THÈSE DE DOCTORAT

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

Préparée à l'École normale supérieure

Zero-Knowledge Proofs for Secure Computation

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité Informatique

Soutenue par
Geoffroy COUTEAU
le 30 novembre 2017

Dirigée par
David POINTCHEVAL
et **Hoeteck WEE**

COMPOSITION DU JURY

M. POINTCHEVAL David
École Normale Supérieure
Directeur de thèse

M. WEE Hoeteck
École Normale Supérieure
Directeur de thèse

M. DAMGÅRD Ivan
Aarhus University
Rapporteur

M. GROTH Jens
University College of London
Rapporteur

M. ISHAI Yuval
Technion
Examinateur

Mme. CHEVALIER Céline
Université Panthéon-Assas Paris II
Examinateur

M. LIBERT Benoît
École Normale Supérieure de Lyon
Président du Jury

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

Zero-Knowledge Proofs for Secure Computation

Geoffroy COUTEAU

Supervisors: David POINTCHEVAL and Hoeteck WEE

Résumé

Dans cette thèse, nous étudions les preuves à divulgation nulle de connaissance, une primitive cryptographique permettant de prouver une assertion en ne révélant rien de plus que sa véracité, et leurs applications au calcul sécurisé. Nous introduisons tout d'abord un nouveau type de preuves à divulgation nulle, appelées arguments implicites à divulgation nulle, intermédiaire entre deux notions existantes, les preuves interactives et les preuves non-interactives à divulgation nulle. Cette nouvelle notion permet d'obtenir les mêmes bénéfices en terme d'efficacité que les preuves non-interactives dans le contexte de la construction de protocoles de calcul sécurisé faiblement interactifs, mais peut être instanciée à partir des mêmes hypothèses cryptographiques que les preuves interactives, permettant d'obtenir de meilleures garanties d'efficacité et de sécurité. Dans un second temps, nous revisitons un système de preuves à divulgation nulle de connaissance qui est particulièrement utile dans le cadre de protocoles de calcul sécurisé manipulant des nombres entiers, et nous démontrons que son analyse de sécurité classique peut être améliorée pour faire reposer ce système de preuve sur une hypothèse plus standard et mieux connue. Enfin, nous introduisons une nouvelle méthode de construction de systèmes de preuves à divulgation nulle sur les entiers, qui représente une amélioration par rapport aux méthodes existantes, tout particulièrement dans un modèle de type client-serveur, où un client à faible puissance de calcul participe à un protocole de calcul sécurisé avec un serveur à forte puissance de calcul.

Abstract

In this thesis, we study zero-knowledge proofs, a cryptographic primitive that allows to prove a statement while yielding nothing beyond its truth, and their applications to secure computation. Specifically, we first introduce a new type of zero-knowledge proofs, called implicit zero-knowledge arguments, that stands between two existing notions, interactive zero-knowledge proofs and non-interactive zero-knowledge proofs. Our new notion provides the same efficiency benefits as the latter when used to design round-efficient secure computation protocols, but it can be built from essentially the same cryptographic assumptions as the former, which leads to improved efficiency and security guarantees. Second, we revisit a zero-knowledge proof system that is particularly useful for secure computation protocols manipulating integers, and show that the known security analysis can be improved to base the proof system on a more well-studied assumption. Eventually, we introduce a new method to build zero-knowledge proof systems over the integers, which particularly improves over existing methods in a client-server model, where a weak client executes a secure computation protocol with a powerful server.

Acknowledgments

I have been struggling with this acknowledgment section for quite a time now. Writing a thesis takes some time, but it is made easier by the comfortable feeling that almost no one will ever read it, that all my inaccuracies, stylistically inappropriate sentences, and mistakes, are safely locked behind a wall of seemingly obscure mathematics. This is unfortunately not the case with the acknowledgment section, where I can expect to have a bunch of readers wondering why it has been already almost an entire paragraph, and yet they have *still not seen their name*. A noticeable proportion of those same readers should probably be listening to my presentation right now, if only to find out whether I'm almost done and they'll soon be able to drink champagne.

Anyway, it's now time to say thank you. There are many people I want to thank, and many more that I should thank, but will forget to. I have been staring at my screen for about five minutes already, trying to find an appropriate way to formulate it without success, so let's just put it simply: I apologize to all people that should have been mentioned there, but that I forgot. I cannot be completely sure because I cannot remember you right now, but I probably like you very much and would definitely include you in the acknowledgments, if it weren't for my bad memory. Also, a little bit of cash does usually a good job at refreshing my memory, just saying.

First of all, I want to thank David Pointcheval. I cannot express how grateful I am to you. Your guidance, from my master internship to my PhD thesis, has been invaluable, and you've always done a perfect job at encouraging me when I had ideas, helping me when I didn't, teaching me when I had questions, being friendly when I didn't, and pushing me to concentrate on the important things when I was trying to pursue all directions at once (and especially the useless ones). Working and interacting with you has always been a pleasure, and I'm constantly impressed at how you somehow manage to dedicate time to all students while dealing with a thousand other things in parallel, without ever looking overwhelmed by work – I've tried to sum the amount of time taken by all tasks you handle daily, but it always ends up way above 24 hours. I feel extremely lucky that I was supervised by you.

I am also very grateful to Hoeteck Wee. I always admire your intelligence and the sharpness of your remarks, and I enjoyed our discussions. Your advices have been precious to me, and you have an impressive high level view of things – probably because you're often looking at them from the window of a plane.

I want to thank the researchers with whom I had many wonderful discussions about crypto, complexity theory, or science in general. These discussions shaped my taste for theoretical computer science, and gave me a clear view of what I love about research and which goals I want to pursue. I am particularly thankful for that to Yuval Ishai, Pooya Farshim, and Chris Brzuska. I am looking forward to get more occasions to discuss or debate with you. I also had thrilling discussions with Alain Passelègue, even though I'm not sure that I entirely agree with his point that "no one ever prove non-trivial results in crypto, we cryptographers are just not good enough to do hard stuff".

I am grateful to Ivan Damgård and Jens Groth, who agreed to perform the tedious task of

reviewing my thesis, in spite of their busy schedules. I also want to thank Céline Chevalier, Jens Groth, Benoît Libert, and Yuval Ishai, who accepted to be part of my thesis jury.

I want to thank the researchers who gave me the opportunity to visit them and their team: Jens Groth, for inviting me for one week at UCL in November 2015, Ivan Damgård, for inviting me in Aarhus to give a talk at the TPMPC 2016 workshop, Melissa Chase, for inviting me for one day at Microsoft Research when I was in Seattle in August 2016 (I also want to thank Josh Benaloh, with whom I had insightful discussions on e-voting there), Yuval Ishai and Amit Sahai, for inviting me for two weeks at UCLA in May 2017, and Dennis Hofheinz, who invited me twice at KIT so that I could get to meet my future colleagues. I also want to thank Dennis for offering me a postdoctoral position at KIT, I am already convinced that I will have a wonderful time there.

I would also like to thank my coauthors, without whom my research attempts would have been far less productive: Fabrice Benhamouda, Elette Boyle, Pyrros Chaidos, Niv Gilboa, Yuval Ishai, Michele Orrù, Thomas Peters, David Pointcheval, and Hoeteck Wee. I already expressed the depth of my gratitude to David and Hoeteck; among my coauthors, I also want to insist on the invaluable help and encouragements that I received from Fabrice Benhamouda (who is simultaneously a patient teacher, and an impressive problem solver), Thomas Peters (whose way of thinking and approach to research I really enjoyed), and Yuval Ishai (Hoeteck once told me "it's highly recommanded to talk to Yuval when you get occasions to, his view on things is extremely insightful". I could not agree more with this advice, and I'd like to add that it's impressive that Yuval seems to find time to share his fantastic way of thinking with many people around while at the same time writing eight papers at once). Getting to meet the three of you has been the core milestones toward becoming a researcher, during each of my three years of phd.

Having fun, hanging around, talking, and drinking coffee with colleagues is an important part of a phd. For that, I want to thank my colleagues at ENS: Michel Abdalla, Balthazar Bauer, Sonia Belaïd, Fabrice Benhamouda, Raphaël Bost, Florian Bourse, Céline Chevalier, Jérémy Chotard, Simon Cogliani, Mario Cornejo, Edouard Dufour Sans, Angelo De Caro, Rafaël Del Pino, Itai Dinur, Aurélien Dupin, Pierre-Alain Dupont, Pooya Farshim, Houda Ferradi, Georg Fuchsbauer, Romain Gay, Rémi Géraud, Dahmun Goudarzi, Giuseppe Guagliardo, Chloé Hébant, Julia Hesse, Duong Hieu Phan, Louiza Khati, Tancrède Lepoint, Baptiste Louf, Vadim Lyubashevsky, Pierrick Méaux, Thierry Mefenza, Michele Minelli, David Naccache, Anca Nitulescu, Michele Orrù, Alain Passelègue, Thomas Peters, David Pointcheval, Thomas Prest, Răzvan Roşie, Mélissa Rossi, Sylvain Ruhault, Olivier Sanders, Antonia Schmidt-Lademann, Adrian Thillard, Bogdan Ursu, Damien Vergnaud, and Hoeteck Wee. A special thanks goes to Florian, Rafaël, Pierrick, Romain, and Rémi, with whom I spent most of my time during conferences (for purely work-related reasons, obviously). I had also a lot of fun hanging around with Alain, Dahmun, Aurélien, the true Michele, the other Michele, Adrian, and Anca.

I would like to thank the administrative staff of the DI, and the members of the SPI: Jacques Beigbeder, Lise-Marie Bivard, Isabelle Delais, Nathalie Gaudechoux, Joëlle Isnard, Valérie Mongiat, Ludovic Ricardou, and Sophie Jaudon.

Looking further away in the past, I want to express my deepest gratitude to two wonderful people, who paved my way to mathematics and to research. The first of them is my grandfather, a crazy astronomer who enjoyed calculating integrals for fun, or playing the game "try to compute this square root on a calculator before I find the result by head". He would have loved to see the grandson he spent hours explaining trigonometry and logarithms

to during holiday evenings doing a phd in science. He is the one who taught me the beauty of mathematics. The second is Pierre-François Berger, my math teacher from preparatory class for entrance to Grandes Ecoles, another crazy person with a wonderful sense for the beauty of mathematics who, in spite of all my laziness, never stopped to believe that I could succeed in math. Without his passion and his tireless support, I would never have believed it myself.

