixed pseudorandomness, we should use Construction 5.1.3 for the second DVS $\mathcal{V}_2$, while for the other ones, we should use the construction in Section 3.4.1 for the second DM $\mathcal{M}_2$.

All our constructions, except the t-time simulation-sound NIZK for $t > 1$ (in Construction 6.2.5), were introduced in [ABP15c] and were later improved by Kiltz and Wee to rely on computational assumptions (instead of decisional ones) in [KW15]. The schemes common to both papers are the same up to a small difference: the schemes based on mixed pseudorandomness used one more group element in the CRS in [ABP15c], because we did not choose an optimal representation for the matrix Γ for the second DM.

Let us show an instantiation under the κ-linear assumption in $\mathbb{G}_2$. Concretely, we see the κ-linear assumption as the $\mathcal{D}$-MDDH assumption, with $\mathcal{D}$ being the following matrix distribution:

$$\mathcal{D} = \left\{ \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & a_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & a_k \\ a_1 & \ldots & a_1 & a_1 \end{pmatrix} \in \mathbb{Z}_p^{(\kappa+1)\times\kappa} \;\middle|\; a_1, \ldots, a_k \xleftarrow{\$} \mathbb{Z}_p \right\} \ .$$

This is slightly non-standard: usually, as was implicitly done in [ABP15c], the top-left element is a uniform scalar a_1, while the last row only contain the scalar 1. This non-standard representation (completely equivalent to the standard one) enables to reduce the representation of the first κ rows of the matrix (denoted $\bar{A}$ in Section 5.1.3) by one element and to get the same CRS size as Kiltz and Wee in [KW15]. We also recall that the κ-linear assumption corresponds to the DDH assumption when $\kappa = 1$, and to the DLin assumption when $\kappa = 2$.

Let us now compare our constructions in the DDH and DLin cases with other constructions of (one-time simulation-sound or normal) NIZK. We recall that [KW15] improves the assumptions from decisional to computational.

Table 6.2 compares NIZK for linear subspaces over $\mathbb{G}_1$. Some of the entries of this table were derived from [JR14] and from [LPJY14]. The DDH (in $\mathbb{G}_2$) variant requires asymmetric bilinear groups, while the DLin-based (and more generally, the one based on the κ-linear assumption) can also work on symmetric bilinear groups.

First of all, as far as we know, our one-time simulation-sound NIZK was the most efficient such NIZK with a constant-size proof: the single-theorem relatively-sound construction of Libert et al. [LPJY14] is weaker than our one-time simulation-sound NIZK and requires at least one more group element in the proof, while their universal simulation-sound construction is much more inefficient. A direct application of our construction is our efficient structure-preserving threshold IND-CCA encryption scheme, under DDH, in Section 6.2.4.

Second, the DLin version of our NIZK in Construction 6.2.1 is similar to the one by Libert et al. [LPJY14], but our DLin version of our NIZK in Construction 6.2.3 is more efficient (the proof has 2 group elements instead of 3). Furthermore, the ideas of the constructions in [LPJY14] seem quite different.

Third, our NIZK in Construction 6.2.3 is similar to the one by Jutla and Roy in [JR14] for DDH. However, in our opinion, our construction seems to be more modular and simpler to understand. In addition, under the κ-linear assumption, with $\kappa \geq 2$, our construction is slightly more efficient in terms of CRS size and verification time.

Composite order. We can instantiate our NIZK constructions that are not based on pseudo-randomness, in any bilinear group $(M, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of composite order M, using any sound DM $\mathcal{M}_2 = (M, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ over $\mathbb{G}_2$ such that a random word in $\mathcal{X}_2 \setminus \mathscr{L}_2$ is strongly non-degenerated with overwhelming probability. This include DMs constructed from any $\mathcal{D}$-MDDH assumptions, with $\mathcal{D}$ being a distribution of matrices in $\mathbb{Z}_M^{n_2 \times k_2}$ with $n_2 > k_2$, as long as the smallest prime factor p of M has at least $\mathfrak{K}$ bits, using Example 4.3.14. To our knowledge, our NIZK are the first constant-size NIZK for linear spaces over bilinear groups of composite order.

6.2.4 Application: Threshold Cramer-Shoup-like Encryption Scheme

As already explained in Section 2.2.2.3, the Cramer-Shoup encryption scheme [CS98] is one of the most efficient IND-CCA encryption schemes with a proof of security in the standard model. In this section, we use notation from Section 2.2.2.3. We remark that, if we replace the last part of a Cramer-Shoup ciphertext (i.e., the group element w) by a one-time simulation-sound NIZK proof π to prove that the tuple (u_1, u_2) is a DH tuple in basis (g_1, g_2), we can obtain an IND-CCA scheme supporting efficient threshold decryption. Intuitively, this comes from the

fact that the resulting scheme becomes "publicly verifiable", in the sense that, after verifying the NIZK (which is publicly verifiable), we can obtain the underlying message via "simple" algebraic operations which can easily be "distributed".

Previous one-time simulation-sound NIZK were quite inefficient and the resulting scheme would have been very inefficient compared to direct constructions of threshold IND-CCA encryption schemes. However, using our new one-time simulation-sound NIZK based on mixed pseudorandomness, we do not increase the size of the ciphertexts, in term of number of group elements. In addition, while we need to use an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, both the encryption and the decryption algorithms only do operations in the first group $\mathbb{G}_1$, and ciphertexts only contain group elements in $\mathbb{G}_1$.

Constructions. Here is the formal construction.

Construction 6.2.9. *We suppose that we have access to a one-time simulation-sound NIZK working in a bilinear asymmetric group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ for the DDH language in $\mathbb{G}_1$:*

$$\ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}} = \{(u_1, u_2) \in \mathbb{G}_2^2 \mid \exists r,\ (u_1, u_2) = (g_{1,1}^r,\ g_{1,2}^r)\} \quad \subseteq \quad \mathbb{G}_1^2 =: \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}}\,,$$

where $\ddot{\mathsf{lpar}} = (g_{1,1}, g_{1,2})$ is a tuple of two generators of $\mathbb{G}_1$. The scheme is defined as follows:

- $\mathsf{Setup.gpar}(1^{\mathfrak{K}})$ *generates an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of prime order p and picks a collision-resistant hash function $\mathcal{H}$ from a hash family $\mathcal{HF}$. It then outputs the global parameters $\mathsf{gpar} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathcal{H})$; we suppose that there exists an efficiently computable and efficiently reversible injective map $\mathcal{G}$ from the public set of messages $\mathscr{M}$ to the group $\mathbb{G}_1$;*

- $\mathsf{KeyGen}(\mathsf{gpar})$ *picks two generators $g_{1,1}, g_{1,2} \xleftarrow{\$} \mathbb{G}^*$, a random scalar $z \xleftarrow{\$} \mathbb{Z}_p$, and a CRS $\mathsf{crs} \xleftarrow{\$} \mathsf{NIZK.Setup}(\mathsf{gpar}, (g_{1,1}, g_{1,2}))$ for the NIZK. It sets the encryption key to $\mathsf{ek} := (g_{1,1}, g_{1,2}, \mathsf{crs})$ and the decryption key to $\mathsf{dk} := (z, \mathsf{crs})$. It finally outputs $(\mathsf{ek}, \mathsf{dk})$.*

- $\mathsf{Enc}(\ell, \mathsf{ek}, \mathsf{m})$ *computes $\mathsf{M} := \mathcal{G}(\mathsf{m})$, picks a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and outputs the ciphertext*
$$\mathsf{c} := (\ell,\ u_1 := g_1^r,\ u_2 := g_2^r,\ v := h^r \cdot \mathsf{M},\ \pi)\,,$$
where $\pi \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{tag}, (u_1, v_1), r)$ and $\mathsf{tag} := \mathcal{H}(\ell, u_1, u_2, v)$;

- $\mathsf{Dec}(\mathsf{dk}, \mathsf{c})$ *first computes $\mathsf{tag} = \mathcal{H}(\ell, u_1, u_2, v)$ and then checks the proof π. If this proof is valid ($\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, (u_1, u_2), \pi) = 1$), it computes $\mathsf{m} \leftarrow \mathcal{G}^{-1}(v/u_1^z)$ and outputs m. Otherwise, it outputs $\perp$.*

The scheme is clearly perfectly correct if the NIZK is perfectly complete. We show later that our scheme is IND-CCA if the NIZK is one-time simulation-sound and if DDH holds in the group $\mathbb{G}_1$.

The NIZK can be instantiated as in Section 6.2.3 (in prime order groups, under DDH in $\mathbb{G}_2$): using Construction 6.2.5 or Construction 6.2.7, with the second DM or DVS $\mathcal{M}_2$ being based on the DDH language (namely Example 3.1.8 and Example 5.1.5). The resulting scheme is secure under the *symmetric external Diffie-Hellman* (SXDH) assumption which just states that the DDH assumption holds in $\mathbb{G}_1$ and in $\mathbb{G}_2$.

Optimization of the decryption. The previous decryption procedure uses NIZK.Ver that needs to compute pairings. However, we can use a different key generation and decryption procedures to check the NIZK proof without pairing: KeyGen generates the CRS using $\mathsf{NIZK.Sim}_1$ and keep the trapdoor trap in the decryption key. Then Dec checks that $\mathsf{NIZK.Sim}_2(\mathsf{trap}, \mathsf{tag}, (u_1, u_2)) = \pi$ instead of $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{tag}, (u_1, u_2), \pi) = 1$. We recall that $\mathsf{NIZK.Sim}_2$ only performs operations in $\mathbb{G}_1$.

We need to prove that this change has no effect on the correctness nor on the IND-CCA property of the encryption scheme. Perfect correctness is directly implied by the fact that simulated proof using trap are always accepted by NIZK.Ver, thanks to Equation (3.10) on page 79.

For the IND-CCA property, let us show that no polynomial time adversary can generate a proof accepted by NIZK.Ver that is different from the one generated by $\mathsf{NIZK.Sim}_2$, even if it has access to everything (including the decryption key). Let us suppose that the adversary manage to generate a proof $\boldsymbol{\pi}^\intercal$ for some word χ and some tag tag, distinct from the proof $\boldsymbol{\pi}'^\intercal = \mathsf{NIZK.Sim}_2(\mathsf{trap}, \mathsf{tag}, \chi)$. As both satisfy Equation (3.9) on page 79, using notation in Construction 3.3.1, we have:[2]

$$(\boldsymbol{\pi}^\intercal - \boldsymbol{\pi}'^\intercal) \bullet \Gamma_2 = \mathbf{0}^\intercal \ .$$

In the case of Construction 6.2.7 with Example 5.1.5 as the second DVS, this is just impossible, as we have $\Gamma_2 = (g_2) \in \mathbb{G}_2^{1\times 1}$. In the case of Construction 6.2.5 with Example 3.1.8 as the second DVS, we have:

$$\Gamma_2 = \begin{pmatrix} g_2 \\ h_2 \end{pmatrix} \in \mathbb{G}_2^{2\times 1} \ ,$$

and the vector $\boldsymbol{\pi}^\intercal - \boldsymbol{\pi}'^\intercal \in \mathbb{G}_1^{1\times 2}$ can be used to solve the DDH problem in basis (g_2, h_2): a tuple $(u_2, v_2) \in \mathbb{G}_2^2$ is a DH tuple in basis (g_2, h_2) if and only

$$(\boldsymbol{\pi}^\intercal - \boldsymbol{\pi}'^\intercal) \bullet \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = 1_{\mathbb{G}_T} \ .$$

This concludes the adaptation of the proof of the IND-CCA property.

Threshold and structure-preserving properties. The validity of the ciphertext can be verified publicly, just knowing ek (or more precisely crs), and not dk, and then after this test has been performed, we just need to compute v/u_1^z, to get the message. We often say in this case that the ciphertext is "publicly verifiable", though it is not clear that a proper definition exists.

In any case, this property just means that to threshold decrypt the ciphertext, we just need to use Shamir's threshold secret sharing over $\mathbb{Z}_p$ [Sha79], exactly as in [SG02]. If in addition, we want to be able to verify decryption shares without random oracle, we can replace the Fiat-Shamir-based NIZK in [SG02] by one of ours.

Furthermore, our two schemes are *structure-preserving* [AFG+10]: they are "compatible" with Groth-Sahai NIZK [GS08], in the sense that we can do a Groth-Sahai NIZK to prove that we know the plaintext of a ciphertext for our encryption schemes.

[2] The constructions we use are like Construction 3.3.1, the only difference is the exact form of the two DVSs $\mathcal{V}_1$ and $\mathcal{V}_2$.

Comparison with existing schemes. A comparison with existing efficient IND-CCA encryption schemes based on cyclic or bilinear groups is given in Table 6.3, whose entries have been partially derived from similar tables in [BMW05; Kil06].

The two other efficient threshold and structure-preserving IND-CCA encryption schemes are those based on the Canetti-Halevi-Katz [CHK04] transform, the one of Boyen, Mei and Waters [BMW05] and the one of Kiltz [Kil06]. But for all except the one of Kiltz, the plaintext and one element of the ciphertext has to be in $\mathbb{G}_T$, which is not compatible with Groth-Sahai NIZK [GS08] (proving that we know the plaintext of a ciphertext cannot efficiently be done with Groth-Sahai NIZK for such encryption schemes). In addition, elements in $\mathbb{G}_T$ have a much longer representation than elements in $\mathbb{G}$, $\mathbb{G}_1$ or $\mathbb{G}_2$. And, even though our second encryption scheme uses exactly the same number of group elements as Kiltz's encryption scheme [Kil06], these groups elements are about 6 times smaller in practice (for $\mathfrak{K} = 128$ bits of security), since we use an asymmetric pairing while Kiltz's scheme uses a symmetric one (see Section 2.1.5). So even our first construction is more efficient (regarding ciphertext size) than Kiltz's construction. To summarize, to the best of our knowledge, our two constructions are the most efficient threshold and structure-preserving IND-CCA encryption schemes.

IND-CCA security proof. We have the following security theorem.

Theorem 6.2.10. *The encryption scheme of Construction 6.2.9 is IND-CCA, if the underlying NIZK is one-time simulation-sound, perfectly complete, and zero-knowledge, and if DDH holds in the group $\mathbb{G}_1$.*

Proof. The proof is quite straightforward and basically uses ideas in the security proof of the Cramer-Shoup encryption scheme [CS98]. Here is a sketch of a sequence of games proving the IND-CCA property:

Game $\mathbf{G}_0$: This is the game for $\mathsf{Exp}^{\texttt{ind-cca-}b}$ for $b = 0$ (see Definition 2.2.4).

Game $\mathbf{G}_1$: In this game, we generate $g_{1,2}$ as $g_{1,1}^t$ (with $t \xleftarrow{\$} \mathbb{Z}_p$) and reject all ciphertexts $\mathsf{c} = (u_1, u_2, v, \pi)$ submitted to the decryption oracle for which $u_2 \neq u_1^t$. This game is indistinguishable from the previous one under the soundness of the NIZK, which ensures that if the proof π is not rejected, $(g_{1,1}, g_{1,2}, u_1, u_2)$ is a DDH tuple.

Game $\mathbf{G}_2$: In this game, we generate h as $h = g_{1,1}^{z_1} g_{1,2}^{z_2}$ (with $z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$) instead of $h = g_{1,1}^{z}$. In addition, we decrypt ciphertexts $\mathsf{c} = (u_1, u_2, v, \pi)$ by first rejecting if π is not a valid proof or $u_2 \neq u_1^t$ (as before) and then outputting $v/(u_1^{z_1} u_2^{z_2})$ (instead of v/u_1^z). This game is perfectly indistinguishable from the previous one, because h can be written $h = g_{1,1}^{z_1+tz_2}$ and $u_1^{z_1} u_2^{z_2} = u_1^{z_1+tz_2}$.

Game $\mathbf{G}_3$: In this game, we do not check anymore that $u_2 = u_1^t$ and we generate $g_{1,2}$ directly as a random group element in $\mathbb{G}_1$. This game is indistinguishable from the previous one under the soundness of the NIZK.

Game $\mathbf{G}_4$: In this game, for the challenge ciphertext $\mathsf{c}^* = (u_1^*, u_2^*, v^*, \pi^*)$, we compute v^* as $v^* = u_1^{*z_1} u_2^{*z_2}$, instead of h^r, where $u_1 = g_{1,1}^r$ and $u_2 = g_{1,2}^r$. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_5$: In this game, we simulate the proof π^* in the challenge ciphertext $C^* = (u_1^*, u_2^*, v^*, \pi^*)$. This game is indistinguishable from the previous one under the zero-knowledge property of the NIZK. In addition, in this game, knowledge of r in c^* is no longer required.

Game $\mathbf{G}_6$: In this game, we replace (u_1^*, u_2^*) which was a DDH tuple in basis $(g_{1,1}, g_{1,2})$ by a random tuple. This game is indistinguishable from the previous one under the DDH assumption.