Looking even further away in the past, I have been blessed from the start with a wonderful family, without which none of this would have been possible. I owe so much to my parents that I could never express it with words. They always maintained a perfect balance between giving me the freedom of building my own experiences, and pushing me toward doing what was right, to ensure that I would not miss opportunities nor have regrets. They also managed to accept my strange tastes, especially for music, which was not an obvious task. I also want to thank my brothers and sisters for their love and support. And the dog. I'm from a family where it's important to mention the dog, too. Thank you, dog.[1]

I cannot name all friends who made these last three years so fun and enjoyable, if only because I would forget to mention people that counted a lot for me. I still want to thank two people in particular, Patrick Lambein and Geoffrey Poncelet, for too many reasons to list them.

Last and above all, I want to thank Camille Jandot. Being by your side for the last four years has been the best thing that ever happened to me, and I could not imagine life without you. I am also grateful to all the efforts you made to bear my tendencies to constantly talk about crypto.

[1] Dog's name is Baltique.

Contents

Résumé iii

Abstract v

Acknowledgments vii

1 Introduction 1
- 1.1 Provable Security . . . 2
- 1.2 Proofs in Cryptography . . . 3
  - 1.2.1 Zero-Knowledge Proofs . . . 4
- 1.3 Secure Two-Party Computation . . . 5
  - 1.3.1 Active Security and Passive Security . . . 5
- 1.4 Our Results . . . 6
  - 1.4.1 Implicit Zero-Knowledge Arguments . . . 6
  - 1.4.2 Improved Security Analysis of Integer Zero-Knowledge Arguments . . . 8
  - 1.4.3 New Integer Zero-Knowledge Arguments . . . 8
- 1.5 Our Other Contributions . . . 9
  - 1.5.1 Encryption Switching Protocols [CPP16; CPP15b] . . . 9
  - 1.5.2 Homomorphic Secret Sharing [BCG+17] . . . 9
  - 1.5.3 Secure Equality Tests and Comparisons [Cou16a] . . . 10
  - 1.5.4 Covert Multiparty Computation [Cou16b] . . . 10
- 1.6 Organization of this Thesis . . . 11

2 Preliminaries 13
- 2.1 Notation and Preliminaries . . . 14
  - 2.1.1 Mathematical Notations . . . 14
  - 2.1.2 Algorithms . . . 15
  - 2.1.3 Provable Security . . . 16
- 2.2 Computational Assumptions . . . 17
  - 2.2.1 Discrete-Logarithm-Based Assumptions . . . 18
  - 2.2.2 Factorization-Based Assumptions . . . 22
- 2.3 Cryptographic Primitives . . . 26
  - 2.3.1 One-Way Functions . . . 26
  - 2.3.2 Commitment Schemes . . . 26
  - 2.3.3 Public-Key Encryption Schemes . . . 29
  - 2.3.4 Smooth Projective Hash Functions . . . 30

3 Zero-Knowledge Proofs and Arguments 35
- 3.1 Interactive Proofs . . . 37
  - 3.1.1 Definitions . . . 37
  - 3.1.2 Historical Notes . . . 38

3.2 Interactive Zero-Knowledge Proofs . . . 39
3.2.1 Definitions . . . 39
3.2.2 Brief Survey of Known Results . . . 42
3.3 Interactive Zero-Knowledge Arguments . . . 43
3.3.1 Definitions . . . 43
3.3.2 Historical Notes . . . 44
3.4 Proofs and Arguments of Knowledge . . . 44
3.4.1 Definitions . . . 45
3.5 Σ-Protocols . . . 46
3.5.1 Definition . . . 46
3.6 The Common Reference String Model . . . 50
3.6.1 Trusted Setup Assumptions . . . 51
3.6.2 Proving Security of Zero-Knowledge Proof Systems . . . 51
3.6.3 Simulation Soundness . . . 53
3.7 Zero-Knowledge Arguments over the Integers . . . 53
3.7.1 Zero-Knowledge Proofs of Non-Algebraic Statements . . . 53
3.7.2 Range Proofs . . . 54
3.7.3 Zero-Knowledge Arguments from Diophantine Relations . . . 54
3.8 Non-Interactive Zero-Knowledge Arguments . . . 55
3.8.1 Definition and Security Properties . . . 55
3.8.2 Brief Survey of Known Results . . . 56
3.8.3 Fiat-Shamir Heuristic . . . 57
3.8.4 Groth-Sahai Proofs . . . 57

4 Implicit Zero-Knowledge Arguments 59
4.1 Introduction . . . 60
4.1.1 Enforcing Honest Behavior in Two-Party Computation . . . 60
4.1.2 On Round-Efficiency . . . 61
4.1.3 Contributions of this Chapter . . . 62
4.1.4 New Notion: Implicit Zero-Knowledge Arguments . . . 62
4.1.5 Overview of our iZK and SSiZK Constructions . . . 63
4.1.6 Related Work . . . 65
4.2 Definition of Implicit Zero-Knowledge Arguments . . . 66
4.2.1 Definition . . . 66
4.2.2 Security Requirements . . . 67
4.3 Construction of Implicit Zero-Knowledge Arguments . . . 69
4.3.1 Limitations of Smooth Projective Hash Functions . . . 69
4.3.2 iZK Construction . . . 70
4.3.3 Notes . . . 73
4.3.4 Proof of Security . . . 73
4.3.5 SSiZK Construction . . . 75
4.3.6 Proof of Security . . . 75
4.3.7 More Efficient iZK Constructions . . . 78
4.3.8 iZK for Languages Defined by a Computational Structure . . . 80
4.4 Applications . . . 85
4.4.1 Semi-Honest to Malicious Transformation . . . 85
4.4.2 Secure Computation of Inner Products . . . 87

4.4.3 Details on the Inner Product Protocols . . . 88
4.4.4 Proof of Security . . . 94

5 Zero-Knowledge Arguments over the Integers under the RSA Assumption 99
5.1 Introduction . . . 100
5.1.1 Our Contribution . . . 101
5.1.2 Related Works . . . 101
5.1.3 Organization . . . 102
5.2 Commitment of Integers Revisited . . . 102
5.2.1 Commitments over the Integers . . . 102
5.2.2 Zero-Knowledge Argument of Opening . . . 103
5.3 Proof of Theorem 5.2.1 . . . 106
5.3.1 Preliminaries . . . 106
5.3.2 Detailed Proof . . . 107
5.4 Classical Extensions and Applications . . . 109
5.4.1 Generalized Commitment of Integers . . . 110
5.4.2 Zero-Knowledge Argument of Opening . . . 110
5.4.3 Equally Efficient Range Proofs from RSA . . . 111
5.4.4 A Correction on Lipmaa's Argument for Positivity . . . 111

6 More Efficient Zero-Knowledge Arguments over the Integers 115
6.1 Introduction . . . 115
6.1.1 Our Method in a Nutshell . . . 115
6.1.2 Organization . . . 116
6.2 Commitment with Knowledge-Delayed Order . . . 116
6.2.1 RSA-based Commitments with Known Order . . . 116
6.2.2 Commitment with Knowledge-Delayed Order . . . 117
6.2.3 Improving Zero-Knowledge Arguments over the Integers . . . 118
6.3 Application to Range Proofs . . . 122
6.3.1 Lipmaa's Compact Argument for Positivity . . . 122
6.3.2 Three-Square Range Proof . . . 122
6.3.3 Results . . . 123

7 Conclusion and Open Questions 127
7.1 Conclusion . . . 127
7.2 Open Questions . . . 127

Notation 131

Abbreviations 133

List of Illustrations 135
Figures . . . 135
Tables . . . 136

Bibliography 137

1

Introduction

Historically, cryptography has been concerned with the design of ciphers, or *encryption schemes*, with the goal of permitting secure communication between distant parties in the presence of an eavesdropper. In the last half-century, however, its purpose evolved considerably. Nowadays, cryptography is concerned with the broad task of designing methods that allow to realize some specified functionality, while preventing malicious abuse of the functionality. This imprecise definition calls for a better understanding of what we can know, or assume, about the behavior of a malicious user. We could think at first sight that we could design perfectly secure cryptographic schemes, for which malicious abuse would be provably impossible. However, the seminal work of Shannon on the theory of information has made this hopeless. Indeed, Shannon proved the following:

For any cipher that perfectly hides all information about the message, the secret key used to encrypt the message must be at least as long as the message itself.

Therefore, if one aims at constructing cipher that will provably withstand all malicious behaviors, this theorem considerably limits the efficiency of the cryptographic systems that one could design: for *each* message that must be transmitted securely, a key of the same length must have been securely exchanged first.

So, is security impossible to achieve? It is, if we do not want to assume anything about the adversary. However, if we are willing to make assumptions, we could hope to prove that even though no efficient cipher can hide all informations about a message, it would still be infeasible for an adversary with limited abilities to *extract* this information from the ciphertext. This raises the following question: which kind of limitation can we assume regarding an adversary?

It can be tempting to estimate that an attacker against a scheme will have to follow some natural strategy, and to design the scheme so as to make such natural strategies inefficient. However, this has long been known to be an unrealistic estimation, and is commonly illustrated by a famous quote of Bruce Schneier, which is known as Schneier's law:

'Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break.'

The crux of the work of a cryptographer, therefore, is to come up with algorithms that not

only he himself cannot break, but that will withstand even malicious strategies that he had not anticipated at the time of their design – even strategies imagined after the scheme was made public. And for this task, we need more realistic assumptions regarding the limitations of a malicious user. This is best put through the words of Oded Goldreich, in the preface of his book Foundations of Cryptography [Gol06]:

'We believe that it makes little sense to make assumptions regarding the specific *strategy* that an adversary may use. The only assumptions that can be justified refer to the computational *abilities* of the adversary.'

Without assuming anything about the strategies that a party will come up with, we are left with using the fact that this adversary necessarily has limited *resources*. Typically, it is very reasonable to assume that no user will be able to perform an immensely long computation. This guides the design and defines the goals of the security analysis of any cryptographic scheme: any malicious behavior that deviates from the specified purpose of the scheme should *provably* require too much resources to be mounted in practice.

1.1 Provable Security

> *The best scientists can do is fail to disprove things while pointing to how hard they tried.*
>
> – Richard Dawkins

For any cryptographic scheme to provide meaningful security guarantees, abusing the scheme should be a computationally difficult task. This requires, at the very least, that there *exist* computationally difficult tasks. In the theoretical study of algorithms, tasks that can be performed efficiently (in a computational sense) are formally defined as algorithms whose running time grows at most polynomially with the size of their inputs, which is known as the complexity class P. Therefore, to *prove* that a cryptographic scheme satisfies some security property, the cryptographer must show that no algorithm running in polynomial time can be able to perform the task of breaking this security property.

Unfortunately, we do not know how to do this. Broadly, tasks of interest for the design of functionalities are those that attempt to find a solution to a given problem, so that when a solution is found, anyone can efficiently verify that a valid solution has been found. The complexity class of all such problems is known as the class NP. The question of whether this class contains any problem that *cannot* be solved efficiently is the deepest and most intriguing open question of the theory of computation. In lack of a breakthrough providing insights regarding this incredibly hard question, we are left with assuming that such problems exist, and that specific problems that we know are of this kind. The choice of these problems can only be based on our trust in the fact that a long-standing resistance to all attempts of experts to design efficient algorithms solving these problems is an indication of security.

When proposing a new construction, a cryptographer could hope that it will attract attention from the cryptographic community, and will undergo a careful scrutiny, so that after some time, the scheme can be deemed secure. This is, however, hopeless in regard of the growing number of new cryptographic constructions that appear each year. The goal of provable security is to cope with this issue: rather than hoping that the security of a new scheme will eventually become a well-established assumption, we seek to show

that the security of the scheme is, in fact, already captured by the strengths of an already well-established assumption. This is done by the mean of cryptographic reduction: a proof of security under a well-established assumption A is a proof that, if there is any adversary $\mathscr{A}$ that can abuse the scheme in polynomial time, then there must exist a polynomial adversary that will contradict the assumption A. Typically, such reductions are constructive, providing an explicit adversary $\mathscr{A}'$ that runs $\mathscr{A}$ internally to break the assumption A. Therefore, provable security creates a network of seemingly unrelated primitives achieving very different goals, interconnected by their security reduction to a set of well-established assumptions. The foundations of this vast program have been laid in [GM82], and the network of new constructions has been constantly evolving ever since.

1.2 Proofs in Cryptography

We mentioned above that the complexity class NP refers to the problems for which solutions can be verified efficiently. Another way of stating it is as follows: NP is the class of all statements for the truth of which a *short* proof exists, and can be verified in *reasonable* time. Here, short and reasonable mean of size and with a verification running time which are bounded by a polynomial in the size of the statement. Hence, NP can be seen as the set of all assertions that can be proven efficiently (without considering how hard it might be to actually *find* the proof). This makes NP an object of deep philosophical interest, as the question of whether P is contained in NP fundamentally asks the following: if we can efficiently verify a mathematical proof for a theorem, can we just as easily *find* this proof? Or, as put by Scott Aaronson in his book [Aar13], could mathematical creativity be automated?

Proofs are at the heart of cryptography. While it is fairly easy to come up with advanced algorithms realizing tasks of interest as long as the parties running the algorithm can be trusted, cryptography asks for the design of solutions where one does not have to trust his adversaries. The necessity for proofs in this setting emerges quite obviously: if we do not plan to trust a statement made by an adversary, it is natural to ask him for a proof of the truth of this statement. The class NP, however, captures only a limited 'static' type of proofs: those that consist in a deterministic demonstration written once for all, that can be efficiently verified. To get more expressiveness or better efficiency, it is useful to relax this definition, by allowing proofs to be *probabilistic* (i.e., to let them depend on coins flipped by the parties) and *interactive* (i.e., to give the verifier of the proof the opportunity to ask questions). This leads to the study of interactive proofs, which has been extremely fruitful and produced some of the most successful results in complexity.

Fundamentally, a proof reveals whether a statement is true. In 1989, Goldwasser, Micali, and Rackoff [GMR89] raised a question that turned out to be of great interest for cryptography: what *else* is revealed by the proof of a statement? Suppose for example that you have solved a very hard mathematical problem – say, one of the millenium problems whose solutions are rewarded a million dollar by the Clay Mathematics Institute. You can easily prove to a friend that you have solved it: simply show him your solution. However, revealing the actual solution gives him way more information than the fact that you know a solution – in particular, it gives him the opportunity to claim the prize himself. Could you avoid this pitfall, showing him that you solved this problem without revealing your solution? This boils down to the following general question:

Is it possible to produce a proof of the truth of a statement, that yields nothing but the

validity of this statement?

Goldwasser, Micali, and Rackoff's positive answer to this question in their seminal work [GMR89] has laid the foundations of cryptographic primitives known as *zero-knowledge proofs*, which have attracted considerable attention in the past decades, and enjoyed a tremendous number of applications. Zero-knowledge proofs, and their many natural variants, are the main focus of this thesis.

1.2.1 Zero-Knowledge Proofs

A zero-knowledge proof system is an interactive protocol between a prover and a verifier, which aims at demonstrating that some statement is true. Typical statements are membership statements to an NP-language: the prover should demonstrate that a public word x belongs to a given language $\mathscr{L}$ from the class NP. Informally, the proof must satisfy three properties:

1. correctness: if the statement is true, and the prover knows a proof of this, he will succeed in convincing the verifier;

2. soundness: if the statement is false, no prover can convince the verifier of the truth of the statement;

3. zero-knowledge: the interaction yields nothing beyond the fact that the statement is true. This is captured by requiring the existence of a *simulator* that can produce an honest-looking transcript for the protocol, without knowing anything about the statement.

Round-Efficient Zero-Knowledge Proofs. Zero-knowledge proof systems are usually interactive: they involve exchanging several messages between the prover and the verifier. In some scenarios, interactions are undesirable. In these situations, the standard approach is to use a specific type of zero-knowledge proofs, called non-interactive zero-knowledge proofs, which consist of a single flow from the prover to the verifier (after some one-time trusted setup has been performed). Generic methods have been designed to convert classical zero-knowledge proofs into non-interactive one, but they either provide only heuristic security guarantees [BR93] (relying on an ideal abstraction called the random oracle model), or lead to proofs which cannot be publicly verified, which limits their applicability [DFN06]. Provably secure publicly verifiable non-interactive zero-knowledge proofs have been introduced in [GS08], under a specific type of assumptions known as pairing-based assumptions; it is currently a major open problem whether proof systems with comparable efficiency could be constructed without such assumptions. In this thesis, we will describe an alternative type of round-efficient zero-knowledge proofs, which do not require such assumptions (but which are *not* non-interactive zero-knowledge proofs).

Zero-Knowledge Proofs over the Integers. Classically, efficient zero-knowledge proof systems deal with algebraic structures, such as finite groups. This makes existing proof systems well-suited for proving algebraic relations (for example, proving that some encrypted values satisfy some polynomial relation). On the other hand, several statements of interest are not efficiently captured by algebraic relations – the most standard example is that of range proofs, where the prover would like to show that some hidden value belongs to a given range.

Zero-knowledge proofs over the integers [Lip03] allow to efficiently prove algebraic relations between values, while treating these values as integers, instead of elements of finite groups. Intuitively, this is done by letting the parties perform a zero-knowledge proof over a finite group whose order is unknown to the prover: if the algebraic relation holds over this group, then unless the prover knows the order of the group, it must also hold over the integers. Dealing with groups of unknown order makes such proofs typically harder to analysis (for examples, the security reduction cannot rely on computing inversions, which require knowing the order of the group).

1.3 Secure Two-Party Computation

Secure two-party and multiparty computation (respectively abbreviated 2PC and MPC) has been introduced in the seminal works of Yao [Yao86], and Goldwasser, Micali, and Wigderson [GMW87b; GMW87a]. While cryptography had been traditionally concerned with securing *communication*, MPC asks whether it is possible to make *computation* secure. Slightly more formally, the problem of secure multiparty computation can be stated as follows: consider n parties $(P_1, \ldots, P_n)$, each holding respective inputs $(x_1, \ldots, x_n)$, knowing the description of a function $f : (\{0,1\}^*)^n \to \{0,1\}^*$. The MPC problem asks whether it is possible for all parties to obtain $f(x_1, \ldots, x_n)$, with the following security guarantee: all informations that can be deduced by any subset $S \subset [n]$ of the parties from the transcript of the protocol can already be deduced from $(x_i)_{i \in S}$ and $f(x_1, \ldots, x_n)$ (in other words, the transcript of the protocol can be efficiently *simulated* given only $(x_i)_{i \in S}$ and $f(x_1, \ldots, x_n)$).

The potential applications of MPC are manifold. It can be seen as providing a solution to the apparent opposition between privacy and usability: it allows any company (or any individual) owning data to ensure their privacy while at the same time letting them combine their data with other people's data, and evaluating any function of their choice to extract useful informations from it, getting potential benefits comparable to what they would get if every party was making his data publicly available. Therefore, secure computation has been a very active research field. While initial solutions were regarded as being mainly of theoretical interest, three decades of new ideas and clever optimizations have taken secure computation from being an intriguing theoretical object to becoming a practical and implementable tool, susceptible to provide an elegant solution to numerous problems. In fact, the first real-world uses of secure computation have already emerged [BLW08; BCD+09; BTW12; BKK+15], and the field is growing rapidly.

1.3.1 Active Security and Passive Security

Secure computation aims at protecting the privacy of data even when a subset of the parties has been corrupted by an adversary. Different security models emerge from the type of corruption that is considered. *Passive corruption* refer to adversaries that will only get access to the view of the corrupted parties. This corresponds to a situation where the computation has been performed by honest parties, and the transcript of the computation (which has been recorded and stored by the parties) is later leaked through an adversarial breakthrough (say, a hack of a party's computer). Security with respect to passive corruption (also known as security against honest-but-curious adversaries, or semi-honest security) tells that this transcript should not reveal any information about the inputs (apart from what can already be deduced given the output).

Security against *active corruption*, on the other hand, is a much stronger model. In this situation, the adversary is given full control on the parties it corrupts, and can arbitrarily modify their behavior. This corresponds to a situation where some of the parties are actively cheating during the protocol, in an attempt to gain information, or to alter the outcome of the protocol. This model is also known as security against malicious adversaries, or malicious security. It is obviously the most desirable security that can be achieved, and is also harder to realize than the semi-honest model.