Game $\mathbf{G}_7$: In this game, we again generate $g_{1,2}$ as $g_{1,1}^t$ (with $t \xleftarrow{\$} \mathbb{Z}_p$) and reject all ciphertexts $\mathsf{c} = (u_1, u_2, v, \pi)$ submitted to the decryption oracle for which $u_2 \neq u_1^t$. This game is indistinguishable form the previous one under the one-time simulation-soundness of the NIZK.

Game $\mathbf{G}_8$: As in the proof of Cramer and Shoup [CS98], it is easy to show that the only information (from an information theoretic point of view) the adversary sees of z_1 and z_2 except from c^* is $z_1 + tz_2$. So $u_1^{*z_1} u_2^{*z_2}$ looks completely random to the adversary if (u_1, u_2) is not a DDH tuple in basis $(g_{1,1}, g_{1,2})$ (which happens with probability $1 - 1/p$). Therefore we can replace v^* by a random value, and this game is statistically indistinguishable from the previous one.

Finally, we can redo all the previous games in the reverse order and see that the experiment $\mathsf{Exp}^{\mathtt{ind\text{-}cca}\text{-}b}$ with $b = 0$ is indistinguishable from the experiment $\mathsf{Exp}^{\mathtt{ind\text{-}cca}\text{-}b}$ with $b = 1$.

□

Table 6.2: Comparison of NIZK for linear subspaces

		WS	Proof $\lvert\pi\rvert$	CRS $\lvert\mathsf{crs}\rvert$	Pairings
Instantiations based on DDH in $\mathbb{G}_2$					
Groth-Sahai [GS08]			$n+2k$	5	$2n(k+2)$
Jutla-Roy [JR13]		✓	$n-k$	$2k(n-k)+2$	$(n-k)(k+2)$
Jutla-Roy [JR14]		✓	1	$n+k+1$	$n+1$
Cons. 6.2.1			2	$n+2k+1$	$n+2$
Cons. 6.2.1.2		✓	1	$n+k$	$n+1$
Cons. 6.2.2.1 ($t=1$)	OTSS		2	$2(n+2k)+1$	$2n+2$
Cons. 6.2.2.2	OTSS	✓	1	$2\nu(2n+3k)$	$\nu n+2$
Instantiations based on DLin in $\mathbb{G}_2$					
Groth-Sahai [GS08]			$2n+3k$	6	$3n(k+3)$
Jutla-Roy [JR13]			$2n-2k$	$4k(n-k)+3$	$2(n-k)(k+2)$
Libert et al. [LPJY14]			3	$2n+3k+3$	$2n+4$
Libert et al. [LPJY14]	RSS		4	$4n+8t+5$	$2n+6$
Jutla-Roy [JR14]		✓	2	$2(n+k+2)$	$2(n+2)$
Cons. 6.2.1.1			3	$2n+3k+2$	$2n+3$
Cons. 6.2.1.2		✓	2	$2n+2k+1$	$2n+2$
Cons. 6.2.2.1 ($t=1$)	OTSS		3	$2(2n+3k)+2$	$4n+3$
Cons. 6.2.2.2	OTSS	✓	2	$2\nu(2n+3k)+1$	$2\nu n+2$

- $n=n_1$, $k=k_1$, and ν is the length of the tag tag; pairings: number of pairings required to verify the proof;
- sizes $\lvert\cdot\rvert$ are measured in term of group elements ($\mathbb{G}_1$ and $\mathbb{G}_2$, or $\mathbb{G}$ if the bilinear group is symmetric). Generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ (for DDH in $\mathbb{G}_2$) or $g \in \mathbb{G}$ (for DLin) are not counted in the CRS;
- *OTSS*: one-time simulation-soundness; *RSS*: single-theorem relative simulation-soundness [JR12] (weaker than OTSS);
- WS: witness samplability, i.e., requirement that $\mathsf{Setup.lpar_1}$ generates a trapdoor $\mathsf{ltrap_1}$ allowing us to compute the discrete logarithms of Γ_1. This slightly restricts the set of languages which can be handled.

Table 6.3: Comparison of IND-CCA encryption schemes over cyclic and bilinear groups

		Time Complexity[a]			Ciphertext Overhead		
Scheme	Assumption	Encryption	Decryption	Public key	Hybrid	SP[b]	Th.[c]
KD	DDH	[1,2,0]	0+[1,0,0]	4 $\mathbb{G}$	2 $\mathbb{G}$ (+dem)	n/a	✓
CS	DDH	[1,3,0]	0+[1,1,0]	5 $\mathbb{G}$	3 $\mathbb{G}$	+ $\mathbb{G}$	
CHK/BB1	BDDH	[1,2,0]	1+[1,0,0]	$O(1)$[d]	2 $\mathbb{G}$ + sig	+ $\mathbb{G}_T$	✓
CHK/BB2	q-BDDHI	[1,2,0]	1+[0,1,1]	$O(1)$[d]	2 $\mathbb{G}$ + sig	+ $\mathbb{G}_T$	✓
BK/BB1	BDDH	[1,2,0]	1+[1,0,0]	$O(1)$[d]	2 $\mathbb{G}$ + cm	+ $\mathbb{G}_T$	
BK/BB2	q-BDDHI	[1,2,0]	1+[0,1,1]	$O(1)$[d]	2 $\mathbb{G}$ + cm	+ $\mathbb{G}_T$	
BMW	BDDH	[1,2,0]	1+[0,1,0]	2 $\mathbb{G}$ + $\mathbb{G}_T$	2 $\mathbb{G}$	+ $\mathbb{G}_T$	✓
Kiltz	DLin	[2,3,0]	0+[1,0,0]	5 $\mathbb{G}$	4 $\mathbb{G}$[e]	+ $\mathbb{G}$[e]	✓
Ours 1	SXDH	[2,3,0]	0+[2,1,0]	6 $\mathbb{G}_1$	4 $\mathbb{G}_1$	+ $\mathbb{G}_1$	✓
Ours 2	SXDH	[0,4,0]+2$\mathfrak{K}$	0+[0,2,0]+2$\mathfrak{K}$	(3+4$\mathfrak{K}$) $\mathbb{G}_1$	3 $\mathbb{G}_1$	+ $\mathbb{G}_1$	✓

- Ours 1: Construction 6.2.9 with Example 3.1.8;
- Ours 2: Construction 6.2.7 with Example 5.1.5;
- KD: Kurosawa-Desmedt [KD04], CS: Cramer-Shoup [CS98], CHK: Canetti-Halevi-Katz transform [CHK04] for BB1/BB2 Boneh-Boyen IBE [BB04], BK: Boneh-Katz transformation [BK05], BMW: Boneh-Mey-Waters [BMW05], Kiltz [Kil06]
- dem: data encapsulation mechanism; sig: verification key of a one-time signature scheme + signature; cm: commitment + mac (message authentication code)

[a] (#pairing +) [#multi, #regular, #fix]-exponentiation (+ #multiplication) (in $\mathbb{G}$ or $\mathbb{G}_1$), a multi-exponentiation being a computation of the form $a_1^{b_1} \cdots a_k^{b_k}$, where $a_1, \ldots, a_k \in \mathbb{G}$ and $b_1, \ldots, b_k \in \mathbb{Z}_p$; the number of multiplications is approximate and only written when it depends on $\mathfrak{K}$, since multiplications are way faster than pairings and exponentiations;

[b] number of other elements for the DEM part (data encapsulation mechanism — basically the part containing the message) to make the scheme, a structure-preserving encryption scheme; see text; concretely, in our scheme, the DEM part is $v = h \cdot \mathsf{M}$;

[c] supports threshold decryption;

[d] depends on parameters for the signature/commitment/mac and if we use symmetric or asymmetric groups, but a small constant in any case;

[e] $\mathbb{G}$ has to be a cyclic group from a symmetric bilinear group, and so element representation is often 50% bigger than for the other scheme where $\mathbb{G}$ is either just a cyclic group, or can be the first group ($\mathbb{G}_1$) of an asymmetric bilinear group;

[f] supposing $\nu = 2\mathfrak{K}$.

6.3 Trapdoor Smooth Projective Hashing and Implicit Zero-Knowledge

While SPHFs and universal PHFs enable to construct efficient honest-verifier zero-knowledge arguments for languages for which there exists a DVS or a DM, they do not provide zero-knowledge (see Sections 2.5.3.1 and 6.1.2.4). More generally, from a high-level point of view, SPHFs and universal PHFs do not have a zero-knowledge flavor, but can only be seen as honest-verifier zero-knowledge: nothing is guaranteed when the projection key hp is maliciously generated. In security reductions, we do not have any trapdoor enabling to compute the hash value of a word $\chi \in \mathscr{L}$, for a projection key hp without knowing a witness w for this word χ nor the associated hashing key.

That is why, we now introduce two new primitives: *trapdoor smooth projective hash functions* (TSPHFs) and *implicit zero-knowledge arguments* (iZK).

6.3.1 Overview

Before diving into technical details, let us informally introduce TSPHFs and iZK, some of their applications, and the high level ideas for their constructions.

6.3.1.1 Trapdoor Smooth Projective Hash Functions (TSPHFs)

Overview and applications. TSPHFs are basically SPHFs with a CRS, such that there is a way to generate the CRS with an additional trapdoor enabling to compute the hash value of any word $\ddot{\chi} \in \ddot{\mathscr{L}}$ for any (potentially maliciously generated) projection key hp, without knowing the associated hashing key. They were initially introduced in [BBC+13c] to construct the most efficient (at the time) one-round PAKE in the UC model with static corruptions. They can also replace universal PHFs in all the constructions in Section 6.1, to get zero-knowledge instead of just honest-verifier zero-knowledge. Furthermore, they can replace SPHFs in the secret agent application in Section 2.5.3.3 to get security against a malicious informer.

Constructions. A naive way to construct a TSPHF is just to add to the projection key hp, an extractable NIZK proving the knowledge of the hashing key hk. However, our constructions of TSPHFs are much more efficient than this naive construction.

Let us give the intuition behind our constructions in the prime order case. Our construction uses the classical "or trick": we consider the disjunction of a DVS for the language $\ddot{\mathscr{L}} = \mathscr{L}_1$ for the TSPHF, with a DVS for a hard-subset-membership language $\mathscr{L}_2$. The CRS contains a word $\chi_2 \in \mathcal{X}_2$, and the hash value of a word $\ddot{\chi} = \chi_1$ is the hash value of the word (χ_1, χ_2).

On the one hand, if $\chi_2 \in \mathscr{L}_2$ and we know a witness w_2 for this fact, we can compute the hash value of any word $\ddot{\chi}$ using this trapdoor w_2. That way we get our zero-knowledge-like property. On the other hand, if $\chi_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$, and if $\chi_1 \notin \mathscr{L}_1$ the hash value (χ_1, χ_2) looks uniformly random given only the projection key, and we still get smoothness. We remark that the two settings ($\chi_2 \in \mathscr{L}_2$ and $\chi_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$) are computationally indistinguishable under hard subset membership of $\mathscr{L}_2$.

As already remarked in Section 6.1.2.4, there might be an issue with this "or trick": it might be possible that for some maliciously generated projection keys hp, the projected hash value computed using a witness w_2 is different from the projected hash value computed using

Table 6.4: Duality of NIZK and TSPHF constructed from the disjunction of DMs

	NIZK	TSPHF
CRS for soundness/smoothness	hp	$x_2 \in \mathcal{X}_2 \setminus \mathcal{L}_2$
CRS for zero-knowledge	hp	$x_2 \in \mathcal{L}_2$
Trapdoor	hk	w_2
Verification	public using CRS	private using hk
Projection key	honestly generated	potentially malicious
Proof / hash value	in $\mathfrak{G}_1$	in $\mathfrak{G}$

the real witness w_1. This would break the zero-knowledge property. Fortunately, if we use CS/KV disjunctions of DVSs (Construction 3.2.8), we can easily check the validity of the projection key.

Therefore, all our constructions of TSPHFs use the CS/KV disjunction of the two DVSs $\mathcal{V}_1$ and $\mathcal{V}_2$, and thus require that these two DVSs are over two multiplicatively compatible sub-graded rings of some graded ring $\mathfrak{G}$. Concretely, this often means that we have a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ and that the original language $\dot{\mathcal{L}} = \mathcal{L}_1$ is over $\mathbb{G}_1$, while the hard-subset-membership language $\mathcal{L}_2$ is over $\mathbb{G}_2$.

Finally, we can either extend our construction to DMs or extend it to use a Pr-DVS as second DVS $\mathcal{V}_2$, similarly to what we did in Section 6.2 for constructions of NIZK.

Comparison with the constructions of NIZK. We remark that both our constructions of NIZK and TSPHFs use a CS/KV disjunction of the language we are interested in, with a hard-subset-membership language (or a Pr-DVS). The use of this second language is however very different in both cases and can be seen as dual from each other.

In NIZK, the second language is used to add public verifiability of the proof, as checking the hash value of a PHF requires the knowledge of the hashing key, which has to be kept secret to ensure soundness (from smoothness). The CRS is a projection key and the hashing key is its trapdoor. In particular, the projection key is always correctly generated. Furthermore the proof of a NIZK is completely in the first sub-graded ring $\mathfrak{G}_1$.

In TSPHFs, we do not try to achieve public verifiability: only the user who generated the hashing key has the ability to check hash values without knowing a witness (if we forget about the trapdoor). The projection key can be maliciously generated. The CRS contains a word of the second language $\mathcal{L}_2$ and the trapdoor is its witness.

Table 6.4 summarizes these differences.

6.3.1.2 Implicit Zero-Knowledge Arguments (iZK)

Overview. One drawback of TSPHFs is that they necessarily require two multiplicatively compatible sub-graded rings: one for the real language and one to add the zero-knowledge property. Concretely, this means that we need bilinear groups. However, as explained in Section 2.1.5, cyclic groups without pairings are likely to be safer and have faster group operations. We might think that if we allow for a *trapdoor GL-smooth projective hash function* (GL-TSPHF), we could replace the CS/KV disjunction (Construction 3.2.8 which is the reason why we need pairings) in the TSPHF construction by a GL disjunction (Construction 3.2.4 which can work in a cyclic group). Unfortunately, we do not know any way (not requiring

pairings) to efficiently check the validity of a projection key for a GL disjunction non-interactively. And there really is an attack shown in Section 6.1.2.4, if we cannot do this check.

That is why in [BCPW15], we introduced iZK which can be seen as generalizations of TSPHFs, in which the user who has to generate the hash value (called the *prover*) is allowed to first send a message to the user who generates the projection key (called the *verifier*).

To formalize the notion of iZK, we use an extension of the definition of witness encryption [GGSW13], rather than an extension of the definition of SPHFs. We recall that GL-SPHFs are basically equivalent to statistically sound witness encryption schemes (see Section 2.5.3.2).

More precisely, we see iZK as key encapsulation mechanisms in which the public key iek (corresponding to this above additional message) is associated with a word $\ddot{\mathfrak{x}}$. The projection key is part of what is called a ciphertext c, while the hash value is part of the associated plaintext (a.k.a., *ephemeral key*) K. A prover knowing a witness $\ddot{w}$ for $\ddot{\mathfrak{x}}$ can generate an encryption key iek together with its associated decryption key idk. The latter enables him to decrypt any ciphertext c for iek and $\ddot{\mathfrak{x}}$.

Intuitively, we want that if $\ddot{\mathfrak{x}} \notin \ddot{\mathscr{L}}$, a polynomial-time adversary cannot generate a public key iek, such that it can distinguish the plaintext K corresponding to a ciphertext c for this public key iek and this word $\ddot{\mathfrak{x}}$, from a uniform plaintext K. This property is called soundness, and basically corresponds to a computational version of smoothness.

We also want that there is a CRS and there is a way to generate it with a trapdoor enabling to produce fake keys iek for any word $\ddot{\mathfrak{x}} \in \ddot{\mathscr{L}}$, together with fake decryption keys itk also enabling to decrypt ciphertexts c for iek and $\ddot{\mathfrak{x}}$.

Construction. We focus on the prime-order setting.

We consider the GL disjunction DVS $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ of a DVS for the language $\ddot{\mathscr{L}} = \mathscr{L}_1$ we consider, with a DVS for a hard-subset-membership language $\mathscr{L}_2$. We suppose that $\mathcal{V}_1$ does not use ρ, for this overview. The CRS crs contains a word $\mathfrak{x}_2 \in \mathcal{X}_2$: in the soundness setting, $\mathfrak{x}_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$, while in the zero-knowledge setting, $\mathfrak{x}_2 \in \mathscr{L}_2$ and the trapdoor of the CRS is $\mathsf{trap} = w_2$, a witness for $\mathfrak{x}_2$. We recall that we need to find a way to check the validity of a projection key for $\mathcal{V}$, using the public key iek.