The possibility of securely computing any two-party functionality in the semi-honest model has been first observed by Yao [Yao86]. Goldwasser, Micali, and Wigderson later extended this result to an arbitrary number of parties [GMW87b], still in the semi-honest setting. One year later, in [GMW87a], the same authors gave the first solution to general multiparty computation in the malicious model. Their solution gave an elegant blueprint that has been followed by most subsequent works on multiparty computation: start from a protocol secure against semi-honest adversaries, who follow the specifications of the protocol, and ask every party to *prove*, after sending each flow, that this flow was indeed computed honestly. As a flow can depend on private inputs held by the parties, this proof must carefully avoid to leak any private information; this is ensured by relying on zero-knowledge proofs, that are guaranteed to leak nothing beyond the validity of the statement.

The possibility of compiling any semi-honest protocol into a malicious protocol using zero-knowledge proofs is one of their most compelling applications to cryptography. In this thesis, we will be specifically interested in zero-knowledge proofs as a tool to ensure honest behavior in secure computation protocols.

1.4 Our Results

In this thesis, we introduce new types of zero-knowledge proofs, and revisit the security analysis of existing zero-knowledge proofs. Our results have implications for various types of secure computation protocols that rely on discrete-logarithm-based assumptions, or factorization-based assumptions. We expand below on the three contributions that are developed in this thesis, and outline some of their implications. The results discussed below have been mainly taken from two papers, [BCPW15] (co-authored with Fabrice Benhamouda, David Pointcheval, and Hoeteck Wee) and [CPP17] (co-authored with Thomas Peters and David Pointcheval), which have been presented respectively at CRYPTO 2015 and EUROCRYPT 2017.

1.4.1 Implicit Zero-Knowledge Arguments

In Chapter 4, we introduce a new type of zero-knowledge proofs, called implicit zero-knowledge arguments (iZK). While standard zero-knowledge arguments aim at producing a convincing proof that a statement is true, implicit zero-knowledge arguments are an *encapsulation mechanism*, that allows to mask a message, so that the message can be recovered if and only if the statement is true. iZK retains the same zero-knowledge properties as standard zero-knowledge arguments, in that the ability to unmask a message only leaks whether the statement was true, and nothing more.

Motivation. The main motivation for implicit zero-knowledge arguments is secure computation – more specifically, for compiling semi-honest two-party protocols into protocols

secure against malicious adversaries. It is indeed fairly easy to see that iZK can play a role comparable to standard zero-knowledge in secure computation: to guarantee the privacy of the inputs of the parties, it is not necessary to explicitly check that the opponent behaved honestly. Rather, it suffices to make sure that if it is not the case, it will be impossible for the other party to recover any subsequent messages of the protocol.

Comparison with Standard Zero-Knowledge. As explained above, iZK plays a role comparable to standard zero-knowledge in secure two-party computation. We now explain their benefits over classical approaches. An issue with the standard approach for compiling semi-honest protocols into malicious ones is the round-efficiency of the compiled protocol. Standard zero-knowledge proofs are interactive, hence their use for each flow of the protocol results in a blowup for the number of rounds. For secure computation in a WAN setting, this can be a major efficiency drawback.

To avoid the issue of blowing up the number of rounds in such settings, the traditional solution is to rely on *non-interactive* zero-knowledge proofs, in which the proof consists of a single flow from the prover to the receiver. However, unless one is willing to assume the random oracle model [BR93] (this is an idealized model that allows to give heuristic arguments of security, which cannot be turned into real proofs of security as this model cannot be instantiated in the real world, and leads to insecure constructions for some contrived examples [CGH98]), the only known (publicly verifiable) non-interactive zero-knowledge proofs are based on elliptic curves equipped with bilinear maps (also called pairings) [GS08].[1]

This raises two issues. The first one is that it forces to base the security of the compiled protocol on a narrower range of assumptions: while classical zero-knowledge proofs can be based on a wide variety of well-studied assumption, pairing-based non-interactive zero-knowledge proofs rely on more specific assumptions about elliptic curves with bilinear maps. The second one is that it comes at a computational cost by requiring to work over structure equipped with a pairing, as group operations on elliptic curves with pairings are typically slower than on the best elliptic curves without pairings. Furthermore, computing the proofs requires evaluating pairing operations, which are particularly slow.

Implicit zero-knowledge arguments aim at providing almost the same benefits as non-interactive zero-knowledge proofs regarding the round-efficiency of compiled protocols, under a wider range of assumptions. More precisely, while an n-round protocol is compiled to a $3n$-round protocol with standard zero-knowledge in general, and to an n-round protocol with non-interactive zero-knowledge, iZKs allow to compile it to an $(n + 2)$-round protocol in general, while being constructible from a wide variety of assumptions. They can be based on essentially the same assumptions as classical zero-knowledge protocols, and do not require pairings. In particular, this implies that they require less computation and communication that existing non-interactive zero-knowledge proofs, hence can act as a good alternative to them for round-efficient compiling of secure computation protocols. Our constructions are based on a primitive called hash proof system, or smooth projective hash function [CS02].

Applications. The most obvious application is the description of a new round-efficient general compiler from semi-honest to malicious security for two-party computation protocols. To illustrate the fact that iZK can often be applied more efficiently than with the general compiler for structured problems, we describe two more specific applications:

[1] In fact, publicly-verifiable non-interactive zero-knowledge proofs are also known from factorization-based assumptions, but they are rather inefficient and mainly of theoretical interest [BFM88; BDMP91].

- We show how iZK can be used to prove correctness of a computation represented by a given computational structure at a cost proportional to the representation size of the computation. We illustrate this on computations represented by boolean circuits, which are very generic, but also on computations represented by arithmetic branching programs, which often allow for more compact representation of arithmetic computations.

- We describe an optimized conversion for a specific two-party protocol (which, in particular, takes one round less than what the generic compiler would give, and is exactly as round-efficient as solutions based on non-interactive zero-knowledge) which computes the inner-product between private inputs of the parties. This illustrates the fact that iZK can be used more efficiently in specific settings. It also outlines some additional advantages of iZK: their structure make them easily amenable to batch techniques, which can be used to reduce communication and computation.

1.4.2 Improved Security Analysis of Integer Zero-Knowledge Arguments

Numerous secure computation protocols involve manipulating integers, seen as elements of $\mathbb{Z}$ and not as elements of a finite group. Examples include some electronic voting schemes, or some electronic cash systems. More generally, this situation is common for protocols manipulating values that should be in a specific range.

Standard zero-knowledge proofs can of course be used to convert such protocols into protocols secure against malicious adversaries, but this will in general involve treating the integer values as a bit-string and interpreting the statement as some polynomial relation between the bits of the strings, which can be quite inefficient in many scenarios. Zero-knowledge proofs over the integers are specific types of zero-knowledge proofs which aim at overcoming this issue, by providing tools to prove statements directly on integers, seen as atomic objects. These proofs often result in more efficient protocols, by avoiding blowups proportional to the bit-length of the integer values manipulated in the protocol.

This comes, however, at a cost regarding the assumptions on which such constructions can be based: all known zero-knowledge proofs over the integers require to assume a strong variant of the classical RSA assumption. While the RSA assumption is a very well studied assumption (related to the hardness of factoring integers), this variant gives a lot of freedom to the adversary, has been less studied, and can be argued to be less desirable.

In Chapter 5, we revisit the security analysis of the standard construction of zero-knowledge proofs over the integers. Our improved analysis shows that this construction can in fact be proven secure directly under the standard RSA assumption, without any modification. It was previously unknown whether even a modified version of this construction could be proven secure from the standard RSA assumption, and this question was a quite old open problem. Furthermore, our analysis involves interesting new ideas that could potentially be of independent interest and allow to create new types of constructions whose security can be reduced to the RSA assumption.

1.4.3 New Integer Zero-Knowledge Arguments

Eventually, in Chapter 6, we introduce a new method to construct zero-knowledge arguments over the integers. Our new method allows to reduce the work of the verifier in such proofs, as well as to reduce communication in many settings, still under the RSA assumption. This

makes it suited for use in secure computation protocols in a client-server setting, where the bulk of the computation should be performed by the (computationally more powerful) server, but the client should still be able to verify that the server behaved honestly. The method relies on the observation that existing integer commitment schemes (which bind the prover to some integer values while keeping those values hidden to the verifier) can be converted into another type of commitment scheme where the committed values are reduced modulo a small prime number, making these values less costly to manipulate and to communicate. Letting the prover perform this conversion during the proof allows the verifier to save a significant amount of computation for the verification of the proof.

1.5 Our Other Contributions

In addition to the contributions outlined above and developed in this manuscript, we have worked during our thesis on various other problems related to secure computation. These contributions are not described in this thesis because, even though they also target problems related to secure computation, they do not achieve their results through the analysis and design of zero-knowledge proofs, which we wanted to be the main focus of this thesis.

1.5.1 Encryption Switching Protocols [CPP16; CPP15b]

In this work, we introduced and studied a new cryptographic primitive, called encryption switching protocol (ESP). An ESP allows two players to convert an encryption of a message m with some cryptographic scheme into an encryption of the *same* message m, under a *different* cryptographic scheme. The crucial security requirement of an ESP is that this conversion should not reveal anything about the message m. This primitive can be used to efficiently reconcile the malleability properties of different encryption scheme, to be able to benefit from the properties of both schemes in secure computation protocols. More specifically, we build an ESP to swich between two encryption schemes, one which allows homomorphic evaluation of arbitrary linear functions, and one which allows homomorphic evaluation of arbitrary monomials (i.e., products and exponentiations). Together, these malleability properties allow to evaluate any function; for functions that can be represented efficiently by (sparse) multivariate polynomials, this allows to save communication, as the communication can be made independent of the degree of the function. In [CPP16], we introduce the primitive, instantiate it under standard assumptions, study its security, develop new types of zero-knowledge proofs to make it secure against malicious adversaries, and describe some applications to secure computation. In a workshop paper [CPP15b], we also describe an additional application of ESPs to the setting of delegation of computation.

These results appear in the proceedings of CRYPTO 2016, and of the workshop FPS 2015 (co-authored with Thomas Peters and David Pointcheval).

1.5.2 Homomorphic Secret Sharing [BCG+17]

In this work, we study a primitive called homomorphic secret sharing, which was introduced by Boyle *et al.* in [BGI16], and studied further in [BGI17]. This primitive allows to share an input between two parties, with the following guarantees: each share hides the input, yet each party can locally perform an evaluation procedure for some function f (from a class of functions specified by the scheme), so that for a share input x, the outputs of the players form

shares of $f(x)$. The core contribution of the work of Boyle *et al.* is a homomorphic secret sharing scheme for the class of all functions that can be computed by a branching program, under the decisional Diffie-Hellman assumption. This implies in particular the existence of secure computation protocols with communication sublinear in the circuit size of the function, still under the decisional Diffie-Hellman assumption; previously, such protocols were only known from lattice-based assumptions. In [BCG+17], we make four types of contributions: we optimize the scheme on all aspects (communication, computation, security analysis, usability, range of applications) using a combination of standard techniques and new methods, we describe applications for which our improved schemes provide practically efficient solutions that outperform alternative methods, we give a detailed report on implementation results (with non-trivial machine-level optimizations), and we introduce and study a new primitive that can be built from homomorphic secret sharing. The latter, called cryptocapsule, allows to greatly reduce the communication overload of the preprocessing phase of secure protocols. This primitive is still mainly of theoretical interest, but we describe new algorithmic techniques to make strong asymptotic improvements over a natural constructions of cryptocapsules from a method of [BGI17].

These results appear in the proceedings of CCS 2017 (co-authored with Elette Boyle, Niv Gilboa, Yuval Ishai, and Michele Orrù).

1.5.3 Secure Equality Tests and Comparisons [Cou16a]

In this work, we study a specific type of two-party secure computation protocol, where the goal of the two players is to learn which of their private inputs is greater. This protocol is commonly used as a subroutine in many protocols for secure computation, (examples include, but are not limited to, protocols for secure machine learning, or protocols for face recognition). We design a new protocol for comparing private inputs, in the semi-honest model, that relies only on a primitive known as oblivious transfer. Our protocol has an extremely efficient, information-theoretic online phase, and improves regarding communication and computation over the best existing solutions, at the cost of a higher number of rounds.

Toward constructing this protocol, we also introduce another protocol, which has independent applications: a protocol for securely testing whether two private strings are equal, so that the parties obtain bit-shares of the result of the test. This protocol also enjoys an extremely efficient, information-theoretic online phase, and a small communication and computation.

These results are described in a yet unpublished manuscript.

1.5.4 Covert Multiparty Computation [Cou16b]

In this work, we study a very strong form of secure computation, which was introduced in [AHL05; CGOS07]. A covert multiparty computation protocol allows parties to securely evaluate a function while hiding not only their input, but also the very fact that they are taking part to the protocol. Only when the final result is obtained are the players made aware of the fact that a computation indeed took place (and this happens only if the result was deemed favorable by all the parties).

While previous work had established feasibility results for covert MPC, these results were essentially of theoretical interest. In this work, we show that this very strong notion can be achieved at a surprisingly small cost, by modifying a state-of-the-art MPC protocol to

make it covert (at an additive cost independent of the size of the circuit), under standard assumptions. We develop a framework to argue the security of covert protocols that enhance them with non-trivial composability properties. As for our iZKs of Chapter 4, this protocol is built out of smooth projective hash functions.

These results are described in a yet unpublished manuscript.

1.6 Organization of this Thesis

The rest of this manuscript is organized as follows: in Chapter 2, we introduce the necessary preliminaries for this thesis, by recalling mathematical, algorithmic and computational notions, and by describing standard computational assumptions and cryptographic primitives. We devote an entire chapter to zero-knowledge proofs in Chapter 3, as they are the main subject of this thesis. This chapter aims at being somewhat hybrid between a survey and a preliminary chapter for our work: we formally introduce zero-knowledge proofs and some of their many variants, but also recall classical methods used for their design and their analysis, as well as historical results on their study. We try to provide a (non-comprehensive) overview of this subject to the reader, while at the same time introducing the necessary background for our contributions. Chapters 4, 5, 6 focus on our personal contributions, which we outlined above.

We attempted to gather all technical preliminaries that are not our contributions in the chapters 2 and 3, to clearly separate existing results that we use from our new contributions. Chapters 4, 5, 6, which focus only on our contributions, do refer the reader to the appropriate section of the preliminaries when necessary. However, it should be fairly easy for the reader to identify the sections of the preliminaries that are necessary to understand the details of each contribution.

Personal Publications

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. "Homomorphic Secret Sharing: Optimizations and Applications". In: *ACM CCS 17*. ACM Press, 2017, pp. 1–2 (cit. on pp. 9, 10).

[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. 6, 73).

[Cou16a] Geoffroy Couteau. *Efficient Secure Comparison Protocols*. Cryptology ePrint Archive, Report 2016/544. http://eprint.iacr.org/2016/544. 2016 (cit. on p. 10).

[Cou16b] Geoffroy Couteau. *Revisiting Covert Multiparty Computation*. Cryptology ePrint Archive, Report 2016/951. http://eprint.iacr.org/2016/951. 2016 (cit. on p. 10).

[CPP15b] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Secure Distributed Computation on Private Inputs". In: *International Symposium on Foundations and Practice of Security*. Springer. 2015, pp. 14–26 (cit. on p. 9).

[CPP16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Encryption Switching Protocols". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 308–338. DOI: 10.1007/978-3-662-53018-4_12 (cit. on p. 9).

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Heidelberg, May 2017, pp. 321–350 (cit. on p. 6).

2

Preliminaries

In this chapter, we introduce the notations and the basic notions that will be used throughout this thesis. We recall standard mathematical and algorithmic concepts, and introduce the main notions related to provable security. Afterward, we recall and discuss standard computational assumptions (discrete-logarithm-based assumptions and factorization-based assumptions) and cryptographic primitives that will be involved in this work. Most of the material presented in this section is rather standard and can be easily skimmed through.

Contents

2.1 Notation and Preliminaries 14
2.1.1 Mathematical Notations 14
2.1.2 Algorithms 15
2.1.3 Provable Security 16
2.2 Computational Assumptions 17
2.2.1 Discrete-Logarithm-Based Assumptions 18
2.2.2 Factorization-Based Assumptions 22
2.3 Cryptographic Primitives 26
2.3.1 One-Way Functions 26
2.3.2 Commitment Schemes 26
2.3.3 Public-Key Encryption Schemes 29
2.3.4 Smooth Projective Hash Functions 30

2.1 Notation and Preliminaries

2.1.1 Mathematical Notations

Sets, Integers. We denote by $\mathbb{Z}$ the set of integers, and by $\mathbb{N}$ the set of non-negative integers. The integer range $[\![a\,;b]\!]$ stands for $\{x \in \mathbb{Z} \mid a \leq x \leq b\}$, and $[\![a\,;b]\!]_c$ stands for $\{x \in \mathbb{Z} \mid a \leq x \leq b \ \wedge \ \gcd(x,c) = 1\}$. For a positive integer $k \in \mathbb{N}$, we denote by $\{0,1\}^k$ the set of bitstrings of length k. We denote by $\{0,1\}^*$ the set of all bitstrings. When an element s is represented by an integer, $|s|$ is the bit-length of the integer, and $||s||$ denotes its absolute value (or norm).

Modular Arithmetic. For an integer $x \in \mathbb{Z}$, the reduction of x modulo k, denoted $x \bmod k$, is the remainder of the Euclidean division of x by k. We denote $(\mathbb{Z}_k, +)$ the additive group of integers modulo k, i.e., the set $\{0, \ldots, k-1\}$ of non-negative integers smaller than k, equipped with the addition operation modulo k. We write $a = b \bmod k$ to specify that $a = b$ in $\mathbb{Z}_k$ and we write $a \leftarrow [b \bmod k]$ to affect the smallest positive integer to a so that $a = b \bmod k$. We denote by $(\mathbb{Z}_k, +, \cdot)$ the ring of integers modulo k. Furthermore, we denote by $(\mathbb{Z}_k^*, \cdot)$ the multiplicative subgroup of $(\mathbb{Z}_k, +, \cdot)$ of invertible integers modulo k. We will often abuse these notations and write $\mathbb{Z}_k$ for $(\mathbb{Z}_k, +)$, and $\mathbb{Z}_k^*$ for $(\mathbb{Z}_k^*, \cdot)$. Note that when k is a prime, which will often be the case in this thesis, $\mathbb{Z}_k$ is also a field, and $\mathbb{Z}_k^* = \mathbb{Z}_k \setminus \{0\}$. For arbitrary integers, the size of $\mathbb{Z}_k^*$ (the number of invertible elements modulo k) is given by Euler's totient function, which we denote $\varphi(k)$. It corresponds to the number of integers n between 1 and k such that $\gcd(k,n) = 1$, where gcd denotes the greatest common divisor.

Cyclic Groups. A cyclic group is a finite group generated by a single element. In particular, a cyclic group is Abelian (commutative). A generator g of a cyclic group $\mathbb{G}$ of order p is an element of $\mathbb{G}$ that generates the entire group, i.e., $\mathbb{G} = \{1, g, g^2, \ldots, g^{p-1}\}$ ($1 = g^0$ denotes the identity element). We denote this $\mathbb{G} = \langle g \rangle$. Given a group g, we denote by $\mathsf{ord}(\mathbb{G})$ its order.

Legendre Symbol and Jacobi Symbol. For an odd prime p, the Legendre symbol of a determines whether a is a quadratic residue modulo p. More specifically, it is 1 if a is a non-zero quadratic residue modulo p, -1 if it is a quadratic non-residue, and 0 if $a = 0 \bmod p$. It can be computed as $a^{(p-1)/2} \bmod p$. The Jacobi symbol generalizes the Legendre symbol with respect to every odd number: the Jacobi symbol of a modulo n is the product of its Legendre symbol with respect to every prime factor of n.

Vectors and Matrices. Vectors are denoted with an arrow. For a vector $\vec{x} = (x_1, \cdots, x_\ell)$, $g^{\vec{x}}$ denotes $(g^{x_1}, \cdots, g^{x_\ell})$. Matrices are denoted with capital greek letters (e.g., Γ).

Assignation. Given a finite set S, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable x: for any $s \in S$ we have $\Pr_S[x = s] = 1/|S|$ where $|S|$ denotes the cardinality of S.

Probabilities. We denote by $\Pr[X = x]$ the probability of a random variable X taking value x, and $\Pr_{x \in \mathcal{D}}[f(x) = y]$ to denote the probability that $f(x)$ is equal to some fixed value y, when x is sampled from the distribution $\mathcal{D}$. We denote by U_n the uniform distribution over $\{0,1\}^n$.