The first idea is the following: a valid projection key hp is a row vector of the form $\mathsf{hp} = \boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma \in \mathfrak{G}^{1\times k}$ for some row vector $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times n}$. The language of the valid projection keys can therefore be seen as a DVS where the functions "$\boldsymbol{\theta}$" and "$\boldsymbol{\lambda}$" are the identity and the matrix "Γ" is the transpose of Γ. We can then try to use the SPHF associated to this transposed DVS to check the validity of hp: the secret key idk contains a uniform hashing key $\mathsf{tk} := \boldsymbol{\alpha}'^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$, while the public key iek contains the associated projection key $\mathsf{tp} := \boldsymbol{\gamma}'^\intercal := \boldsymbol{\alpha}'^\intercal \bullet \Gamma^\intercal \in \mathfrak{G}^{1\times n}$.

The verifier then constructs the ciphertext c as a pair of a projection key $\mathsf{hp} = \boldsymbol{\gamma}^\intercal$ for the original DVS $\mathcal{V}$ (with $\mathsf{hk} := \boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}$ a random hashing key) and its associated hash value under tp: $\mathsf{tpH} := \boldsymbol{\gamma}'^\intercal \bullet \boldsymbol{\alpha}$. The associated plaintext K is just the hash value of the word $\mathfrak{x}$: $\mathsf{K} := \mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(\mathfrak{x}) \in \mathfrak{G}$, where $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2)$.

To decrypt a ciphertext $\mathsf{c} = (\mathsf{hp} = \boldsymbol{\gamma}^\intercal, \mathsf{tpH})$, the prover first checks the validity of hp by checking that $\mathsf{tpH} \stackrel{?}{=} \mathsf{tH} =: \boldsymbol{\alpha}'^\intercal \bullet \boldsymbol{\gamma}$, and aborts if it is the case. Then, he computes the plaintext as $\mathsf{K} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(\mathfrak{x}, (w_1, \perp))$, with w_1 a witness of $\mathfrak{x}_1$.

It is easy to see that the construction is zero-knowledge, as for TSPHFs, since now the verifier cannot cheat on the projection key. Unfortunately, soundness (or smoothness) does

not hold anymore: the prover can just choose $\mathsf{tp} = \boldsymbol{\gamma}'^{\mathsf{T}} = \boldsymbol{\theta}(\mathcal{X})$ and then

$$\mathsf{tpH} = \boldsymbol{\gamma}'^{\mathsf{T}} \bullet \boldsymbol{\alpha} = \boldsymbol{\alpha}^{\mathsf{T}} \bullet \boldsymbol{\theta}(\mathcal{X}) = \mathsf{H} = \mathsf{K} \ .$$

In [BCPW15], we solved this issue by replacing $\mathcal{M}$ by its 2-sound extension (Construction 4.4.15 extended to the GL case) and by making the verifier chooses the tag tag as part of the ciphertext. Since the prover does not know the tag tag when he generates iek and tp, the previous attack does not work anymore. This technique works but approximately doubles the size of c, iek, and idk, as it doubles the number of rows and of columns of the matrix Γ. In this thesis, we propose a new solution which is much more efficient: we remove tpH from c and instead add a random scalar $\zeta \xleftarrow{\$} \mathbb{Z}_p$ in c. And the plaintext is now $\mathsf{K} = (\zeta \bullet \mathsf{H}) \boldsymbol{+} \mathsf{tpH}$.

The idea is that, for any vector $\boldsymbol{\gamma}'$, there is only at most one value $\zeta \in \mathbb{Z}_p$ such that $\zeta \bullet \boldsymbol{\theta}(\mathcal{X}) \boldsymbol{+} \boldsymbol{\gamma}' \in \mathrm{ColSpan}\,(\Gamma)$, if $\boldsymbol{\theta} \notin \mathrm{ColSpan}\,(\Gamma)$. By contradiction, if there were two distinct such scalars ζ, denoted by ζ_1 and ζ_2, we would have:

$$(\zeta_1 - \zeta_2) \bullet \boldsymbol{\theta} \in \mathrm{ColSpan}\,(\Gamma) \ .$$

As with TSPHFs, we can extend everything to composite order, or use a Pr-DVS as second DVS.

Applications. As TSPHFs, iZK can be used to replace universal PHFs in the constructions in Section 6.1 to get zero-knowledge, and also in the protocol for secret agents in Section 2.5.3.3 to get security against a malicious informer. This increases the number of rounds in some cases.

A more interesting application is the original one proposed in [BCPW15]: enforcing semi-honest behavior. Let us suppose that we have a two-party protocol secure against semi-honest adversaries. If we want to transform it into a protocol secure against malicious adversaries, we can follow the idea of Goldreich, Micali, and Wigderson in their seminal work [GMW87]: each user proves that he generated correctly each of his flows using a zero-knowledge argument.

We notice that when a user Alice sends a flow to the other user Bob, Bob cannot send back the next flow until he is sure that Alice's flow has been generated correctly. Otherwise, he might reveal too much information. Therefore, if we use interactive zero-knowledge arguments, we will at least multiply the number of rounds of the initial protocol by three. We can instead use non-interactive zero-knowledge arguments (NIZK), but this either requires strong assumption such as random oracles [BR93; FS87], or bilinear groups [GS08], or is completely inefficient. We could also use preprocessing NIZK, but known constructions [DMP90] are also very inefficient.

On the other hand, iZK can be based on the DDH assumption in any cyclic group and provide a lightweight alternative to NIZK. It is based on the following idea: in the previous transform, Bob does not really need to know whether Alice generated correctly her flow or not, he just needs to be sure that she does not learn anything about his next flow if her flow was invalid. He can therefore use an iZK to encrypt his flow in such a way Alice will be able to decrypt it only if she generated her flow correctly.

More precisely, to ensure semi-honest behavior, as depicted in Figure 6.3, each time Alice sends a flow $\ddot{\mathcal{X}}$, she also sends a public key iek and keeps the associated secret key idk. To answer back, Bob generates a key encapsulation c for iek and $\ddot{\mathcal{X}}$, of a random ephemeral key K. Bob then use K to encrypt (using symmetric encryption or pseudo-random generators and one-time pad) all the subsequent flows he sends to Alice.

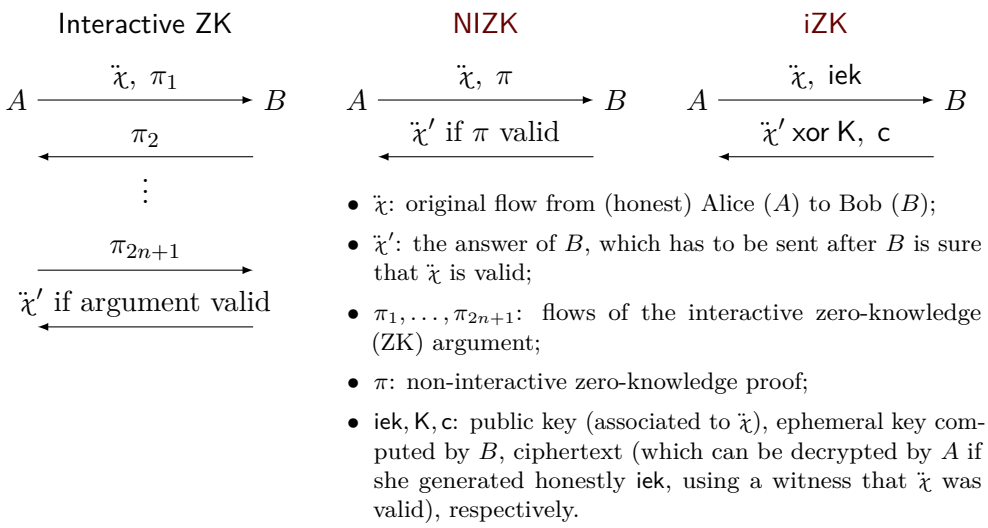

Figure 6.3: Enforcing semi-honest behavior of Alice (A) by Bob (B)

At the end, both users still needs to be check whether their last flows were valid, to know whether the protocol succeeded or not. This can be done using a classical zero-knowledge argument.

While the previous transformation with iZK is completely generic (see [BCPW15]), on "non-algebraic" protocols, it is often not the best solution. For example, for Yao-based protocols, cut-and-choose methods are more efficient [IKLP06; LP07; LP11; sS11; sS13; Lin13; HKE13].

But, when the original semi-honest protocol is an algebraic protocol over some cyclic group $(p, \mathbb{G}, g)$ and all the languages we need to consider can be handled by DVSs, we get very efficient protocols. In [BCPW15], we give a concrete example of such protocol, to compute the Hamming distance of two strings. The resulting protocol outperforms the state of the art.

6.3.2 Trapdoor Smooth Projective Hash Functions (TSPHFs)

6.3.2.1 Definition

Let us formally define *trapdoor smooth projective hash function* (TSPHF). As for SPHFs, there are three variants: *trapdoor GL-smooth projective hash function* (GL-TSPHF), *trapdoor CS-smooth projective hash function* (CS-TSPHF), and *trapdoor KV-smooth projective hash function* (KV-TSPHF).

Definition 6.3.1 (GL-TSPHF)**.** *A GL-TSPHF (a.k.a, TSPHF) for a language* $(\ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}})$ *is defined by a tuple of eight polynomial-time algorithms* (T.Setup, T.TSetup, HashKG, ProjKG, Hash, ProjHash, VerHP, THash)*:*

- T.Setup(gpar, $\ddot{\mathsf{lpar}}$) *outputs a CRS* crs *for the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$*; as usual, we suppose that* crs *implicitly contains* gpar *and* $\ddot{\mathsf{lpar}}$*;*
- T.TSetup(gpar, $\ddot{\mathsf{lpar}}$) *generates the (simulated) common reference string* crs *together with a trapdoor* trap *for the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$*, and outputs the pair* (crs, trap)*;*

- $\mathsf{HashKG}(\mathsf{crs}, \ddot{\mathsf{lpar}})$ *generates a hashing key* hk *for the language parameters* $\ddot{\mathsf{lpar}}$*;*
- $\mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi})$ *deterministically derives a projection key* hp *from the hashing key* hk*, the language parameters* $\ddot{\mathsf{lpar}}$*, and possibly the word* $\ddot{\chi} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}}$*;*
- $\mathsf{Hash}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi})$ *deterministically outputs a hash value* H *from the hashing key* hk*, for the word* $\ddot{\chi} \in \mathcal{X}_{\mathsf{lpar}}$ *and the language parameters* $\ddot{\mathsf{lpar}}$*;*
- $\mathsf{ProjHash}(\mathsf{crs}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi}, \ddot{w})$ *deterministically outputs a projected hash value* pH *from the projection key* hp*, and the witness* $\ddot{w}$*, for the word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ *(i.e.,* $\ddot{\mathscr{R}}_{\ddot{\mathsf{lpar}}}(\ddot{\chi}, \ddot{w}) = 1$*) and the language parameters* $\ddot{\mathsf{lpar}}$*;*
- $\mathsf{THash}(\mathsf{crs}, \mathsf{trap}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi})$ *deterministically outputs a (trapdoor) hash value* trapH *from the projection key* hp*, and the trapdoor* trap*, for the word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ *and the language parameters* $\ddot{\mathsf{lpar}}$*;*
- $\mathsf{VerHP}(\mathsf{crs}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi})$ *outputs* 1 *if* hp *is a valid projection key for the language parameters* $\ddot{\mathsf{lpar}}$ *and the word* $\ddot{\chi}$*, and* 0 *otherwise.*

As for SPHF, the set of hash values is called the range *and is denoted by* Π*. A GL-TSPHF has to satisfy the following properties:*

- Perfect correctness. *For any* $\ddot{\mathsf{lpar}}$*:*
  - *for any* crs *generated by* $\mathsf{crs} \xleftarrow{\$} \mathsf{T.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ *or generated by* $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{T.TSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$*, for any word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ *with witness* $\ddot{w}$ *(i.e., such that* $\mathscr{R}_{\mathsf{lpar}}(\ddot{\chi}, \ddot{w}) = 1$*), for any* $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}, \ddot{\mathsf{lpar}})$ *and for any* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}})$*, we have:*

$$\mathsf{Hash}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi}) = \mathsf{ProjHash}(\mathsf{crs}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi}, \ddot{w})\,;$$

  - *for any* $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{T.TSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$*, for any word* $\ddot{\chi} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}}$*, for any* $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}, \ddot{\mathsf{lpar}})$ *and for* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}})$*, we have:*

$$\mathsf{Hash}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi}) = \mathsf{THash}(\mathsf{crs}, \mathsf{trap}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi})\,;$$

- Reference string indistinguishability (a.k.a, setup indistinguishability). *A polynomial-time adversary cannot distinguish a normal CRS generated by* iSetup *from a simulated CRS generated by* $\mathsf{iTSetup}$*. Formally, the advantage of an adversary* $\mathcal{A}$ *against reference string indistinguishability is defined by the experiments* $\mathsf{Exp}^{\mathtt{crs\text{-}ind\text{-}b}}$ *depicted in Figure 6.4. The TSPHF is* reference-string-indistinguishable*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*;*
- Computational GL/CS smoothness. *When the CRS* crs *is generated by* $\mathsf{T.Setup}$*, for any word* $\ddot{\chi} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}} \setminus \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$*, a polynomial-time adversary cannot distinguish a hash value (computed using the hashing key* hk*) from a uniform value in* Π*. Formally, the advantage of an adversary* $\mathcal{A}$ *against computational smoothness is defined by the experiments* $\mathsf{Exp}^{\mathtt{gl/cs\text{-}smooth\text{-}b}}$ *depicted in Figure 6.4. The TSPHF is* computationally GL/CS-smooth*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*;*

- Zero-knowledge. *When the CRS* crs *is generated by* T.TSetup*, for any word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ *with witness* $\ddot{w}$*, the trapdoor hash value* trapH *is indistinguishable from the projected hash value* pH*. Formally, the advantage of an adversary* $\mathcal{A}$ *against zero-knowledge is defined by the experiments* $\mathsf{Exp}^{zk\text{-}b}$ *depicted in Figure 6.4. The iZK is* zero-knowledge*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*.*

Remark 6.3.2 (Computational smoothness vs smoothness)**.** *We remark that if the SPHF part of a TSPHF is smooth for any* $\mathsf{crs} \xleftarrow{\$} \mathsf{T.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$*, then the TSPHF is computationally smooth, and computational smoothness even holds for any (potentially unbounded) adversary. The converse is not true, as smoothness has to hold for any* gpar *and any* $\ddot{\mathsf{lpar}}$*. This minor difference is just historical, and we do not know any application where it has an impact.*

A CS-TSPHF is a GL-TSPHF such that the algorithm ProjKG does not use its input $\ddot{\chi}$. Let us now define KV-TSPHFs.

Definition 6.3.3 (KV-TSPHF)**.** *A KV-TSPHF is a CS-TSPHF where computational GL/CS smoothness is replaced by the following property:*

- Computational KV smoothness. *Computational KV smoothness is similar to computational GL/CS smoothness except that the word* $\ddot{\chi}$ *can be adaptively chosen by the adversary after being given the projection key* hp*. Formally, the advantage of an adversary* $\mathcal{A}$ *against computational smoothness is defined by the experiments* $\mathsf{Exp}^{kv\text{-}smooth\text{-}b}$ *depicted in Figure 6.4. The TSPHF is* computationally KV-smooth*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*;*

Historical note 6.3.4. *Our definition of TSPHF is slightly different than the original one in [BBC+13c]. The main difference is the separation between the setup* T.Setup *from the setup* T.TSetup*. This enables to get statistical soundness in our first construction and to avoid requiring to have a trapdoor* $\ddot{\mathsf{ltrap}}$ *enabling to check in polynomial time whether a word* $\ddot{\chi}$ *is in* $\ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$*. We also have renamed the original "soundness" property to "zero-knowledge", and slightly changed it to make it more natural.*

6.3.2.2 Construction Based on Disjunctions with a Hard-Subset-Membership Language

Prime-order setting. Let us start by the construction in the prime-order setting, which was introduced in [ABP15c].

Construction 6.3.5 (TSPHF from disjunctions of DVSs)**.** *We suppose that we have a DVS* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *for* $\ddot{\mathscr{L}} = \mathscr{L}_1$*, together with a KV-DVS* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *for a hard-subset-membership language* $\mathscr{L}_2$*, such that* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *are sub-graded rings of some graded ring* $\mathfrak{G}$*. Furthermore, we consider the KV disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*. We recall that* $n = n_1 n_2$ *and* $k = k_1 n_2 + n_1 k_2$*.*

We construct a NIZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *as follows:*

- $\mathsf{T.Setup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with a word* $x_2 \xleftarrow{\$} \mathcal{X}_{2,\mathsf{lpar}_2} \setminus \mathscr{L}_{2,\mathsf{lpar}_2}$*. It then outputs* $\mathsf{crs} := (\mathsf{lpar}_2, x_2)$*.*

- $\mathsf{T.TSetup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar}_2(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with a word and its witness:* $(x_2, w_2) \xleftarrow{\$} \mathscr{R}_{2,\mathsf{lpar}_2}$*. It then output* $\mathsf{crs} := (\mathsf{lpar}_2, x_2)$ *and* $\mathsf{trap} := w_2$*.*

Reference String Indistinguishability

$\mathsf{Exp}^{\texttt{crs-ind-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- **if** $b = 0$ **then**
  - $\mathsf{crs} \xleftarrow{\$} \mathsf{T.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **else**
  - $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{T.TSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$

Zero-Knowledge

$\mathsf{Exp}^{\texttt{zk-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{T.TSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $(\ddot{\chi}, \ddot{w}, \mathsf{hp}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathsf{trap}, \ddot{\mathsf{ltrap}})$
- **if** $\mathscr{R}_{\ddot{\mathsf{lpar}}}(\ddot{\chi}, \ddot{w}) = 0$ **then**
  - **return** 0
- **if** $\mathsf{VerHP}(\mathsf{crs}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi}) = 0$ **then**
  - **return** 0
- **if** $b = 0$ **then**
  - $\mathsf{H} \leftarrow \mathsf{ProjHash}(\mathsf{crs}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi}, \ddot{w})$
- **else**
  - $\mathsf{H} \leftarrow \mathsf{THash}(\mathsf{crs}, \mathsf{trap}, \mathsf{hp}, \ddot{\mathsf{lpar}}, \ddot{\chi})$
- **return** $\mathcal{A}(\mathsf{st}, \mathsf{H})$

Computational smoothness

$\mathsf{Exp}^{\texttt{gl/cs-smooth-}b}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Exp}^{\texttt{kv-smooth-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{T.Setup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}, \ddot{\mathsf{lpar}})$
- $\mathsf{hp} \leftarrow \perp$ ▷ only in $\mathsf{Exp}^{\texttt{gl/cs-smooth-}b}$
- $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}})$ ▷ only in $\mathsf{Exp}^{\texttt{kv-smooth-}b}$
- $(\ddot{\chi}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}}, \mathsf{hp})$
- $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi})$ ▷ only in $\mathsf{Exp}^{\texttt{gl/cs-smooth-}b}$
- **if** $b = 1$ or $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ **then**
  - $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{crs}, \mathsf{hk}, \ddot{\mathsf{lpar}}, \ddot{\chi})$
- **else**
  - $\mathsf{H} \xleftarrow{\$} \Pi$
- **return** $\mathcal{A}(\mathsf{st}, \mathsf{hp}, \mathsf{H})$

Figure 6.4: Experiments for Definitions 6.3.1 and 6.3.3 (TSPHF)

- $\mathsf{HashKG}(\mathsf{crs}, \mathsf{lpar}_1)$ *generates and outputs a hashing key for* $\mathcal{V}$*:*

$$\mathsf{hk} := (\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n},\ \rho \xleftarrow{\$} \mathsf{RGen}(\mathsf{lpar}))\,,$$

  where $\mathsf{lpar} := (\mathsf{lpar}_1, \mathsf{lpar}_2)$.

- $\mathsf{ProjKG}(\mathsf{crs}, \mathsf{hk}, \mathsf{lpar}_1, x_1)$ *outputs the projection key* $\mathsf{hp} := (\boldsymbol{\gamma}^\intercal, \rho)$ *where:*

$$\boldsymbol{\gamma}^\intercal := \boldsymbol{\alpha}^\intercal \bullet \Gamma_{\mathsf{lpar}}(x, \rho) \in \mathfrak{G}^{1\times k}\,,$$

  where $x := (x_1, x_2)$*. In the sequel, we split* $\boldsymbol{\gamma}^\intercal$ *in two parts:*

$$\begin{aligned}\boldsymbol{\gamma}_1^\intercal &:= (\gamma_i)_{i=1,\ldots,k_1n_2}^\intercal = \boldsymbol{\alpha}^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{n_2}) \quad \in \mathfrak{G}_1^{1\times(k_1n_2)}\,,\\ \boldsymbol{\gamma}_2^\intercal &:= (\gamma_i)_{i=k_1n_2+1,\ldots,n}^\intercal = \boldsymbol{\alpha}^\intercal \bullet (\mathsf{Id}_{n_1} \otimes \Gamma_2) \quad \in \mathfrak{G}_2^{1\times(n_1k_2)}\,;\end{aligned}$$

- $\mathsf{Hash}(\mathsf{crs}, \mathsf{hk}, \mathsf{lpar}_1, x_1)$ *outputs the hash value*

$$\mathsf{H} := \boldsymbol{\alpha}^\intercal \bullet \boldsymbol{\theta}(x, \rho) \quad \in \mathfrak{G}\,;$$

- $\mathsf{ProjHash}(\mathsf{crs}, \mathsf{hp}, \mathsf{lpar}_1, x_1, w_1)$ *outputs the projected hash value*

$$\mathsf{pH} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(x, (w_1, \perp), \rho) = \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1(x_1, w_1, \rho) \otimes \boldsymbol{\theta}_2(x_2, \rho)) \quad \in \mathfrak{G}\,;$$

- $\mathsf{THash}(\mathsf{crs}, \mathsf{trap}, \mathsf{hp}, \mathsf{lpar}_1, x_1)$ *outputs the trapdoor hash value*

$$\mathsf{trapH} := \boldsymbol{\gamma}^\intercal \bullet \boldsymbol{\lambda}(x, (\perp, w_2), \rho) = \boldsymbol{\gamma}_2^\intercal \bullet (\boldsymbol{\theta}_1(x_1, \rho) \otimes \boldsymbol{\lambda}_2(x_2, w_2, \rho)) \quad \in \mathfrak{G}\,;$$

- $\mathsf{VerHP}(\mathsf{crs}, \mathsf{hp}, \mathsf{lpar}_1, x_1)$ *checks the following equation:*

$$\boldsymbol{\gamma}_1^\intercal \bullet (\mathsf{Id}_{k_1} \otimes \Gamma_2) \stackrel{?}{=} \boldsymbol{\gamma}_2^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{k_2})\ . \tag{6.3}$$

The resulting TSPHF is a KV-TSPHF if $\mathcal{V}_1$ is a KV-DVS, a CS-TSPHF if $\mathcal{V}_1$ is a CS-DVS, and a GL-TSPHF if $\mathcal{V}_1$ is a GL-DVS. More precisely, we have the following theorem.

Theorem 6.3.6. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. We suppose that* $\mathscr{L}_2$ *is a hard-subset-membership language and that* $\mathcal{V}_2$ *is a KV-DVS.*

Then the TSPHF for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.3.5 is perfectly correct, reference-string-indistinguishable, statistically GL/CS smooth, and perfectly zero-knowledge. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against reference string indistinguishability of the TSPHF, we can construct an adversary* $\mathcal{B}$ *against subset-membership for* $\mathscr{L}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{\mathtt{crs\text{-}ind}}(\mathcal{A}, \mathfrak{K}) \le \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}, \mathfrak{K})\ .$$

And if $\mathcal{A}$ *is a (potentially unbounded) adversary against computational GL/CS smoothness and if* $\mathcal{V}_1$ *is* ε*-sound, then:*

$$\mathsf{Adv}^{\mathtt{gl/cs\text{-}smooth}}(\mathcal{A}, \mathfrak{K}) \le \varepsilon\ .$$

Furthermore, if $\mathcal{V}_1$ *is a CS-DVS, then the resulting TSPHF is a CS-TSPHF. If* $\mathcal{V}_2$ *is a KV-DVS, then the resulting TSPHF is a perfectly smooth KV-TSPHF.*

Proof. Perfect correctness directly comes from perfect completeness of $\mathcal{V}$.

Reference string indistinguishability is directly implied by hard subset membership of $\mathscr{L}_2$.

Computational smoothness. When $\mathsf{crs} = (\mathsf{lpar}_2, x_2)$ is generated by $\mathsf{T.Setup}$, $x_2 \notin \mathscr{L}_2$ and therefore $x = (x_1, x_2) \in \mathscr{L}$ if and only $x_1 \in \mathscr{L}_1$. Therefore computational smoothness holds statistically, thanks to the smoothness property of the SPHF derived from the DVS $\mathcal{V}$.

Perfect zero-knowledge. Let x_1 be a word in $\mathscr{L}_1$ with witness w_1, and let $(\mathsf{crs} = (\mathsf{lpar}_2, x_2), \mathsf{trap} = w_2) \xleftarrow{\$} \mathsf{T.TSetup}(\mathsf{lpar}_1)$. We have:

$$\begin{aligned}\mathsf{pH} &= \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\lambda}_1 \otimes \boldsymbol{\theta}_2) = \boldsymbol{\gamma}_1^\intercal \bullet ((\mathsf{Id}_{k_1} \bullet \boldsymbol{\lambda}_1) \otimes (\Gamma_2 \bullet \boldsymbol{\lambda}_2)) = \boldsymbol{\gamma}_1^\intercal \bullet (\mathsf{Id}_{k_1} \otimes \Gamma_2) \bullet (\boldsymbol{\lambda}_1 \otimes \boldsymbol{\lambda}_2)\,,\\ \mathsf{trapH} &= \boldsymbol{\gamma}_1^\intercal \bullet (\boldsymbol{\theta}_1 \otimes \boldsymbol{\lambda}_2) = \boldsymbol{\gamma}_1^\intercal \bullet ((\Gamma_1 \bullet \boldsymbol{\lambda}_1) \otimes (\mathsf{Id}_{k_2} \bullet \boldsymbol{\lambda}_2)) = \boldsymbol{\gamma}_1^\intercal \bullet (\Gamma_1 \otimes \mathsf{Id}_{k_2}) \bullet (\boldsymbol{\lambda}_1 \otimes \boldsymbol{\lambda}_2)\ .\end{aligned}$$

If $\mathsf{hp} = (\boldsymbol{\gamma}^\intercal, \rho)$ is accepted by VerHP, then Equation (6.3) is satisfied, and thus $\mathsf{pH} = \mathsf{trapH}$. Therefore, zero-knowledge holds perfectly. □

Composite-order setting. The construction can be extended to the composite-order setting (replacing DVSs $\mathcal{V}_1$ and $\mathcal{V}_2$ by DMs $\mathcal{M}_1$ and $\mathcal{M}_2$), in the CS-TSPHF and GL-TSPHF case, as long as, with overwhelming probability, a random word x_2 is strongly non-degenerated in $\mathcal{M}_2$ mod M' with M' a factor of the order M of the DMs, such that $\mathcal{M}_1$ mod M' is sound. We just need to change $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_p^{1\times n}$ into $\boldsymbol{\alpha}^\intercal \xleftarrow{\$} \mathbb{Z}_M^{m\times n}$ for some large enough integer m, and to extract from the hash value (H, pH, or trapH) a uniform bit string, by using a randomness extractor. The seed for the randomness extractor seed is generated by HashKG and added to hk and hp.

If we can use a deterministic randomness extractor, we could also get a KV-TSPHF using the same construction starting with a KV-DM $\mathcal{M}_1$. Unfortunately, while we know a randomness extractor for any distribution with high enough min-entropy (see Section 2.2.3), there does not necessarily exist a deterministic randomness extractor for the distributions of hash values coming from PHFs from DMs.

6.3.2.3 Construction Based on Mixed Pseudorandomness

Similarly to NIZK, we can optimize the previous construction in the prime-order setting by replacing the second DVS $\mathcal{V}_2$ by a Pr-DVS. We need to suppose that $\ddot{\mathscr{L}}$ is witness samplable (see Section 5.2.3).

Construction 6.3.7. *The construction is the same as Construction 6.3.5, except that:*

- $\mathsf{T.Setup}$ *behaves like* $\mathsf{T.TSetup}$ *and picks* $x_2 \xleftarrow{\$} \mathscr{L}_2$ *instead of* $x_2 \xleftarrow{\$} \mathcal{X}_2 \setminus \mathscr{L}_2$*;*
- *we do not require that* $\mathscr{L}_2$ *is a hard-subset-membership language but only that* $\mathcal{V}_2$ *is a Pr-DVS;*
- *we also suppose that* $\mathscr{L}_1$ *is witness samplable.*

We point out that we never use the trapdoor ltrap_1 of $\mathscr{L}_1$ (for witness samplability) in the TSPHF scheme. It is only used for the proof.

Theorem 6.3.8. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs over two multiplicatively compatible sub-graded rings* $\mathfrak{G}_1$ *and* $\mathfrak{G}_2$ *of some graded ring* $\mathfrak{G}$*. We suppose that* $\mathcal{V}_2$ *is a Pr-DVS and that* $\mathscr{L}_1$ *is witness samplable.*

Then the TSPHF for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.3.7 is perfectly correct, perfectly reference-string-indistinguishable, computationally GL/CS-smooth, and perfectly zero-knowledge. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against GL/CS smoothness of the TSPHF, we can construct an adversary* $\mathcal{B}$ *against pseudorandomness of* $\mathcal{V}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{crs\text{-}ind}(\mathcal{A}, \mathfrak{K}) \leq m_1 \cdot \mathsf{Adv}^{psrnd}(\mathcal{B}, \mathfrak{K}) + \varepsilon\,,$$

where $m_1 := n_1 - \max_{\mathsf{lpar}_1}(\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1})$*, where* $\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$ *is the dimension of the vector space* $\hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$*, or in other words the rank of* $\Gamma_{1,\mathsf{lpar}_1}$*. Furthermore, if* $\mathcal{V}_1$ *is a CS-DVS, then the resulting TSPHF is a CS-TSPHF. If* $\mathcal{V}_2$ *is a KV-DVS, then the resulting TSPHF is a KV-TSPHF.*

Proof. *Perfect correctness* and *perfect zero-knowledge* are proven exactly as for Theorem 6.3.6, while *perfect reference string indistinguishability* is trivial.

Finally, *computational smoothness* is directly implied by mixed pseudorandomness and Theorem 5.2.6. □

Historical note 6.3.9. *When the second DVS* $\mathcal{V}_2$ *corresponds to the DDH assumption (Example 5.1.5), we get the construction in [BBC+13c].*

6.3.3 Implicit Zero-Knowledge Arguments (iZK)

6.3.3.1 Definition and Direct Application

Definition. Let us formally define iZK.

Definition 6.3.10 (iZK)**.** *An iZK for a language* $(\ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}})$ *is defined by a tuple of seven polynomial-time algorithms* $(\mathsf{iSetup}, \mathsf{iKG}, \mathsf{iTKG}, \mathsf{iEnc}, \mathsf{iDec}, \mathsf{iTDec})$*, where:*

- $\mathsf{iSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ *outputs a CRS* crs *for the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$*; as usual, we suppose that* crs *implicitly contains* gpar *and* $\ddot{\mathsf{lpar}}$*;*
- $\mathsf{iTSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ *generates the (simulated) common reference string* crs *together with a trapdoor* trap *for the global parameters* gpar *and the language parameters* $\ddot{\mathsf{lpar}}$*, and outputs the pair* $(\mathsf{crs}, \mathsf{trap})$*;*
- $\mathsf{iKG}(\mathsf{crs}, \ddot{\chi}, \ddot{w})$ *generates an encryption/decryption (a.k.a., public/secret) key pair* $(\mathsf{iek}, \mathsf{idk})$*, associated to a word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$*, with witness* $\ddot{w}$*;*
- $\mathsf{iTKG}(\mathsf{crs}, \mathsf{trap}, \ddot{\chi})$ *generates a encryption/trapdoor (a.k.a., public/trapdoor) key pair* $(\mathsf{iek}, \mathsf{itk})$*, associated to a word* $\ddot{\chi} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}}$*;*
- $\mathsf{iEnc}(\mathsf{crs}, \mathsf{iek}, \ddot{\chi})$ *outputs a pair* (c, K)*, where* c *is a ciphertext corresponding to the plaintext* K *(an ephemeral key), for the public key* iek *and the word* $\ddot{\chi} \in \ddot{\mathcal{X}}_{\ddot{\mathsf{lpar}}}$*;*
- $\mathsf{iDec}(\mathsf{crs}, \mathsf{idk}, \mathsf{c})$ *determistically decrypts the ciphertext* c *using the decryption key* idk *and outputs the ephemeral key* K*;*

- $\mathsf{iTDec}(\mathsf{crs}, \mathsf{itk}, \mathsf{c})$ *determistically decrypts the ciphertext* c *using the trapdoor key* itk *and outputs the ephemeral key* K.