Miscellaneous. For a function f, we write '$f(x) = \mathrm{poly}(x)$ for all x' to indicate that there exists a polynomial p such that for every x in the domain of f, $||f(x)|| \leq p(x)$.

2.1.2 Algorithms

Turing Machines. A Turing machine is an abstract model of computations in which a tape head reads symbols on the tape and perform operations (writing, moving left, moving right) that depend on the symbol that is read. More formally, a Turing machine is a 7-tuple $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Γ is an alphabet (a set of symbols that can be on the machine tape), $b \in \Gamma$ is the blank symbol (to indicate that a cell of the tape is empty), $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of symbols initially written on the tape, δ is the transition function (it takes as input a state and a symbol, and outputs a new state, a new symbol, and a move indication, either left or right), $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. An accepting initial state corresponds to a state for which the machine M eventually halts in a state from F. This mathematical model provides a convenient abstraction to describe the computations that an algorithm can perform.

Interactive Probabilistic Turing Machines. All algorithms discussed in this work will be probabilistic Turing machines. A probabilistic Turing machine is a multi-tape Turing machines that can use an additional tape containing random bits (usually called random coins). When discussing interactive protocols, that involve interactive parties, the parties will be modeled as interactive probabilistic Turing machines, i.e., multi-tape Turing machines equipped with a read-only input tape, a read-only random tape, a read-and-write work tape, a write-only output tape, and a pair of communication tapes, one read-only and one write-only. Interaction between Turing machine is captured by letting pairs of machines share their communication tape: the read-only communication tape of one machine is the write-only communication tape of another machine, and vice versa.

Polynomial-Time Algorithms. We will call a *PPT algorithm*, or equivalently an *efficient algorithm*, a probabilistic algorithm running in time polynomial in its input size, on all inputs and all random coins. An *expected PPT algorithm* is a probabilistic algorithm whose expected running time is bounded by a polynomial in his input size, where the expectation is taken over the random coins of the algorithm.

Algorithm Execution. We write $y \leftarrow A(x)$ for 'y is the output of the algorithm A on the input x', while $y \xleftarrow{\$} A(x)$ means that A will additionally use random coins. We sometimes write st the state of the adversary.

Classes P, BPP, and NP. Decisions problems ar problems whose solutions are of the form 'yes' or 'no'. A decision problem defines a language, which is the set of all instances for which the answer to the decision problem is 'yes'. Solving an instance x of a decision problem d can be therefore formulated as finding out whether the *word* x belongs to the language $\mathscr{L}_d$ associated to d. We denote by P the class of languages that can be decided by a Turing machines running in time polynomial in its input size. BPP corresponds of the class of languages that can be decided by a polynomial-time probabilistic Turing machine with less than $1/3$ errors (both for positive answers and negative answers). Eventually, the class NP contains all languages $\mathscr{L}_R$ of the form $\mathscr{L}_R = \{x \mid \exists w, (|w| = \text{poly}(|x|)) \wedge (R(x, w) = 1)\}$, where R is a polynomial-time computable *relation*.

2.1.3 Provable Security

Negligibility. We say that a function μ is *negligible*, and write $\mu(x) = \text{negl}(x)$, if for any constant $c \in \mathbb{N}$ there exists $x \in \mathbb{N}$ such that for all $y \geq x$, $||\mu(y)|| \leq 1/y^c$. We say that a function μ is *overwhelming* if $1 - \mu$ is negligible.

Security Parameter. As is common in cryptography, the security properties of most primitives discussed in this thesis break down if the attacker has a sufficiently powerful computer, or is allowed a sufficiently long running time. Moreover, the computational power of an adversary can greatly evolve over time. This is captured by introducing a *security parameter*, which will be denoted κ throughout this thesis, and feeding all PPT algorithms with the unary representation 1^κ of the security parameter (this will sometimes be done implicitely). That way, all efficient algorithms are guaranteed to run in time polynomial in κ. The parameters of the system will be chosen so that the system is estimated to provide κ bits of security – i.e., such that the best known attack on the system requires 2^κ steps to be mounted. A common widely accepted value of the security parameter is 128: if 2^{128} computational steps are necessary to break a system, attacking the system can be considered infeasible within a reasonable amount of time, with the current computing power of computers.

Adversaries, which will be denoted with calligraphic letters (e.g., $\mathscr{A}$) will be usually modeled as efficient algorithms taking 1^κ as input. We will sometimes also consider security against *unbounded* adversaries, which can run in arbitrary time.

Oracle Access. In addition to inputs and random coins, algorithms will sometimes be given access to *oracles*. An oracle is an ideal black-box that receives some inputs and returns some output, and are used to capture the fact that an algorithm might get access to the answers to some queries, without specifying how these queries are asked, or how these answers are computed. Given an oracle O, we write $\mathscr{A}^{\mathsf{O}}(x)$ to indicate that the algorithm $\mathscr{A}$ run on input x is given oracle access to O.

Success, Advantage, Experiments. We define, for a distinguisher A and two distributions $\mathcal{D}_0, \mathcal{D}_1$, the advantage of A (i.e., its ability to distinguish those distributions) by $\mathsf{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(A) = \Pr_{x\in\mathcal{D}_0}[A(x) = 1] - \Pr_{x\in\mathcal{D}_1}[A(x) = 1]$. The qualities of adversaries will also be measured by their successes and advantages in certain experiments. An experiment is a game played between an adversary and a challenger; Figure 2.1 illustrates the way we represent an experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}$ for a property $\mathtt{sec}$ with an adversary $\mathscr{A}$. Successes refer to experiment where the adversary attempts to win with non-negligible probability: the experiment is denoted $\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}$, and the success is defined as $\mathsf{Succ}^{\mathsf{sec}}(\mathscr{A}, \kappa) = \Pr[\mathsf{Exp}^{\text{sec}}_{\mathscr{A}}(1^\kappa) = 1]$. Advantages refer to experiment where the adversary attempts to win with non-negligible advantage over the random guess: the experiment is denoted $\mathsf{Exp}^{\mathsf{sec}-b}_{\mathscr{A}}$, where a bit b distinguishes between two variants of the experiment that the adversary should distinguish between, and the advantage is defined as $\mathsf{Adv}^{\mathsf{sec}}(\mathscr{A}, \kappa) = \Pr[\mathsf{Exp}^{\text{sec}\,-1}_{\mathscr{A}}(1^\kappa) = 1] - \Pr[\mathsf{Exp}^{\text{sec}\,-0}_{\mathscr{A}}(1^\kappa) = 1]$. For both types of qualities, probabilities are over the random coins of the challenger and of the adversary.

Statistical Indistinguishability. The statistical distance between two distributions $\mathcal{D}_0$

$\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}(1^\kappa)$:
challenger interacts with adversary $\mathscr{A}$, by generating some material and running $\mathscr{A}$ on this material.
If the experiment proceeds in several rounds, $\mathscr{A}$ may pass a state st from one round to the next.
An experiment typically ends by checking for a condition:
if condition **then return** 1
else return 0

Figure 2.1: Template for representing an experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}(1^\kappa)$ with a challenger and an adversary $\mathscr{A}$

and $\mathcal{D}_1$ over some finite set S is given by

$$\sum_{i\in S}\left|\Pr_{x\xleftarrow{\$}\mathcal{D}_0}[x=i]-\Pr_{x\xleftarrow{\$}\mathcal{D}_1}[x=i]\right|.$$

For any integers $a \leq b$, the statistical distance between two uniform distributions, over $U_a = [\![1\,;a]\!]$ and $U_b = [\![1\,;b]\!]$ respectively, is given by $\sum_{i=1}^{b} |\Pr_{U_a}[x=i] - \Pr_{U_b}[x=i]| = \sum_{i=1}^{a}(1/a - 1/b) + \sum_{i=a+1}^{b} 1/b = 2(b-a)/b$. Two distributions are said statistically indistinguishable if their statistical distance is negligible. Note that this is equivalent to saying that any algorithm (not necessarily polynomial time) has negligible advantage over distinguishing the two distributions.

2.2 Computational Assumptions

In this section, we recall the classical computational assumptions on which we will rely throughout this work. As most cryptographic assumptions (and unlike standard assumptions in complexity theory, which are worst-case hardness assumptions), they are concerned with the average-case hardness of certain mathematical problems. The assumptions we will discuss can be divided in two main categories, discrete-logarithm-based assumptions and factorization-based assumptions, which are sometimes referred to as the "20$^{\text{th}}$ century assumptions". Indeed, they were for a long time the assumptions underlying most constructions of public-key cryptography (with some noticeable exceptions [McE78]), starting with the famous RSA [RSA78a] and ElGamal [ElG85] cryptosystems. Even though the last decade has witnessed the emergence of new types of cryptographic assumptions (the most prominent being lattice-based assumptions [Ajt96; Reg05; HPS98; LPR10]), they remain widely used to date and as such, a large body of work has been dedicated to their study; we will recall the main cryptanalytic results and cryptographic reductions when introducing the assumptions. A general issue with 20$^{\text{th}}$ century assumptions, which was one of the main motivations for the study of alternative assumptions, is that they are only conjectured to hold against classical PPT adversaries: it was shown in the seminal work of Shor [Sho99] that they do not hold against quantum polynomial-time adversaries, hence the advent of general-purpose quantum computers would render insecure the constructions based on these assumptions. However,

$\mathsf{Exp}^{\mathsf{dlog}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$:
$x \xleftarrow{\$} \mathbb{Z}_p$
$X \leftarrow g^x$
$x' \leftarrow \mathscr{A}(X)$
if $x' = x$ **then return** 1
else return 0

Figure 2.2: Experiment $\mathsf{Exp}^{\mathsf{dlog}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the discrete logarithm problem over a group $\mathbb{G}$ of order p with generator g

their security against classical computers is quite well understood, and they enjoy a number of algebraic properties which make them well suited for a wide number of applications and amenable to practical instantiations.

2.2.1 Discrete-Logarithm-Based Assumptions

Given a cyclic group $\mathbb{G}$ with a generator g, the discrete logarithm assumption over $\mathbb{G}$ states, informally, that it is computationally infeasible given a random group element $h \in \mathbb{G}$ to find an integer x such that $h = g^x$. Generic algorithms, which are independent of the particular structure of the underlying group $\mathbb{G}$, have a running time proportional to the square root of the group order. In spite of more than four decades of intense cryptanalytic effort, there exist certain groups in which we do currently not know of algorithm with better efficiency than the generic algorithms. For all assumptions discussed in this section, no attack significantly better than solving the discrete logarithm assumption is known, although in most cases, no formal reduction showing that a PPT algorithm breaking the assumption would imply a PPT algorithm breaking the discrete logarithm assumption are known. As previously mentioned, the discrete logarithm assumption (hence all assumptions discussed in this section) does not hold against quantum polynomial-time adversaries [Sho99].

2.2.1.1 The Discrete Logarithm Assumption

Let $\mathbb{G}$ be a cyclic group, with a generator g. The discrete logarithm assumption states that:

Assumption 2.2.1. *(Discrete Logarithm Assumption) For any efficient algorithm $\mathscr{A}$, it holds that* $\mathsf{Succ}^{\mathsf{dlog}}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$.

The experiment $\mathsf{Exp}^{\mathsf{dlog}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ is represented Figure 2.2.

Below, we briefly discuss standard property of the dlog assumption, generic attacks against the assumption, and standard groups over which the assumption is commonly instantiated.

Random Self-Reducibility. An important property of the discrete logarithm assumption is its *random self-reducibility*, a property introduced in [AFK87]: if the discrete logarithm problem is hard for some specific instances over a group $\mathbb{G}$, then it remains hard for random instanced over $\mathbb{G}$. Indeed, suppose that an oracle solves random instances of dlog over $\mathbb{G}$, and let $h = g^x$ be some fixed dlog instance. Then it is easy to see that $h' \leftarrow hg^r$ for a uniformly random r follows a uniformly random distribution in $\mathbb{G}$, and from the discrete logarithm $y \in \mathbb{Z}_{\mathsf{ord}(\mathbb{G})}$ of h' returned by the oracle, one can compute the logarithm x of h as $x = y - r \bmod \mathsf{ord}(\mathbb{G})$.

Generic Attacks. Let $t \leftarrow \mathsf{ord}(\mathbb{G})$. A generic deterministic algorithm for computing discrete logarithms in arbitrary groups was designed by Shanks [Sha71]; it involves $O(\sqrt{t})$ group operations (and comparable space complexity). This algorithm was improved by Pollard in [Pol78] to use only constant space (at the cost of being probabilistic rather than deterministic), and still $O(\sqrt{t})$ group operations. It is also more amenable to distributed computation than Shank's baby-step-giant-step algorithm. Pollard's rho algorithm, for which practical improvements were suggested in [Tes01; BLS11], is the state-of-the-art algorithm for computing arbitrary discrete logarithm in generic groups. However, more efficient algorithms can exist for specific group, which might have additional structure. Shoup's proof of optimality of Pollard's algorithm in a model known as the generic group model [Sho97] suggests that it might be asymptotically optimal in arbitrary groups. This has led to two research directions: designing improved algorithms for specific groups commonly used in crypto, or designing specific groups in which no attack better than Pollard's rho algorithm is known. Both directions are closely interleaved, as new cryptanalytic insights from the former influence the design strategies of the latter.

Instantiating $\mathbb{G}$ with Multiplicative Subgroups of Finite Fields. One of the most common instantiations of $\mathbb{G}$ is as follows: let p be a random large *strong prime* (which means that $p = 2p' + 1$, where p' is itself a prime). Then $\mathbb{F}_p$ is a field, and the subgroup of squares of $\mathbb{F}_p^*$ (i.e., group elements of the form u^2 for $u \in \mathbb{F}_p^*$) is a cyclic group of order p' where the discrete logarithm assumption is conjectured to hold. Note that to construct a generator of $\mathbb{G}$, it suffices to pick any $u \in \mathbb{F}_p^* \setminus \{1\}$ and set $g \leftarrow u^2$. Note that the assumption is also conjectured to hold directly over $\mathbb{F}_p^*$; however, considering the subgroup of squares is common because some assumptions related to dlog, such as the decisional Diffie-Hellman assumption (which we will discuss in this section), are insecure over $\mathbb{F}_p^*$. Over such groups, the best known attacks are known as *index calculus methods*, whose fundational ideas were developed in [Kd22]. It gives rise to algorithms with subexponential complexity ($O(\exp(a \log^b t \log\log^{1-b} t))$, for some constants a and $b \leq 1/3$). The latest development in this family of attacks was provided by Joux in [Jou14] for fields of small characteristic. The existence of subexponential algorithms implies that parameters must be chosen quite large: current recommandations suggest using 2048-bit primes. A survey of state-of-the-art index calculus-based methods for discrete logarithms in finite fields is given in [JOP14].

Instantiating $\mathbb{G}$ with Elliptic Curves. Alternatively, it is common to instantiate the discrete logarithm assumptions over elliptic curves, which are algebraic curves defined by an equation of the form $y^2 = x^3 + ax + b$ on which a group operation can be defined. Elliptic curves have been the subject of a very rich study. In spite of an important cryptanalytic effort that showed how to use various methods (such as index calculus-based methods and bilinear maps-based methods) to speed up discrete logarithm computation over elliptic curves, Pollard's rho algorithm is still the only known attack over some well-chosen curves, which nonetheless admit efficient algorithms for computing group operations (e.g., [Mil86; Kob87; Ber06]). For such curves, choosing the order of a group as a 256-bit prime suffices to give 128 bits of security with respect to all known attacks. Group elements are therefore up to ten times smaller over elliptic curves than over finite fields for comparable security requirements.

Other Common Instantiations of $\mathbb{G}$. In addition to finite fields and elliptic curves, the discrete logarithm problems has been considered in various other groups in the literature. One of them is the subgroup of squares of $\mathbb{Z}_n^*$, where n is a product of two primes. It is

$\mathsf{Exp}^{\mathsf{cdh}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$:
$(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$
$(X, Y) \leftarrow (g^x, g^y)$
$Z \leftarrow \mathscr{A}(X, Y)$
if $Z = X^y$ **then return** 1
else return 0

Figure 2.3: Experiment $\mathsf{Exp}^{\mathsf{cdh}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the computational Diffie-Hellman problem over a group $\mathbb{G}$ with generator g

worth noting that the discrete logarithm in this group reduces to the discrete logarithm over the subgroup of squares of both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ when the factorization $n = pq$ of the modulus is known, using CRT decomposition; when the factorization is unknown, the hardness of the dlog assumption over this group is implied by the hardness of the factorization assumption (which will be discussed later on in this section).

2.2.1.2 The Computational Diffie-Hellman Assumption

The computational Diffie-Hellman assumption was introduced by Diffie and Hellman in their seminal work on public-key cryptography [DH76], and has been used as a basis for a tremendous number of cryptographic applications. As above, let $\mathbb{G}$ be a cyclic group, with a generator g. The computational Diffie-Hellman assumption states that:

Assumption 2.2.2. *(Computational Diffie-Hellman Assumption (CDH)) For any efficient algorithm $\mathscr{A}$, it holds that* $\mathsf{Succ}^{cdh}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$.

The experiment $\mathsf{Exp}^{\mathsf{cdh}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ is represented Figure 2.3. It is easy to see that CDH is implied by the dlog assumption. In the reverse direction, no attack significantly better than solving a discrete logarithm problem is known against CDH. However, no formal reduction is known in general. It was proven by Boer [den90] that CDH is as hard as the discrete logarithm problem over $\mathbb{F}_p^*$ if $\varphi(p-1)$ is *smooth* (i.e., its prime factors are small). A general algorithm for solving dlog given access to a CDH oracle in arbitrary group is due to Maurer [Mau94], but it assumes (informally) that some additional information is known about the order of the group. We also note that as for the DLOG assumption, the CDH assumption satisfies random self-reducibility.

2.2.1.3 The Decisional Diffie-Hellman Assumption

While the discrete logarithm assumption and the computational Diffie-Hellman assumptions are based on search problem (they assume that finding a solution to some problem is infeasible in polynomial time), the decisional variant of the Diffie-Hellman assumption is based on a decision problem.

Assumption 2.2.3. *(Decisional Diffie-Hellman Assumption (DDH)) For any efficient algorithm $\mathscr{A}$, it holds that* $\mathsf{Adv}^{\mathsf{ddh}}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$.

The experiments $\mathsf{Exp}^{\mathsf{ddh}-b}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$, indexed by a bit b, are represented Figure 3.2. The DDH assumption states that any efficient algorithm $\mathscr{A}$ has negligible advantage in distinguishing $\mathsf{Exp}^{\mathsf{ddh}-0}_{\mathscr{A}}$ from $\mathsf{Exp}^{\mathsf{ddh}-1}_{\mathscr{A}}$; as for dlog and CDH, it is random self-reducible. Like the

$\mathsf{Exp}^{\mathsf{ddh}-0}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$:
$(x, y, z) \xleftarrow{\$} \mathbb{Z}_p^3$
$(X, Y, Z) \leftarrow (g^x, g^y, g^z)$
return $\mathscr{A}(X, Y, Z)$

$\mathsf{Exp}^{\mathsf{ddh}-1}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$:
$(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$
$(X, Y, Z) \leftarrow (g^x, g^y, g^{xy})$
return $\mathscr{A}(X, Y, Z)$

Figure 2.4: Experiments $\mathsf{Exp}^{\mathsf{ddh}-0}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ and $\mathsf{Exp}^{\mathsf{ddh}-1}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the decisional Diffie-Hellman problem over a group $\mathbb{G}$ with generator g

$\mathsf{Exp}^{\mathsf{dlin}-0}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$:
$(x, y, z) \xleftarrow{\$} \mathbb{Z}_p^3$
$F \xleftarrow{\$} \mathbb{G}$
$(X, Y, Z) \leftarrow (g^x, h^y, F^z)$
return $\mathscr{A}(X, Y, F, Z)$

$\mathsf{Exp}^{\mathsf{dlin}-1}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$:
$(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$
$F \xleftarrow{\$} \mathbb{G}$
$(X, Y, Z) \leftarrow (g^x, h^y, F^{x+y})$
return $\mathscr{A}(X, Y, F, Z)$

Figure 2.5: Experiments $\mathsf{Exp}^{\mathsf{dlin}-0}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$ and $\mathsf{Exp}^{\mathsf{dlin}-1}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$ for the decision linear assumption over a group $\mathbb{G}$ with generators (g, h)

CDH assumption, no attack significantly better than solving a discrete logarithm problem is known against DDH. It is easy to solve DDH given oracle access to a CDH solver; however, DDH is not believed to be equivalent to dlog in general, or even to CDH, as there exists groups in which CDH is believed to hold, while DDH does not hold (examples include $\mathbb{F}_p^*$, where computing the Legendre symbol gives non-negligible advantage in distinguishing DDH tuples from random tuples, or elliptic curves equipped with a symmetric pairing).