The set of plaintexts or ephemeral keys is denoted Π. *An iZK has to satisfies the following properties:*

- Perfect correctness. *The encryption is the reverse operation of the decryption, with or without a trapdoor: for any* $\ddot{\mathsf{lpar}}$*:*
  - *for any* crs *generated by* $\mathsf{crs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$ *or generated by* $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$*, for any word* $\ddot{\chi} \in \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ *with witness* $\ddot{w}$ *(i.e.,* $\mathscr{R}_{\mathsf{lpar}}(\ddot{\chi}, \ddot{w}) = 1$*), for any key pair* $(\mathsf{iek}, \mathsf{idk}) \xleftarrow{\$} \mathsf{iKG}(\mathsf{crs}, \ddot{\chi}, \ddot{w})$*, and any* $(\mathsf{c}, \mathsf{K}) \xleftarrow{\$} \mathsf{iEnc}(\mathsf{iek}, \ddot{\chi})$*, we have* $\mathsf{K} = \mathsf{iDec}(\mathsf{idk}, \mathsf{c})$*;*
  - *for any* $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$*, for any word* $\ddot{\chi} \in \ddot{\mathscr{X}}_{\ddot{\mathsf{lpar}}}$*, for any key pair* $(\mathsf{iek}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}(\mathsf{trap}, \ddot{\chi})$ *and* $(\mathsf{c}, \mathsf{K}) \xleftarrow{\$} \mathsf{iEnc}(\mathsf{iek}, \ddot{\chi})$*, we have* $\mathsf{K} = \mathsf{iTDec}(\mathsf{itk}, \mathsf{c})$*;*

- Reference string indistinguishability (a.k.a, setup indistinguishability). *A polynomial-time adversary cannot distinguish a normal CRS generated by* iSetup *from a trapdoor CRS generated by* $\mathsf{iTSetup}$. *Formally, the advantage of an adversary* $\mathcal{A}$ *against reference string indistinguishability is defined by the experiments* $\mathsf{Exp}^{crs\text{-}ind\text{-}b}$ *depicted in Figure 6.5. The iZK is* reference-string-indistinguishable*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$.

- Soundness. *When the CRS* crs *is generated by* iSetup*, and when* $\ddot{\chi} \notin \mathscr{L}$*, the distribution of* K *is indistinguishable from the uniform distribution, even given* c. *Formally, the advantage of an adversary* $\mathcal{A}$ *against soundness is defined by the experiments* $\mathsf{Exp}^{sound\text{-}b}$ *depicted in Figure 6.5. The iZK is* sound*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$.

- Zero-knowledge. *When the CRS* crs *is generated by* $\mathsf{iTSetup}$*, keys generation by* iKG *and decryption by* iDec *are indistinguishable from key generation by* iTKG *and decryption by* iTDec. *Formally, the advantage of an adversary* $\mathcal{A}$ *against zero-knowledge is defined by the experiments* $\mathsf{Exp}^{zk\text{-}b}$ *depicted in Figure 6.5. The iZK is* zero-knowledge*, if this advantage is negligible for any polynomial-time adversary* $\mathcal{A}$.

Historical note 6.3.11. *In [BCPW15], we defined our security notions with a "composable" security flavor, as Groth and Sahai in [GS08]: soundness and zero-knowledge were statistical properties, the only computational property was the setup indistinguishability property. Here, we allow computational soundness and computational zero-knowledge, to enable more efficient instantiations based on pseudorandomness. This weaker definition still works for all the applications in [BCPW15].*

Furthermore, our definition does not include labels nor tags, as they are not useful for basic soundness, and contrary to [BCPW15], we do not consider simulation-soundness in this thesis.

Direct application. As already noted, iZK can replace PHFs in the honest-verifier zero-knowledge arguments in Sections 2.5.3.1 and 6.1 to get zero-knowledge arguments. A concrete example of such a scheme is depicted in Figure 6.6.

Reference String Indistinguishability

$\mathsf{Exp}^{\texttt{crs-ind-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- **if** $b = 0$ **then**
  - $\mathsf{crs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **else**
  - $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- **return** $\mathcal{A}(\mathsf{crs}, \ddot{\mathsf{ltrap}})$

Soundness

$\mathsf{Exp}^{\texttt{sound-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $\mathsf{crs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $(\mathsf{iek}, \ddot{\chi}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$
- $(\mathsf{c}, \mathsf{K}) \xleftarrow{\$} \mathsf{iEnc}(\mathsf{crs}, \ddot{\chi})$
- **if** $b = 0$ and $\ddot{\chi} \notin \ddot{\mathscr{L}}_{\ddot{\mathsf{lpar}}}$ **then**
  - $\mathsf{K} \xleftarrow{\$} \Pi$
- **return** $\mathcal{A}(\mathsf{st}, \mathsf{c}, \mathsf{K})$

Zero-Knowledge

$\mathsf{Exp}^{\mathsf{zk}\text{-}b}(\mathcal{A}, \mathfrak{K})$
- $\mathsf{gpar} \xleftarrow{\$} \mathsf{Setup.gpar}(1^{\mathfrak{K}})$
- $(\ddot{\mathsf{lpar}}, \ddot{\mathsf{ltrap}}) \xleftarrow{\$} \ddot{\mathsf{Setup}}.\mathsf{lpar}(\mathsf{gpar})$
- $(\mathsf{crs}, \mathsf{trap}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{gpar}, \ddot{\mathsf{lpar}})$
- $(\ddot{\chi}, \ddot{w}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathsf{trap}, \ddot{\mathsf{ltrap}})$
- **if** $\mathscr{R}_{\ddot{\mathsf{lpar}}}(\ddot{\chi}, \ddot{w}) = 0$ **then**
  - **return** 0
- **if** $b = 0$ **then**
  - $(\mathsf{iek}, \mathsf{idk}) \xleftarrow{\$} \mathsf{iKG}(\mathsf{crs}, \ddot{\chi}, \ddot{w})$
- **else**
  - $(\mathsf{iek}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}(\mathsf{crs}, \mathsf{trap}, \ddot{\chi})$
- $(\mathsf{c}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{st}, \mathsf{iek})$
- **if** $b = 0$ **then**
  - $\mathsf{K}' \leftarrow \mathsf{iDec}(\mathsf{crs}, \mathsf{idk}, \mathsf{c})$
- **else**
  - $\mathsf{K}' \leftarrow \mathsf{iTDec}(\mathsf{crs}, \mathsf{itk}, \mathsf{c})$
- **return** $\mathcal{A}(\mathsf{st}, \mathsf{K}')$

Figure 6.5: Experiments for Definition 6.3.10 (iZK)

Prover		Verifier
Input: $(\ddot{\chi}, \ddot{w})$	CRS: crs	Input: $\ddot{\chi}$
$(\mathsf{iek}, \mathsf{idk}) \xleftarrow{\$} \mathsf{iKG}(\mathsf{crs}, \ddot{\chi}, \ddot{w})$	$\xrightarrow{\mathsf{iek}}$	
	$\xleftarrow{\mathsf{c}}$	$(\mathsf{c}, \mathsf{K}) \xleftarrow{\$} \mathsf{iEnc}(\mathsf{crs}, \mathsf{iek})$
$\mathsf{K}' \leftarrow \mathsf{iDec}(\mathsf{crs}, \mathsf{idk}, \mathsf{c})$	$\xrightarrow{\mathsf{K}'}$	
		if $\mathsf{K} = \mathsf{K}'$ **then** **accept** **else reject**

Figure 6.6: Zero-knowledge proof from an iZK

6.3.3.2 Construction Based on Disjunctions with a Hard-Subset-Membership Language

Prime-order setting. Let us start by the construction in the prime-order setting, which is an improved (and extended to any hard-subset-membership language) version of the construction in [BCPW15].

Construction 6.3.12 (iZK from disjunctions of DVSs)**.** *We suppose that we have a DVS* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *for* $\ddot{\mathscr{L}} = \mathscr{L}_1$*, together with a KV-DVS* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *for a hard-subset-membership language* $\mathscr{L}_2$*. We suppose that* Γ_1 *does not depend on* ρ_1*, but* $\boldsymbol{\theta}_1$ *can depend on* ρ*. Therefore,* Γ *does not depend on* ρ*. Furthermore, we consider the GL disjunction* $\mathcal{V} = (p, \mathfrak{G}, \mathcal{X}, \mathscr{L}, \mathscr{R}, n, k, \mathsf{RGen}, \Gamma, \boldsymbol{\theta}, \boldsymbol{\lambda})$ *of* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*. We recall that* $n = n_1 + n_2 + 2$ *and* $k = k_1 + k_2 + 1$*.*

We construct an iZK for $\ddot{\mathscr{L}} := \mathscr{L}_1$ *as follows:*

- $\mathsf{iSetup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar_2}(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with a word* $x_2 \xleftarrow{\$} \mathcal{X}_{2,\mathsf{lpar}_2} \setminus \mathscr{L}_{2,\mathsf{lpar}_2}$*. It then outputs* $\mathsf{crs} := (\mathsf{lpar}_2, x_2)$*.*

- $\mathsf{iTSetup}(\mathsf{lpar}_1)$ *generates language parameters* $\mathsf{lpar}_2 \xleftarrow{\$} \mathsf{Setup.lpar_2}(\mathsf{gpar})$ *for* $\mathscr{L}_2$*, together with a word and its witness:* $(x_2, w_2) \xleftarrow{\$} \mathscr{R}_{2,\mathsf{lpar}_2}$*. It then output* $\mathsf{crs} := (\mathsf{lpar}_2, x_2)$ *and* $\mathsf{trap} := w_2$*.*

- $\mathsf{iKG}(\mathsf{crs}, x_1, w_1)$ *outputs a pair* $(\mathsf{iek}, \mathsf{idk})$*, where:*

$$\begin{aligned} \mathsf{tk} &:= \boldsymbol{\alpha}'^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times k} & \mathsf{tp} &:= \boldsymbol{\gamma}'^{\mathsf{T}} := \boldsymbol{\alpha}'^{\mathsf{T}} \bullet \Gamma^{\mathsf{T}}(x) \quad \in \mathfrak{G}^{1\times n} \\ \mathsf{iek} &:= \mathsf{tp} & \mathsf{idk} &:= (x_1, \mathsf{tk}, w_1)\,, \end{aligned}$$

  where $x = (x_1, x_2)$*;*

- $\mathsf{iTKG}(\mathsf{crs}, \mathsf{trap}, x_1)$ *outputs a pair* $(\mathsf{iek}, \mathsf{idk})$ *where:*

$$\begin{aligned} \mathsf{tk} &:= \boldsymbol{\alpha}'^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times k} & \mathsf{tp} &:= \boldsymbol{\gamma}'^{\mathsf{T}} := \boldsymbol{\alpha}'^{\mathsf{T}} \bullet \Gamma^{\mathsf{T}}(x) \quad \in \mathfrak{G}^{1\times n} \\ \mathsf{iek} &:= \mathsf{tp} & \mathsf{itk} &:= (x_1, \mathsf{tk}, \mathsf{trap})\,; \end{aligned}$$

- $\mathsf{iEnc}(\mathsf{crs}, \mathsf{iek}, x_1)$ *outputs a pair* (c, K) *where:*

$$\begin{aligned} \rho &\xleftarrow{\$} \mathsf{RGen}(\ddot{\mathsf{lpar}}) & \zeta &\xleftarrow{\$} \mathbb{Z}_p \\ \mathsf{hk} &:= \boldsymbol{\alpha}^{\mathsf{T}} \xleftarrow{\$} \mathbb{Z}_p^{1\times n} & \mathsf{hp} &:= \boldsymbol{\gamma}^{\mathsf{T}} := \boldsymbol{\alpha}^{\mathsf{T}} \bullet \Gamma(x) \quad \in \mathfrak{G}^{1\times k} \\ \mathsf{tpH} &:= \boldsymbol{\gamma}' \bullet \boldsymbol{\alpha} \quad \in \mathfrak{G} & \mathsf{H} &:= \boldsymbol{\alpha}^{\mathsf{T}} \bullet \boldsymbol{\theta}(x, \rho) \quad \in \mathfrak{G} \\ \mathsf{c} &:= (\zeta, \mathsf{hp}, \rho) & \mathsf{K} &:= (\zeta \bullet \mathsf{H}) \;+\; \mathsf{tpH} \quad \in \mathfrak{G}\,; \end{aligned}$$

- $\mathsf{iDec}(\mathsf{crs}, \mathsf{idk}, \mathsf{c})$ *outputs* K' *where:*

$$\begin{aligned} \mathsf{tH} &:= \boldsymbol{\alpha}'^{\mathsf{T}} \bullet \boldsymbol{\gamma} \quad \in \mathfrak{G} \\ \mathsf{pH} &:= \boldsymbol{\gamma}^{\mathsf{T}} \bullet \boldsymbol{\lambda}(x, (w_1, \perp), \rho) \\ \mathsf{K}' &:= (\zeta \bullet \mathsf{pH}) \;+\; \mathsf{tH} \quad \in \mathfrak{G}\,; \end{aligned}$$

- $\mathsf{iTDec}(\mathsf{crs}, \mathsf{itk}, \mathsf{c})$ *outputs* K' *where:*

$$\begin{aligned}\mathsf{tH} &:= \boldsymbol{\alpha}'^{\mathsf{T}} \bullet \boldsymbol{\gamma} \quad \in \mathfrak{G} \\ \mathsf{pH} &:= \boldsymbol{\gamma}^{\mathsf{T}} \bullet \boldsymbol{\lambda}(\mathfrak{x}, (\perp, w_2), \rho) \\ \mathsf{K}' &:= (\zeta \bullet \mathsf{pH}) \ \boldsymbol{+} \ \mathsf{tH} \quad \in \mathfrak{G}\,;\end{aligned}$$

We have the following security theorem.

Theorem 6.3.13. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}_1, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}_2, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs. We suppose that* Γ_1 *does not depend on* ρ_1*. We suppose that* $\mathscr{L}_2$ *is a hard-subset-membership language and* $\mathcal{V}_2$ *is a KV-DVS.*

Then the iZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.3.12 is perfectly correct, reference-string-indistinguishable, statistically sound, and perfectly zero-knowledge. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against reference string indistinguishability of the iZK, we can construct an adversary* $\mathcal{B}$ *against subset-membership for* $\mathscr{L}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{\mathtt{crs\text{-}ind}}(\mathcal{A}, \mathfrak{K}) \leq \mathsf{Adv}^{\mathtt{sub\text{-}memb}}(\mathcal{B}, \mathfrak{K}) \ .$$

And if $\mathcal{A}$ *is a (potentially unbounded) adversary against soundness, and if* $\mathcal{V}_1$ *is* ε*-sound then:*

$$\mathsf{Adv}^{\mathtt{sound}}(\mathcal{A}, \mathfrak{K}) \leq 1/p + \varepsilon \ .$$

Proof. Perfect correctness directly comes from perfect completeness of $\mathcal{V}$.

Reference string indistinguishability is directly implied by hard subset membership of $\mathscr{L}_2$.

Soundness. Let $\mathfrak{x}_1 \in \mathcal{X}_1 \setminus \mathscr{L}_1$, $\mathfrak{x}_2 \in \mathcal{X}_2 \setminus \mathscr{L}_2$, and $\mathsf{tp} := \boldsymbol{\gamma}'^{\mathsf{T}} \in \mathfrak{G}^{1\times n}$. We suppose that $\boldsymbol{\theta} \notin \mathrm{ColSpan}\,(\Gamma)$, which happens with probability at least $1-\varepsilon$ by ε-soundness of $\mathcal{V}_1$ implied by ε-soundness of $\mathcal{V}$. Let us first prove that there exists at most one scalar $\zeta \in \mathbb{Z}_p$ such that

$$(\zeta \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}' \in \mathrm{ColSpan}\,(\Gamma) \ ,$$

where $\mathfrak{x} = (\mathfrak{x}_1, \mathfrak{x}_2)$. Let us suppose by contradiction that this is not the case and that there exist two distinct scalars ζ_1 and ζ_2 such that:

$$(\zeta_1 \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}' \in \mathrm{ColSpan}\,(\Gamma) \ , \qquad (\zeta_2 \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}' \in \mathrm{ColSpan}\,(\Gamma) \ .$$

By subtracting the values on the left-hand sides, we get:

$$(\zeta_1 - \zeta_2) \bullet \boldsymbol{\theta}(\mathfrak{x}) \in \mathrm{ColSpan}\,(\Gamma) \ ,$$

and as $\zeta_1 \neq \zeta_2$, $\boldsymbol{\theta}(\mathfrak{x}) \in \mathrm{ColSpan}\,(\Gamma)$. This is impossible.

Let us now suppose that $\zeta \in \mathbb{Z}_p$ is chosen such that

$$(\zeta \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}' \notin \mathrm{ColSpan}\,(\Gamma) \ ,$$

which happens with probability at least $1 - 1/p$. We then remark that:

$$\mathsf{K} = (\zeta \bullet \mathsf{H}) \ \boldsymbol{+} \ \mathsf{tpH} = \boldsymbol{\alpha}^{\mathsf{T}} \bullet ((\zeta \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}') \ .$$

In other words, we can see K as the "hash value" of "$((\zeta \bullet \boldsymbol{\theta}(\mathfrak{x})) \ \boldsymbol{+} \ \boldsymbol{\gamma}')$", and therefore K is uniformly random from the point of view of the adversary;

Perfect zero-knowledge. We separate the analysis in two cases:

1. the projection key hp provided by the adversary in c is valid, i.e., there exists $\boldsymbol{\alpha}^\intercal \in \mathbb{Z}_p^{1\times n}$, such that $\mathsf{hp} = \boldsymbol{\gamma}^\intercal = \boldsymbol{\alpha}^\intercal \bullet \Gamma$. In this case, the projected hash value pH computed using the valid witness w_1 (in iDec) is the same as the one computed using the valid witness w_2 (in iTDec). And iDec and iTDec output the same value, if they used the same tk;

2. otherwise, $\boldsymbol{\gamma} \notin \mathrm{ColSpan}\,(\Gamma^\intercal)$. In this case, we can see tk and tp as the hashing key and the projection key for the language of valid (transposed) projection keys ($\mathrm{ColSpan}\,(\Gamma^\intercal) \subseteq \mathfrak{G}^k$), and tH as the hash value of $\boldsymbol{\gamma}$. By smoothness, this hash value is uniformly random from the point of the adversary. Therefore, iDec and iTDec both output a uniform graded ring element.

This concludes the proof. □

Remark 6.3.14 (reducing the ciphertext size)**.** *Similarly to Example 3.1.5, ζ can instead be defined as $\mathsf{Ext}(\boldsymbol{\gamma}^\intercal)$, where Ext is a deterministic randomness extractor (see Section 2.2.3). Such an extractor exists for some elliptic curves [CFPZ09]. The security proof still works in this case, the only difference is the bound for soundness. Concretely, if $\zeta \in \{0, \ldots, 2^m - 1\} \subseteq \mathbb{Z}_p$ (the extractor extracts m bits of entropy), if $\mathcal{A}$ is a (potentially unbounded) adversary against soundness, and if $\mathcal{V}_1$ is ε-sound, then:*

$$\mathsf{Adv}^{sound}(\mathcal{A}, \mathfrak{K}) \leq 1/2^m + \varepsilon \ .$$

Remark 6.3.15 (iZK for any NP language)**.** *We can construct iZK for any NP language using the same ideas as for constructing NIZK from disjunctions of DVSs for any NP language, in Section 6.1.2.3. Concretely, if the NP language is defined by a circuit C, we just add an ElGamal public key in crs and the prover adds ciphertexts encrypting the value of each wire of the circuit evaluated on $\ddot{x}$ and $\ddot{w}$, to the public key K. The language $\mathscr{L}_1$ is used to check the validity of all these ciphertexts (inputs, output and gates), as in Section 6.1.2.3.*

Composite-order setting. The construction can be extended to the composite-order setting, as for TSPHFs in Section 6.3.2.2, using m scalars $\zeta_1, \ldots, \zeta_m$.

6.3.3.3 Construction Based on Mixed Pseudorandomness

Similarly to NIZK and TSPHFs, we can optimize the previous construction in the prime-order setting by replacing the second DVS $\mathcal{V}_2$ by a Pr-DVS. We need to suppose that $\ddot{\mathscr{L}}$ is such that the trapdoor ltrap_1 output by $\mathsf{Setup.lpar}_1$ enables to check whether a word $x_1 \in \mathcal{X}_1$ is in $\mathscr{L}_1$ in polynomial time.

Construction 6.3.16. *The construction is the same as Construction 6.3.5, except that:*

- iSetup *behaves like* $\mathsf{iTSetup}$ *and picks* $x_2 \xleftarrow{\$} \mathscr{L}_2$ *instead of* $x_2 \xleftarrow{\$} \mathcal{X}_2 \setminus \mathscr{L}_2$*;*
- *we do not require that $\mathscr{L}_2$ is a hard-subset-membership language but only that $\mathcal{V}_2$ is a Pr-DVS;*
- *we also suppose that the trapdoor* ltrap_1 *output by* $\mathsf{Setup.lpar}_1$ *enables to check whether a word $x_1 \in \mathcal{X}_1$ is in $\mathscr{L}_1$ in polynomial time.*

We point out that we never use the trapdoor ltrap_1 of $\mathscr{L}_1$ for witness samplability in the TSPHF scheme. It is only used for the proof.

Theorem 6.3.17. *Let* $\mathcal{V}_1 = (p, \mathfrak{G}, \mathcal{X}_1, \mathscr{L}_1, \mathscr{R}_1, n_1, k_1, \mathsf{RGen}_1, \Gamma_1, \boldsymbol{\theta}_1, \boldsymbol{\lambda}_1)$ *and* $\mathcal{V}_2 = (p, \mathfrak{G}, \mathcal{X}_2, \mathscr{L}_2, \mathscr{R}_2, n_2, k_2, \mathsf{RGen}_2, \Gamma_2, \boldsymbol{\theta}_2, \boldsymbol{\lambda}_2)$ *be two DVSs. We suppose that* Γ_1 *does not depend on* ρ_1*. We suppose that* $\mathcal{V}_2$ *is a Pr-DVS and that the trapdoor* ltrap_1 *output by* $\mathsf{Setup.lpar_1}$ *enables to check whether a word* $\underset{\sim}{x}_1 \in \mathcal{X}_1$ *is in* $\mathscr{L}_1$ *in polynomial time.*

Then the iZK for $\ddot{\mathscr{L}} = \mathscr{L}_1$ *in Construction 6.3.16 is perfectly correct, perfectly reference-string-indistinguishable, computationally sound, and perfectly zero-knowledge. More precisely, if* $\mathcal{A}$ *is a polynomial-time adversary against soundness of the iZK, we can construct an adversary* $\mathcal{B}$ *against pseudorandomness of* $\mathcal{V}_2$ *with similar running time such that:*

$$\mathsf{Adv}^{crs\text{-}ind}(\mathcal{A}, \mathfrak{K}) \leq m_1 \cdot \mathsf{Adv}^{psrnd}(\mathcal{B}, \mathfrak{K}) + 1/p\,,$$

where $m_1 := n_1 - \max_{\mathsf{lpar}_1}(\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1})$*, where* $\dim \hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$ *is the dimension of the vector space* $\hat{\mathscr{L}}_{1,\mathsf{lpar}_1}$*, or in other words the rank of* $\Gamma_{1,\mathsf{lpar}_1}$*.*

Proof. *Perfect correctness* and *perfect zero-knowledge* are proven exactly as for Theorem 6.3.13, while *perfect reference string indistinguishability* is trivial.

Finally, *computational smoothness* is implied by mixed pseudorandomness and Theorem 5.2.6, using a proof similar to the one in Theorem 6.3.13. □

Historical note 6.3.18. *This construction is completely new.*

Chapter 7

Conclusion and Open Questions

7.1 Conclusion

In this thesis, we introduced the notion of *diverse modules* (DMs) that are a fruitful way to represent many algebraic languages used in cryptographic schemes. As a particular case of diverse groups [CS02], DMs also yield an (approximate or weakly approximate) universal *projective hash function* (PHF) for the language they represent. Most, if not all, languages for which we know a universal PHF can be represented by DMs. While characterizing the class of languages admitting a universal PHF remain an interesting open problem, the structural properties of DMs enable us to better understand the class of languages represented by DMs. In particular, we showed in this thesis that this class is closed under conjunction, and approximately closed under disjunction. DMs can also be efficiently extended, via t-sound extensions, to yield stronger forms of universal PHFs, namely t-universal PHFs, which provide even more applications.

DMs are also valuable in their own right. Using disjunctions and t-sound extensions, we showed how to construct *non-interactive zero-knowledge arguments* (NIZK), *trapdoor smooth projective hash functions* (TSPHFs), and *implicit zero-knowledge arguments* (iZK). Even though NIZK is a standard primitive in cryptography, DMs shed a new light on it and provided more efficient instantiations. TSPHFs and iZK, on the other hand, were both introduced in this thesis as zero-knowledge variants of PHFs and lightweight alternatives to costly NIZK, in particular for *password authenticated key exchange* (PAKE) in the UC model and for enforcing semi-honest behavior in multi-party computation.

DVSs, which are DMs over prime-order fields, are an important particular case of DMs. While DVSs were already proposed in [CS02], we provide a more in-depth analysis of them. In addition to present conjunctions, disjunctions, and t-sound extensions of DVSs, as a warm-up for the same operations on DMs, we introduced *pseudorandom diverse vector spaces* (Pr-DVSs), which are alternatives to DVSs for hard-subset-membership languages. We analyzed the behavior of Pr-DVSs with respect to disjunctions, and called the resulting property: mixed pseudorandomness. This enabled us to improve the efficiency of our constructions of NIZK, TSPHFs, and iZK, in the prime-order setting.

7.2 Open Questions

In this section, I would like to highlight some open questions that draw me into the direction of constructing diverse modules and that I dreamed of solving during my thesis. Now, I just dream they will be solved during my lifetime.

There are of course many other technical interesting questions ranging from:

Question 7.1. *Can we construct a one-time simulation-sound NIZK with short CRS (independent of the security parameter)?*

to:

Question 7.2. *Can we construct lower bounds for the size of DVS or DM for a given language?*

These questions might be much more tractable but, from my point of view, they are also much less interesting than the following open questions.

Class of languages. The first open question is naturally the following:

Question 7.3. *Can we characterize the class of languages captured by SPHFs (or universal PHFs)?*

On the one hand, we know that there does not exist even a GL-SPHF for a NP-complete language unless the polynomial hierarchy collapses. On the other hand, the only languages for which we know of an SPHF are the ones represented by DMs. We do not even know if GL-SPHFs, CS-SPHFs, and KV-SPHFs capture the same class of languages.

This leads to some simpler open questions:

Question 7.4. *Are the classes of languages captured by GL-SPHF, CS-SPHF, and KV-SPHF the same?*

Question 7.5. *Can we construct an SPHF for a language which cannot be expressed as a diverse module (DM)?*

Other simpler open questions related to the closure of the class of languages captured by SPHF. In [ACP09], Abdalla, Chevalier and Pointcheval already showed that it is closed by conjunction for KV-SPHFs, CS-SPHFs, and GL-SPHFs, and it is closed by disjunction for GL-SPHFs.

Question 7.6. *Are the classes of languages captured by GL-SPHFs, CS-SPHFs, and KV-SPHFs closed by complement?*

Question 7.7. *Are the classes of languages captured by CS-SPHFs and KV-SPHFs closed by disjunction?*

Lattice-based languages Other important languages for which the construction of SPHFs is very interesting are lattice-based languages. In [KV09], Katz and Vaikuntanathan manage to construct the first and only known (approximate) SPHF for a lattice-based language: the language of ciphertexts of some given plaintext for an LWE-based IND-CCA encryption scheme. This encryption scheme uses a special decryption procedure to ensure smoothness on any ciphertext which cannot be decrypted correctly. In particular, its time complexity is linear in the size of the finite field over which the encryption scheme is defined. This makes the use of superpolynomial finite fields impossible.

Witness encryption Finally, we can look at a computational relaxation of the smoothness property. Witness encryption [GGSW13] basically corresponds to computationally smooth GL-SPHF. Some candidate constructions of witness encryption for any NP language were proposed based on multilinear maps [GGSW13; GLW14] or iO [GGH+13]. But, these two assumptions are currently not very well understood.

Question 7.8. *Can we construct a witness encryption scheme for any NP language without multilinear maps or iO?*

Notation

General

$\mathbb{N}$	set of non-negative integers
$p, p_1, p_2, \ldots$	prime numbers
$\mathbb{Z}$	set of integers
$(\mathbb{Z}_M, +, \cdot)$ or $\mathbb{Z}_M$	ring of integers modulo M, with $M \geq 1$ being an integer
$(\mathbb{Z}_M, +)$	$\mathbb{Z}_M$ seen as an additive group
$(\mathbb{Z}_M^*, \cdot)$ or $\mathbb{Z}_M^*$	multiplicative subgroup of $\mathbb{Z}_M$
$(\mathbb{F}_{p^e}, +, \cdot)$ or $\mathbb{F}_{p^e}$	finite field of order p^e
$\mathbb{G}, \mathbb{G}_1, \ldots$	cyclic groups
$\mathbb{G}^*$	set of generators of $\mathbb{G}$
$(M, \mathbb{G})$	cyclic group of order M
$(M, \mathbb{G}, g)$	cyclic group of order M and generated by g
$(M, \mathbb{G}, \mathbb{G}_T, e)$	symmetric bilinear group of order M
$(M, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$	asymmetric bilinear group of order M
$R[X]$	ring of polynomials with coefficients in the ring R
$\lvert\mathcal{S}\rvert$	cardinal of the set $\mathcal{S}$

Bit Strings

$\{0,1\}^n$	set of bit strings of length n
$\{0,1\}^*$	set of all bitstrings
$x \mathsf{xor} y$	exclusive or between $x, y \in \{0,1\}^n$

Assignation

$y \xleftarrow{\$} \mathcal{S}$	y = uniform element from the set $\mathcal{S}$
$y \xleftarrow{\$} \mathcal{D}$	y = random element drawn according to distribution $\mathcal{D}$
$y \xleftarrow{\$} A(x)$	x = result of $\mathcal{A}$ on input x with fresh random coins
$y \leftarrow A(x)$	same when $\mathcal{A}$ is deterministic
$y \leftarrow A(x; r)$	same with random coins r
$:=$	definition

Vectors and Matrices

$\boldsymbol{u}, \boldsymbol{x}, \ldots$	column vectors: $\boldsymbol{u} = (u_i)_{i=1,\ldots,n}$
$\boldsymbol{u}^\intercal, \boldsymbol{x}^\intercal, \ldots$	row vectors: $\boldsymbol{u}^\intercal = (u_i)_{i=1,\ldots,n}^\intercal$

Provable Security

$\mathfrak{K}$	security parameter
$1^{\mathfrak{K}}$	unary representation of the security parameter
$\mathcal{A}, \mathcal{B}, \ldots$	adversaries
$f = O(g)$	$\exists n_0, c, \forall n \geq n_0, \lvert f(x) \rvert \leq c \cdot \lvert g(x) \rvert$
$g = \Omega(f)$	same
$\mathsf{Exp}^{\mathtt{exp}}$	experiment for the security notion/assumption "exp"
$\mathsf{Exp}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K})$	value returned by $\mathsf{Exp}^{\mathtt{exp}}$ when executed with $\mathcal{A}$ and $\mathfrak{K}$
$\mathsf{Succ}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K})$	success probability of $\mathcal{A}$ in $\mathsf{Exp}^{\mathtt{exp}}$: $\mathsf{Succ}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K}) := \Pr\left[\, \mathsf{Exp}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K}) = 1 \,\right]$
$\mathsf{Exp}^{\mathtt{exp}\text{-}b}$	pair of two experiments ($b \in \{0, 1\}$)
$\mathsf{Exp}^{\mathtt{exp}\text{-}b}(\mathcal{A}, \mathfrak{K})$	similar to $\mathsf{Exp}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K})$
$\mathsf{Adv}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K})$	advantage of $\mathcal{A}$ in distinguishing the experiments $\mathsf{Exp}^{\mathtt{exp}\text{-}b}$: $\mathsf{Adv}^{\mathtt{exp}}(\mathcal{A}, \mathfrak{K}) = \left\lvert \Pr\left[\, \mathsf{Exp}^{\mathtt{exp}\text{-}1}(\mathcal{A}, \mathfrak{K}) = 1 \,\right] - \Pr\left[\, \mathsf{Exp}^{\mathtt{exp}\text{-}0}(\mathcal{A}, \mathfrak{K}) = 1 \,\right] \right\rvert$

Projective Hash Functions

hk	hashing key
hp	projection key
H	hash value
pH	projected hash value

Graded Rings

$\mathfrak{G}$	graded ring
$[\tilde{\boldsymbol{v}}, x]$	element of index $\tilde{\boldsymbol{v}}$ and discrete logarithm x
$[x]$	when the index is clear from context: when $\mathfrak{G} = (p, \mathbb{G}, g)$, $[x] = g^x$
$[\tilde{\boldsymbol{v}}, A]$ or $[A]$	extension to matrices
$\bar{0}$ and $\top$	minimum and maximum indices

Miscellaneous

K	used instead of $\exists$ to indicate the extractable part of a witness in a language (see Section 2.3.3)

Abbreviations

BDDH bilinear decisional Diffie-Hellman. 159

BDDHI bilinear decisional Diffie-Hellman inversion. 159

CRS common reference string. 4, 6, 39–44, 47, 51–53, 139–142, 144, 149, 151–155, 160–162, 164–166, 170–172, 178

CRT Chinese remainder theorem. 19, 94, 97, 100, 101, 103, 109, 119

CS-DM CS diverse module. xiii, 85, 92, 107, 110, 112, 114, 115, 119, 120

CS-DVS CS diverse vector space. xiii, 66, 70, 73, 75, 76, 85, 87, 92, 107, 112, 114, 115, 119, 129, 136, 168, 170

CS-SPHF CS-smooth projective hash function. 49–51, 54, 62, 66–68, 87, 88, 115, 178

CS-TSPHF trapdoor CS-smooth projective hash function. 164, 166, 168–170

DCR decisional composite residuosity. 6, 85, 89

DDH decisional Diffie-Hellman. 6, 10, 31–34, 36, 38, 39, 56–58, 61, 62, 65, 71, 74, 80, 86, 101, 103, 106, 108, 126, 129, 153–159, 163, 170

DLin decisional linear. 34, 80, 126, 129, 153, 158, 159

DM diverse module. xii, xiii, 35, 55, 85, 88, 89, 91–93, 95, 97, 99–110, 114, 120, 123, 137–143, 145–147, 149, 152–154, 160, 161, 169, 177, 178, 185

DVS diverse vector space. xiii, 36, 55–60, 62–75, 78, 81–85, 92, 93, 98, 101, 103, 106, 107, 109, 114, 115, 121, 123, 127, 129–131, 133, 136–138, 142, 144, 145, 147, 150–152, 154, 155, 160–164, 166, 168–170, 173–178

GL-DM GL diverse module. 92, 93, 106–108, 143

GL-DVS GL diverse vector space. xiii, 66, 70, 82, 87, 92, 105, 123, 129, 130, 136, 168

GL-SPHF GL-smooth projective hash function. 49–52, 59, 66–68, 72, 82, 87, 88, 162, 178, 179

GL-TSPHF trapdoor GL-smooth projective hash function. 161, 164–166, 168, 169

IND-CCA indistinguishability under chosen ciphertext attacks. 30–34, 41, 137, 153–156, 159, 178, 185

IND-CPA indistinguishability under chosen plaintext attacks. 30–34, 40, 44, 52, 54, 141, 142, 185

iO indistinguishable obfuscation. 179

ITM interactive Turing machine. 20, 42–44, 46, 47

iZK implicit zero-knowledge argument. xiii, 39, 123, 137, 145, 160–164, 166, 170–177, 185

KV-DM KV diverse module. 92, 94, 100, 101, 103, 104, 108, 110, 115, 120, 142, 143, 145, 148, 149, 169

KV-DVS KV diverse vector space. 66, 70, 73, 75, 76, 78, 79, 81–83, 87, 92, 115, 123, 124, 126, 129, 133, 136, 146, 147, 149–152, 166, 168, 170, 173, 174

KV-SPHF KV-smooth projective hash function. 50, 51, 59, 61, 66, 82, 87, 88, 115, 124, 178

KV-TSPHF trapdoor KV-smooth projective hash function. 164, 166, 168–170

MDDH matrix decisional Diffie-Hellman. xii, xiii, 34, 55, 80, 81, 106, 110, 123, 124, 126–128, 152, 153

NIZK non-interactive zero-knowledge argument. xiii, 41, 47, 76, 78, 79, 81, 85, 106, 121, 123, 137, 142, 145–161, 163, 164, 166, 169, 175, 177, 178, 185

PAKE password authenticated key exchange. 37, 48, 50, 82, 137, 160, 177

PHF projective hash function. xii, 17, 35–37, 39, 48–50, 66, 85–88, 90, 92–94, 98, 100–104, 112–115, 123, 124, 126, 129–133, 138, 139, 141–144, 146, 160, 161, 163, 169, 171, 177, 178, 185

Pr-DVS pseudorandom diverse vector space. xiii, 123, 124, 126, 129, 130, 133, 147, 149–151, 161, 163, 169, 170, 175–177

PRG pseudorandom number generator. 139, 141

PrPHF pseudorandom projective hash function. 123, 124

QR quadratic residuosity. 6, 85, 89, 101, 106, 110

SPHF smooth projective hash function. xii, 17, 49–59, 61–69, 72, 76, 77, 80, 81, 86, 87, 93, 123, 128, 136–138, 141, 160, 162, 164–166, 169, 178, 185

SXDH symmetric external Diffie-Hellman. 154, 159

TSPHF trapdoor smooth projective hash function. xiii, 136, 137, 145, 160–170, 175, 177, 185

UC universal composability. 39, 42, 160, 177

List of Illustrations

Figures

2.1 Experiment for Definition 2.2.1 (collision resistance) 29
2.2 Experiments for Definition 2.2.3 (IND-CPA) and Definition 2.2.4 (IND-CCA) . 31
2.3 Experiments for Definition 2.3.3 (hard subset membership) 38
2.4 Experiments for Definition 2.4.3 (zero-knowledge argument) 45
2.5 Experiments for Definition 2.4.4 (honest-verifier zero-knowledge) 46
2.6 Experiment for Definition 2.4.5 (simulation-extractability) 47
2.7 Summary of the definition of PHF 49
2.8 Honest-verifier zero-knowledge proof from an SPHF 51
2.9 Protocol enabling an informer to send a message m to a secret agent 53

5.1 Experiments for Definition 5.1.1 (pseudorandomness) 125
5.2 Experiments for Definition 5.2.1 (GL/CS and KV mixed pseudorandomness) 130

6.1 Honest-verifier zero-knowledge proof from a DM via a PHF 139
6.2 Honest-verifier zero-knowledge proof from a DM via a Sigma-Protocol 140
6.3 Enforcing semi-honest behavior of Alice (A) by Bob (B) 164
6.4 Experiments for Definitions 6.3.1 and 6.3.3 (TSPHF) 167
6.5 Experiments for Definition 6.3.10 (iZK) 172
6.6 Zero-knowledge proof from an iZK 172

Tables

2.1 Relative time of operations on two curves (one with pairing and one without) 28

6.1 Comparison of our constructions of (partially extractable) honest-verifier zero-knowledge arguments from DMs 143
6.2 Comparison of NIZK for linear subspaces 158
6.3 Comparison of IND-CCA encryption schemes over cyclic and bilinear groups . 159
6.4 Duality of NIZK and TSPHF constructed from the disjunction of DMs 161

Bibliography

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. "SPHF-Friendly Non-interactive Commitments". In: *ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013, pp. 214–234. DOI: 10.1007/978-3-642-42033-7_12 (cit. on p. 10).

[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. "Security of the J-PAKE Password-Authenticated Key Exchange Protocol". In: *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 571–587. DOI: 10.1109/SP.2015.41 (cit. on p. 13).

[ABP13] Michel Abdalla, Fabrice Ben Hamouda, and David Pointcheval. "Tighter Reductions for Forward-Secure Signature Schemes". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 292–311. DOI: 10.1007/978-3-642-36362-7_19 (cit. on pp. ix, 13).

[ABP14] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. *Removing Erasures with Explainable Hash Proof Systems*. Cryptology ePrint Archive, Report 2014/125. http://eprint.iacr.org/2014/125. 2014 (cit. on p. 10).

[ABP15a] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 388–409. DOI: 10.1007/978-3-662-47989-6_19 (cit. on pp. ix, 11).

[ABP15b] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved Security". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 103–120. DOI: 10.1007/978-3-662-48797-6_5 (cit. on pp. ix, 11).

[ABP15c] 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. 9, 11, 34, 37, 69, 76, 129, 136, 152, 153, 166).

[ABP15d] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 332–352. DOI: 10.1007/978-3-662-46447-2_15 (cit. on pp. 10, 13, 14, 188).

[ABP16] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Public-Key Encryption Indistinguishable Under Plaintext-Checkable Attacks". In: *IET Information Security* (2016). To appear. Full version of [ABP15d] (cit. on p. 10).

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. "Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 77–94. DOI: 10.1007/978-3-662-44371-2_5 (cit. on pp. ix, 11).

[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 pp. 7, 10, 48, 61, 69, 72, 178).

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. "Structure-Preserving Signatures and Commitments to Group Elements". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 209–236 (cit. on p. 155).

[AH91] William Aiello and Johan Hastad. "Relativized perfect zero knowledge is not BPP". In: *Information and Computation* 93.2 (1991), pp. 223–240 (cit. on p. 52).

[AMOR14] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. "Weakness of $\mathbb{F}_{3^{6509}}$ for Discrete Logarithm Cryptography". In: *PAIRING 2013*. Ed. by Zhenfu Cao and Fangguo Zhang. Vol. 8365. LNCS. Springer, Heidelberg, Nov. 2014, pp. 20–44. DOI: 10.1007/978-3-319-04873-4_2 (cit. on pp. 27, 28).

[Bab85] László Babai. "Trading Group Theory for Randomness". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. Ed. by Robert Sedgewick. ACM, 1985, pp. 421–429. DOI: 10.1145/22145.22192. URL: http://doi.acm.org/10.1145/22145.22192 (cit. on p. 3).

[BB04] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 223–238 (cit. on p. 159).

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. "Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 272–291. DOI: 10.1007/978-3-642-36362-7_18 (cit. on pp. 10, 33, 34).

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. *New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange*. Cryptology ePrint Archive, Report 2013/034. http://eprint.iacr.org/2013/034. 2013 (cit. on p. 9).

[BBC+13c] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. "New Techniques for SPHFs and Efficient One-Round PAKE Protocols". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 449–475. DOI: 10.1007/978-3-642-40041-4_25 (cit. on pp. 5, 9–11, 37, 82, 84, 92, 136, 137, 141, 160, 166, 170).

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. "Verified Proofs of Higher-Order Masking". In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 457–485. DOI: 10.1007/978-3-662-46800-5_18 (cit. on p. 12).

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. "Randomness Complexity of Private Circuits for Multiplication". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 616–648. DOI: 10.1007/978-3-662-49896-5_22 (cit. on pp. ix, 12).

[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 pp. 33, 34).

[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 pp. 34, 80).

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. "Better Zero-Knowledge Proofs for Lattice Encryption and Their Application to Group Signatures". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 551–572. DOI: 10.1007/978-3-662-45611-8_29 (cit. on p. 13).

[BCPW15] Fabrice Benhamouda, Geoffroy Couteau, David Pointcheval, and Hoeteck Wee. "Implicit Zero-Knowledge Arguments and Applications to the Malicious Setting". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9216. LNCS. Springer, Heidelberg, Aug. 2015, pp. 107–129. DOI: 10.1007/978-3-662-48000-7_6 (cit. on pp. 9, 72, 112, 144, 162–164, 171, 173).

[BCTV16] Fabrice Benhamouda, Céline Chevalier, Adrian Thillard, and Damien Vergnaud. "Easing Coppersmith Methods Using Analytic Combinatorics: Applications to Public-Key Cryptography with Weak Pseudorandomness". 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. 36–66. DOI: 10.1007/978-3-662-49387-8_3 (cit. on p. 14).

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. "Secret Handshakes from Pairing-Based Key Agreements". In: *2003 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2003, pp. 180–196 (cit. on p. 10).

[Ber06] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. Springer, Heidelberg, Apr. 2006, pp. 207–228 (cit. on pp. 27, 28).

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. "A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 1–16. DOI: 10.1007/978-3-642-55220-5_1 (cit. on pp. 27, 28).

[BGM+10] Jean-Luc Beuchat, Jorge E. González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves". In: *PAIRING 2010*. Ed. by Marc Joye, Atsuko Miyaji, and Akira Otsuka. Vol. 6487. LNCS. Springer, Heidelberg, Dec. 2010, pp. 21–39 (cit. on p. 28).

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". In: *TCC 2005*. Ed. by Joe Kilian. Vol. 3378. LNCS. Springer, Heidelberg, Feb. 2005, pp. 325–341 (cit. on p. 91).

[BHJL16] Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. "Efficient Cryptosystems From 2^k-th Power Residue Symbols". In: *Journal of Cryptology* (2016). ISSN: 1432-1378. DOI: 10.1007/s00145-016-9229-5. URL: http://dx.doi.org/10.1007/s00145-016-9229-5 (cit. on pp. 13, 16, 197).

[BHZ87] Ravi B Boppana, Johan Hastad, and Stathis Zachos. "Does co-NP have short interactive proofs?" In: *Information Processing Letters* 25.2 (1987), pp. 127–132 (cit. on p. 52).

[BJL13] Fabrice Benhamouda, Marc Joye, and Benoît Libert. "Method for determining a statistic value on data based on encrypted data". Corresponds to [BJL16]. 2013 (cit. on p. 11).

[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 pp. 11, 16, 190).

[BK05] Dan Boneh and Jonathan Katz. "Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption". In: *CT-RSA 2005*. Ed. by Alfred Menezes. Vol. 3376. LNCS. Springer, Heidelberg, Feb. 2005, pp. 87–103 (cit. on p. 159).

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. "Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings". In: *ESORICS 2015, Part I*. Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. LNCS. Springer, Hei-

delberg, Sept. 2015, pp. 305–325. DOI: 10.1007/978-3-319-24174-6_16 (cit. on p. 13).

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. "Direct Chosen Ciphertext Security from Identity-Based Techniques". In: *ACM CCS 05*. Ed. by Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels. ACM Press, Nov. 2005, pp. 320–329 (cit. on pp. 156, 159).

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order". In: *SAC 2005*. Ed. by Bart Preneel and Stafford Tavares. Vol. 3897. LNCS. Springer, Heidelberg, Aug. 2006, pp. 319–331 (cit. on p. 27).

[BP13a] Fabrice Benhamouda and David Pointcheval. *Trapdoor Smooth Projective Hash Functions*. Cryptology ePrint Archive, Report 2013/341. http://eprint.iacr.org/2013/341. 2013 (cit. on p. 9).

[BP13b] Fabrice Benhamouda and David Pointcheval. *Verifier-Based Password-Authenticated Key Exchange: New Models and Constructions*. Cryptology ePrint Archive, Report 2013/833. http://eprint.iacr.org/2013/833. 2013 (cit. on p. 10).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. "Authenticated Key Exchange Secure against Dictionary Attacks". In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 139–155 (cit. on p. 10).

[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. 5).

[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. 26).

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93*. Ed. by V. Ashby. ACM Press, Nov. 1993, pp. 62–73 (cit. on p. 163).

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd FOCS*. IEEE Computer Society Press, Oct. 2001, pp. 136–145 (cit. on pp. 10, 39, 42).

[CC04] Christian Cachin and Jan Camenisch, eds. *EUROCRYPT 2004*. Vol. 3027. LNCS. Springer, Heidelberg, May 2004.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Heidelberg, Aug. 1994, pp. 174–187 (cit. on pp. 5, 138, 139).

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. "Optimal Randomness Extraction from a Diffie-Hellman Element". In: *EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, Heidelberg, Apr. 2009, pp. 572–589 (cit. on pp. 35, 62, 175).

[CG13a] Ran Canetti and Juan A. Garay, eds. *CRYPTO 2013, Part I*. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013.

[CG13b] Ran Canetti and Juan A. Garay, eds. *CRYPTO 2013, Part II*. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. "Zeroizing Without Low-Level Zeroes: New MMAP Attacks and their Limitations". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 247–266. DOI: 10.1007/978-3-662-47989-6_12 (cit. on pp. 11, 89).

[CGV15] Aloni Cohen, Shafi Goldwasser, and Vinod Vaikuntanathan. "Aggregate Pseudorandom Functions and Connections to Learning". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 61–89. DOI: 10.1007/978-3-662-46497-7_3 (cit. on p. 11).

[Cha82] David Chaum. "Blind Signatures for Untraceable Payments". In: *CRYPTO'82*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Plenum Press, New York, USA, 1982, pp. 199–203 (cit. on pp. 4, 5).

[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. 10).

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. "Chosen-Ciphertext Security from Identity-Based Encryption". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 207–222 (cit. on pp. 156, 159).

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. "Cryptanalysis of the Multilinear Map over the Integers". In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 3–12. DOI: 10.1007/978-3-662-46800-5_1 (cit. on pp. 11, 89).

[CLR15] Jung Hee Cheon, Changmin Lee, and Hansol Ryu. *Cryptanalysis of the New CLT Multilinear Maps*. Cryptology ePrint Archive, Report 2015/934. http://eprint.iacr.org/2015/934. 2015 (cit. on pp. 11, 89).

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "Practical Multilinear Maps over the Integers". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 476–493. DOI: 10.1007/978-3-642-40041-4_26 (cit. on p. 89).

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "New Multilinear Maps Over the Integers". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 267–286. DOI: 10.1007/978-3-662-47989-6_13 (cit. on p. 89).

[Cop96a] Don Coppersmith. "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 178–189 (cit. on p. 14).

[Cop96b] Don Coppersmith. "Finding a Small Root of a Univariate Modular Equation". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 155–165 (cit. on p. 14).

[Cra05] Ronald Cramer, ed. *EUROCRYPT 2005*. Vol. 3494. LNCS. Springer, Heidelberg, May 2005.

[Cra97] Ronald Cramer. "Modular design of secure yet practical cryptographic protocols". 1997 (cit. on pp. 5, 138, 139).

[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. ix, 4, 6, 7, 33, 48–50, 55, 57, 85–87, 102, 112, 177).

[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. 10, 32, 33, 57, 153, 156, 157, 159).

[CW13] Jie Chen and Hoeteck Wee. "Fully, (Almost) Tightly Secure IBE and Dual System Groups". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 435–460. DOI: 10.1007/978-3-642-40084-1_25 (cit. on p. 150).

[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 p. 1).

[DMP90] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. "Non-Interactive Zero-Knowledge with Preprocessing". In: *CRYPTO'88*. Ed. by Shafi Goldwasser. Vol. 403. LNCS. Springer, Heidelberg, Aug. 1990, pp. 269–282 (cit. on p. 163).

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, eds. *TCC 2015, Part II*. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015.

[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. 11, 34, 80, 81, 126).

[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 p. 31).

[Firefox Sync] *Firefox Sync*. URL: https://www.mozilla.org/en-US/firefox/sync/ (cit. on p. 13).

[For87] Lance Fortnow. "The Complexity of Perfect Zero-Knowledge (Extended Abstract)". In: *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 204–209 (cit. on p. 52).

[Fra04] Matthew Franklin, ed. *CRYPTO 2004*. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194 (cit. on p. 163).

[GG14a] Juan A. Garay and Rosario Gennaro, eds. *CRYPTO 2014, Part I*. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014.

[GG14b] Juan A. Garay and Rosario Gennaro, eds. *CRYPTO 2014, Part II*. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014.

[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. 179).

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. "Candidate Multilinear Maps from Ideal Lattices". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 1–17. DOI: 10.1007/978-3-642-38348-9_1 (cit. on pp. 11, 63, 89, 90, 92, 139).

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. "Graph-Induced Multilinear Maps from Lattices". In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 498–527. DOI: 10.1007/978-3-662-46497-7_20 (cit. on p. 89).

[GGMZ13] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. "On the Function Field Sieve and the Impact of Higher Splitting Probabilities — Application to Discrete Logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 109–128. DOI: 10.1007/978-3-642-40084-1_7 (cit. on pp. 27, 28).

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. "Witness encryption and its applications". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 467–476 (cit. on pp. 6, 52, 162, 179).

[GK10] Adam Groce and Jonathan Katz. "A new framework for efficient password-based authenticated key exchange". In: *ACM CCS 10*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM Press, Oct. 2010, pp. 516–525 (cit. on pp. 6, 10).

[GKZ14] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. "Breaking '128-bit Secure' Supersingular Binary Curves - (Or How to Solve Discrete Logarithms in $F_{2^{4\cdot 1223}}$ and $F_{2^{12\cdot 367}}$)". In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 126–145. DOI: 10.1007/978-3-662-44381-1_8 (cit. on p. 28).

[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. 129).

[GL06] Rosario Gennaro and Yehuda Lindell. "A Framework for Password-Based Authenticated Key Exchange". In: *ACM Transactions on Information and System Security* 9.2 (2006), pp. 181–234 (cit. on pp. 6, 10, 48, 50, 82, 129).

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. "Witness Encryption from Instance Independent Assumptions". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 426–443. DOI: 10.1007/978-3-662-44371-2_24 (cit. on p. 179).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information". In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*. Ed. by Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber. ACM, 1982, pp. 365–377. ISBN: 0-89791-067-2. DOI: 10.1145/800070.802212. URL: http://doi.acm.org/10.1145/800070.802212 (cit. on p. 2).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption". In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299 (cit. on p. 13).

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. Ed. by Robert Sedgewick. ACM, 1985, pp. 291–304. DOI: 10.1145/22145.22178. URL: http://doi.acm.org/10.1145/22145.22178 (cit. on p. 3).

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design". In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 171–185 (cit. on pp. 8, 163).

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs That Yield Nothing But Their Validity Or All Languages in NP Have Zero-Knowledge Proof Systems". In: *Journal of the ACM* 38.3 (1991), pp. 691–729 (cit. on p. 3).

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. "Strengthening Zero-Knowledge Protocols Using Signatures". In: *Journal of Cryptology* 19.2 (Apr. 2006), pp. 169–209 (cit. on pp. 42, 141).

[GR15] Rosario Gennaro and Matthew J. B. Robshaw, eds. *CRYPTO 2015, Part I*. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015.

[GS08] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 415–432 (cit. on pp. 142, 155, 156, 158, 163, 171).

[GS86] Shafi Goldwasser and Michael Sipser. "Private Coins versus Public Coins in Interactive Proof Systems". In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. Ed. by Juris Hartmanis. ACM, 1986, pp. 59–68. ISBN: 0-89791-193-8. DOI: 10.1145/12130.12137. URL: http://doi.acm.org/10.1145/12130.12137 (cit. on p. 3).

[GSV98] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. "Honest-Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge". In: *30th ACM STOC*. ACM Press, May 1998, pp. 399–408 (cit. on p. 52).

[Gui13] Aurore Guillevic. "Arithmetic of pairings on algebraic curves for cryptography". PhD thesis. École normale supérieure, Paris, Dec. 2013. URL: https://tel.archives-ouvertes.fr/tel-00921940 (cit. on p. 28).

[Helios] *Helios*. https://vote.heliosvoting.org (cit. on p. 1).

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. "A pseudorandom generator from any one-way function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396 (cit. on p. 35).

[HJ15] Yupu Hu and Huiwen Jia. *Cryptanalysis of GGH Map*. Cryptology ePrint Archive, Report 2015/301. http://eprint.iacr.org/2015/301. 2015 (cit. on pp. 11, 89).

[HK07] Dennis Hofheinz and Eike Kiltz. "Secure Hybrid Encryption from Weakened Key Encapsulation". In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. LNCS. Springer, Heidelberg, Aug. 2007, pp. 553–571 (cit. on pp. 126, 129).

[HKE13] Yan Huang, Jonathan Katz, and David Evans. "Efficient Secure Two-Party Computation Using Symmetric Cut-and-Choose". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 18–35. DOI: 10.1007/978-3-642-40084-1_2 (cit. on p. 164).

[HR10] Feng Hao and Peter Ryan. "J-PAKE: Authenticated Key Exchange without PKI". English. In: *Transactions on Computational Science XI*. Ed. by MarinaL. Gavrilova, C.J.Kenneth Tan, and EdwardDavid Moreno. Vol. 6480. Lecture Notes in Computer Science. LNCS, 2010, pp. 192–206. ISBN: 978-3-642-17696-8. DOI: 10.1007/978-3-642-17697-5_10 (cit. on p. 13).

[IC15] Tetsu Iwata and Jung Hee Cheon, eds. *ASIACRYPT 2015, Part I*. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. "Black-box constructions for secure computation". In: *38th ACM STOC*. Ed. by Jon M. Kleinberg. ACM Press, May 2006, pp. 99–108 (cit. on p. 164).

[Ish11] Yuval Ishai, ed. *TCC 2011*. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 463–481 (cit. on pp. 12, 13).

[JBL14] Marc Joye, Fabrice Benhamouda, and Benoît Libert. "Method and device for cryptographic key generation". Corresponds to [BHJL16]. 2014 (cit. on p. 13).

[JG04] Shaoquan Jiang and Guang Gong. "Password Based Key Exchange with Mutual Authentication". In: *SAC 2004*. Ed. by Helena Handschuh and Anwar Hasan. Vol. 3357. LNCS. Springer, Heidelberg, Aug. 2004, pp. 267–279 (cit. on pp. 6, 10).

[Jou09] Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009 (cit. on pp. 95, 97).

[Jou14] Antoine Joux. "A New Index Calculus Algorithm with Complexity $L(1/4 + o(1))$ in Small Characteristic". In: *SAC 2013*. Ed. by Tanja Lange, Kristin Lauter, and Petr Lisonek. Vol. 8282. LNCS. Springer, Heidelberg, Aug. 2014, pp. 355–379. DOI: 10.1007/978-3-662-43414-7_18 (cit. on pp. 27, 28).

[JR12] Charanjit S. Jutla and Arnab Roy. "Relatively-Sound NIZKs and Password-Based Key-Exchange". In: *PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. LNCS. Springer, Heidelberg, May 2012, pp. 485–503 (cit. on p. 158).

[JR13] Charanjit S. Jutla and Arnab Roy. "Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces". In: *ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013, pp. 1–20. DOI: 10.1007/978-3-642-42033-7_1 (cit. on pp. 8, 39, 40, 158).

[JR14] Charanjit S. Jutla and Arnab Roy. "Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces". In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 295–312. DOI: 10.1007/978-3-662-44381-1_17 (cit. on pp. 8, 133, 153, 158).

[JR15] Charanjit S. Jutla and Arnab Roy. "Dual-System Simulation-Soundness with Applications to UC-PAKE and More". In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Heidelberg, Nov. 2015, pp. 630–655. DOI: 10.1007/978-3-662-48797-6_26 (cit. on pp. 10, 11).

[KD04] Kaoru Kurosawa and Yvo Desmedt. "A New Paradigm of Hybrid Encryption Scheme". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 426–442 (cit. on p. 159).

[KH13] Kaoru Kurosawa and Goichiro Hanaoka, eds. *PKC 2013*. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013.

[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 pp. 156, 159).

[Kob87] Neal Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of computation* 48.177 (1987), pp. 203–209 (cit. on p. 27).

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 104–113 (cit. on p. 12).

[KOY09] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. "Efficient and secure authenticated key exchange using weak passwords". In: *Journal of the ACM* 57.1 (2009). DOI: 10.1145/1613676.1613679 (cit. on pp. 6, 10, 48, 82).

[KR01] Joe Kilian and Phillip Rogaway. "How to Protect DES Against Exhaustive Key Search (an Analysis of DESX)". In: *Journal of Cryptology* 14.1 (2001), pp. 17–35 (cit. on p. 26).

[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 pp. 33, 34).

[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 p. 178).

[KV11] Jonathan Katz and Vinod Vaikuntanathan. "Round-Optimal Password-Based Authenticated Key Exchange". In: *TCC 2011*. Ed. by Yuval Ishai. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011, pp. 293–310 (cit. on pp. 10, 50).

[KW15] Eike Kiltz and Hoeteck Wee. "Quasi-Adaptive NIZK for Linear Subspaces Revisited". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 101–128. DOI: 10.1007/978-3-662-46803-6_4 (cit. on pp. 8, 41, 80, 129, 152, 153).

[Lin13] Yehuda Lindell. "Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries". In: *CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. LNCS. Springer, Heidelberg, Aug. 2013, pp. 1–17. DOI: 10.1007/978-3-642-40084-1_1 (cit. on p. 164).

[LP07] Yehuda Lindell and Benny Pinkas. "An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries". In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Heidelberg, May 2007, pp. 52–78 (cit. on p. 164).

[LP11] Yehuda Lindell and Benny Pinkas. "Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer". In: *TCC 2011*. Ed. by Yuval Ishai. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011, pp. 329–346 (cit. on p. 164).

[LPJY14] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. "Non-malleability from Malleability: Simulation-Sound Quasi-Adaptive NIZK Proofs and CCA2-Secure Encryption from Homomorphic Signatures". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 514–532. DOI: 10.1007/978-3-642-55220-5_29 (cit. on pp. 153, 158).

[Mar16] Léo Ducas Martin Albrecht Shi Bai. *A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and Graded Encoding Schemes*. Cryptology ePrint Archive, Report 2016/127. http://eprint.iacr.org/. 2016 (cit. on pp. 11, 89).

[Mau96] Ueli M. Maurer, ed. *EUROCRYPT'96*. Vol. 1070. LNCS. Springer, Heidelberg, May 1996.

[MF15] Brice Minaud and Pierre-Alain Fouque. *Cryptanalysis of the New Multilinear Map over the Integers*. Cryptology ePrint Archive, Report 2015/941. http://eprint.iacr.org/2015/941. 2015 (cit. on pp. 11, 89).

[Mil86] Victor S. Miller. "Use of Elliptic Curves in Cryptography". In: *CRYPTO'85*. Ed. by Hugh C. Williams. Vol. 218. LNCS. Springer, Heidelberg, Aug. 1986, pp. 417–426 (cit. on p. 27).

[Nest] *Nest*. URL: http://nest.com (cit. on p. 13).

[NIS12] NIST. *FIPS PUB 180-4, Secure Hash Standard (SHS)*. 2012 (cit. on p. 29).

[NO14] Phong Q. Nguyen and Elisabeth Oswald, eds. *EUROCRYPT 2014*. Vol. 8441. LNCS. Springer, Heidelberg, May 2014.

[Odl87] Andrew M. Odlyzko, ed. *CRYPTO'86*. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987.

[OF15a] Elisabeth Oswald and Marc Fischlin, eds. *EUROCRYPT 2015, Part I*. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015.

[OF15b] Elisabeth Oswald and Marc Fischlin, eds. *EUROCRYPT 2015, Part II*. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015.

[OpenSSL] *OpenSSL Project*. URL: http://www.openssl.org (cit. on p. 13).

[Rab81] Michael Rabin. *How to exchange secrets with oblivious transfer*. Tech. rep. Technical Report TR-81. Aiken Computation Lab, Harvard University, 1981 (cit. on p. 6).

[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 p. 1).

[Sch90] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 239–252 (cit. on p. 138).

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (1991), pp. 161–174 (cit. on p. 14).

[Sed85] Robert Sedgewick, ed. *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. ACM, 1985.

[SG02] Victor Shoup and Rosario Gennaro. "Securing Threshold Cryptosystems against Chosen Ciphertext Attack". In: *Journal of Cryptology* 15.2 (2002), pp. 75–96 (cit. on p. 155).

[Sha07] Hovav Shacham. *A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants*. Cryptology ePrint Archive, Report 2007/074. http://eprint.iacr.org/2007/074. 2007 (cit. on pp. 126, 129).

[Sha71] Daniel Shanks. "Class number, a theory of factorization, and genera". In: *Proc. Symp. Pure Math.* Vol. 20. 1971, pp. 415–440 (cit. on p. 27).

[Sha79] Adi Shamir. "How to Share a Secret". In: *Communications of the Association for Computing Machinery* 22.11 (Nov. 1979), pp. 612–613 (cit. on p. 155).

[Sha92] Adi Shamir. "IP = PSPACE". In: *J. ACM* 39.4 (1992), pp. 869–877. DOI: 10.1145/146585.146609. URL: http://doi.acm.org/10.1145/146585.146609 (cit. on p. 3).

[Sho01] Victor Shoup. "OAEP Reconsidered". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 239–259 (cit. on pp. 24, 26).

[Sho06] *Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers*. Standard. Geneva, CH: International Organization for Standardization, 2006 (cit. on p. 29).

[sS11] abhi shelat and Chih-Hao Shen. "Two-Output Secure Computation with Malicious Adversaries". In: *EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. LNCS. Springer, Heidelberg, May 2011, pp. 386–405 (cit. on p. 164).

[SS13] Kazue Sako and Palash Sarkar, eds. *ASIACRYPT 2013, Part I*. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013.

[sS13] abhi shelat and Chih-Hao Shen. "Fast two-party secure computation with minimal assumptions". In: *ACM CCS 13*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 523–534 (cit. on p. 164).

[Thread] *Thread Protocol*. URL: http://www.threadgroup.org (cit. on p. 13).

[Wat05] Brent R. Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 114–127 (cit. on p. 52).

Résumé

Les *smooth* (ou *universal*) *projective hash functions* ont été introduites par Cramer et Shoup, à Eurocrypt'02, comme un outil pour construire des schémas de chiffrement efficaces et sûrs contre les attaques à chiffrés choisis. Depuis, elles ont trouvé de nombreuses applications, notamment pour la construction de schémas d'authentification par mot de passe, d'*oblivious transfer*, de signatures en blanc, et de preuves à divulgation nulle de connaissance. Elles peuvent êtres vues comme des preuves implicites d'appartenance à certains langages. Un problème important est de caractériser pour quels langages de telles fonctions existent.

Dans cette thèse, nous avançons dans la résolution de ce problème en proposant la notion de *diverse modules*. Un *diverse module* est une représentation d'un langage, comme un sous-module d'un module plus grand, un module étant un espace vectoriel sur un anneau. À n'importe quel *diverse module* est associée une *smooth projective hash function* pour le même langage. Par ailleurs, presque toutes les *smooth projective hash functions* actuelles sont construites de cette manière.

Mais les *diverse modules* sont aussi intéressants en eux-mêmes. Grâce à leur structure algébrique, nous montrons qu'ils peuvent facilement être combinés pour permettre de nouvelles applications, comme les preuves implicites à divulgation nulle de connaissance (une alternative légère aux preuves non-interactives à divulgation nulle de connaissance), ainsi que des preuves non-interactives à divulgation nulle de connaissance et *one-time simulation-sound* très efficaces pour les langages linéaires sur les groupes cycliques.

Abstract

Smooth (or universal) projective hash functions were first introduced by Cramer and Shoup, at Eurocrypt'02, as a tool to construct efficient encryption schemes, indistinguishable under chosen-ciphertext attacks. Since then, they have found many other applications, including password-authenticated key exchange, oblivious transfer, blind signatures, and zero-knowledge arguments. They can be seen as implicit proofs of membership for certain languages. An important question is to characterize which languages they can handle.

In this thesis, we make a step forward towards this goal, by introducing diverse modules. A diverse module is a representation of a language, as a submodule of a larger module, where a module is essentially a vector space over a ring. Any diverse module directly yields a smooth projective hash function for the corresponding language, and almost all the known smooth projective hash functions are constructed this way.

Diverse modules are also valuable in their own right. Thanks to their algebraic structural properties, we show that they can be easily combined to provide new applications related to zero-knowledge notions, such as implicit zero-knowledge arguments (a lightweight alternative to non-interactive zero-knowledge arguments), and very efficient one-time simulation-sound (quasi-adaptive) non-interactive zero-knowledge arguments for linear languages over cyclic groups.

Mots Clés

cryptographie, module, espace vectoriel, preuve à divulgation nulle de connaissance, *smooth projective hash function*, *hash proof system*

Keywords

cryptography, module, vector space, zero-knowledge, smooth projective hash function, hash proof system