2.2.1.4 Generalizations of the DDH Assumption

In this section, we consider several variants of the DDH assumption that generalize it: the decision linear assumptions, and the matrix Diffie-Hellman assumptions. In our work, we will describe several DDH-based primitives. All of our constructions that can be reduced to the DDH assumption are described with respect to this assumption for the sake of concreteness; they can be generalized in a natural way to hold under the k-Lin assumption for any k, or under any of the MDDH assumptions.

The Decision Linear Assumption. The decision linear assumption (DLIN) is a variant of the DDH assumption introduced in [BBS04] as a way to construct analogeous of DDH-based cryptographic primitives in groups where the DDH assumption does not hold.

Assumption 2.2.4. *(Decision Linear Assumption (DLIN)) For any efficient algorithm $\mathscr{A}$, it holds that* $\mathsf{Adv}^{dlin}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$.

The experiments $\mathsf{Exp}^{\mathsf{dlin}-b}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$, indexed by a bit b, are represented Figure 2.5. The properties of the DLIN assumption are comparable to those of the DDH assumption. The DLIN assumption can be generalized in a natural way to the k-Lin assumption, where the goal is to distinguish $(F, F^{\sum_{i \leq k} x_i})$ from (F, F^z). This leads to a hierarchy of increasingly weaker assumptions.

The Matrix Diffie-Hellman Assumption. The MDDH family of assumptions further generalizes the DDH assumption and the k-Lin assumptions, allowing for more variants of

DDH under which most constructions can be naturally extended to work. It was introduced in [EHK+13]. Relations between the assumptions of this family were recently studied in [Vil17]. As we will not explicitly use it in our work (although our DDH-based constructions can be extended to work under MDDH assumptions), we skip the details.

2.2.1.5 Assumptions on Bilinear Groups

A bilinear map (or pairing) is an application $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of the same order, that satisfies $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all exponents a, b and generators g_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$. Elliptic curves equipped with bilinear maps are commonly used in cryptography, either in the *symmetric pairing* setting ($e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$) or in the *asymmetric pairing* setting ($\mathbb{G}_1 \neq \mathbb{G}_2$). Note that the DDH assumption cannot hold in a group with a symmetric pairing, although it can hold in one of the groups (or both) of a pair of groups equipped with an asymmetric pairing. Curves with bilinear maps have been observed to allow for the construction of advanced cryptographic primitives [Jou00; BF01]. Cryptographic constructions relying on such curves are usually proven under variants of the Diffie-Hellman assumptions over pairing groups. We do not introduce specific instances of such assumptions here, as we will not use them in this work. We note, however, that they are believed to be stronger than the above assumptions (DDH, CDH, and their variants), and can be only instantiated over specific groups, while the previously discussed assumptions can be stated for a large variety of cyclic groups.

2.2.2 Factorization-Based Assumptions

The factorization assumption states, informally, that given a product n with large prime factors, it is computationally infeasible to factor n. With the discrete logarithm assumption, the factorization assumption is one of the most common assumptions in public-key cryptography.

While the assumptions can in theory be instantiated with any product of sufficiently large prime numbers, we will focus in this work on the most standard version, where the modulus is computed as the product of two large primes (p, q). In addition, it is common to restrict the primes (p, q) to be *strong primes*. That means that $p = 2p' + 1$ and $q = 2q' + 1$ for two other primes so that p, p', q, q' are all distinct, and $\varphi(n) = 4p'q'$. One can note that in such groups, p and q are Blum primes: $p = q = 3 \bmod 4$. Assuming that (p, q) have been chosen this way, we can consider the following subgroups of $\mathbb{Z}_n^*$:

- We let $\mathbb{J}_n$ denote the subgroup of $\mathbb{Z}_n^*$ with Jacobi symbol 1. It has order $\varphi(n)/2 = 2p'q'$. It is the largest cyclic group contained in $\mathbb{Z}_n^*$.
- We let QR_n denote the subgroup of the squares, i.e., $\mathsf{QR}_n = \{a \in \mathbb{Z}_n^* \mid \exists b \in \mathbb{Z}_n^*, a = b^2 \bmod n\}$. This is a cyclic subgroup of $\mathbb{Z}_n^*$ of order $\varphi(n)/4 = p'q'$, and a subgroup of $\mathbb{J}_n$.

Properties of QR_n. Below, we outline some classical properties of the group QR_n which will be useful for our work.

Proposition 2.2.5. *The following facts hold:*

1. $-1 \in \mathbb{J}_n \setminus \mathsf{QR}_n$;

$\mathsf{Exp}^{\mathsf{fact}}_{\mathscr{A}}(1^\kappa)$:
$(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$
$(a,b) \leftarrow \mathscr{A}(n)$
if $(p,q) = (a,b)$ **then return** 1
else return 0

Figure 2.6: Experiment $\mathsf{Exp}^{\mathsf{fact}}_{\mathscr{A}}(1^\kappa)$ for the factorization assumption

2. *any square $a \in \mathsf{QR}_n$ has four square roots, with exactly one in QR_n.*

Let us briefly explain why these facts hold, using the Jacobi symbol function $J_n(x) = J_p(x) \times J_q(x)$ in $\mathbb{Z}_n^*$, as the extension of the Legendre symbol on $\mathbb{Z}_p^*$ for prime p, where $J_p(x) = (x)^{(p-1)/2}$ determines whether x is a square or not in $\mathbb{Z}_p^*$. Since p and q are Blum primes, $J_p(-1) = J_q(-1) = -1$, and so $J_n(-1) = 1$, but -1 is not in QR_n, hence the Fact 1. The four square roots of 1, in $\mathbb{Z}_n^*$ are 1 and -1, both with Jacobi symbol $+1$, but respectively $(+1,+1)$ and $(-1,-1)$ for the Legendre symbols in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$, and α, and $-\alpha$, both with Jacobi symbol -1, but respectively $(+1,-1)$ and $(-1,+1)$ for the Legendre symbols in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. As a consequence, given a square $h \in \mathsf{QR}_n$, and a square root u, the four square roots are $u, -u$, and $\alpha u, -\alpha u$, one of which being in QR_n, since the four kinds of Legendre symbols are represented. This leads to the Fact 2.

Modulus Generation Algorithm. In the following, we denote by $\mathsf{GenMod}(1^\kappa)$ a PPT algorithm that, given the security parameter κ, generates a strong RSA modulus n and secret parameters (p,q) of at least κ bits each with the specification that $n = pq$. In the following, we write $(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$. We will sometimes abuse the notation $n \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$ to say that the modulus n has been generated according to this distribution.

2.2.2.1 The Factorization Assumption

We state the factorization assumption below. The experiment $\mathsf{Exp}^{\mathsf{fact}}_{\mathscr{A}}$ is represented Figure 2.6.

Assumption 2.2.6. *(Factorization Assumption) For any efficient algorithm $\mathscr{A}$, it holds that $\mathsf{Succ}^{\mathsf{fact}}(\mathscr{A},\kappa) = \mathrm{negl}(\kappa)$.*

The best known attacks against the factorization assumption are based on an algorithm known as the general number field sieve, and run in subexponential time, with the current record being the factorization of a 768-bit modulus [KAF+10]. Current recommandations for the size of p and q suggest to use 2048-bit random strong primes. As for assumptions related to the discrete logarithm, Shor's algorithm [Sho99] can be used to break this assumption in polynomial time with a quantum computer.

Additional Properties. We outline some useful observations regarding the factorization assumption.

Proposition 2.2.7. *The following facts hold:*

1. *for a random element $h \in \mathsf{QR}_n$, finding a square root of h is equivalent to factoring the modulus n;*
2. *for random elements $g, h \in \mathsf{QR}_n$, finding non-zero integers a, b such that $g^a = h^b \bmod n$ is equivalent to factoring the modulus n;*

$\mathsf{Exp}^{\mathsf{rsa}}_{\mathscr{A}}(1^\kappa)$:
$(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$
$e \xleftarrow{\$} \mathtt{Dist}_n$
$x \xleftarrow{\$} \mathbb{Z}_n$
$y \leftarrow \mathscr{A}(n,x,e)$
if $y^e = x \bmod n$ and $e \neq 1$ **then return** 1
else return 0

Figure 2.7: Experiment $\mathsf{Exp}^{\mathsf{rsa}}_{\mathscr{A}}(1^\kappa, \mathtt{Dist}_n)$ for the RSA family of assumptions, parametrized by a distribution $\mathtt{Dist}_n$ over P_n

Proof. For Fact 1, if one chooses a random $u \in \mathbb{Z}_n^*$ and sets $h = u^2 \bmod n$, $J_n(u)$ is completely hidden. Another square root v has probability one-half to have $J_n(v) = -J_n(u)$. This means that $u^2 = v^2 \bmod n$, but $u \neq \pm v \bmod n$. Then, $\gcd(u-v,n)$ gives a non-trivial factor of n.

For Fact 2, if one chooses a random $u \in \mathbb{Z}_n^*$ and a large random scalar α and sets $h = u^2 \bmod n$ and $g = h^\alpha \bmod n$, h is likely a generator of QR_n, and then $g^a = h^b \bmod n$ means that $m = b - a\alpha$ is a multiple of $p'q'$, the order of the subgroup of the squares. Let us write $m = 2^v \cdot t$, for an odd t, then $p'q'$ divides t: let us choose a random element $u \in \mathbb{Z}_n^*$, with probability close to one-half, $J_n(u) = -1$, and so $J_n(u^t) = -1$ while u^t is a square root of 1. As in the proof of the previous Fact 1, we can obtain a non-trivial factor of n. □

We also note that by Fact 1, the hardness of the factorization assumption implies the hardness of the CDH assumption over the group QR_n. Indeed, let g be a target element of QR_n. Given access to a CDH oracle over QR_n, one can compute a square root of g as follows: pick $(a,b) \xleftarrow{\$} \mathbb{Z}^2_{n/4}$ (observe that a, b follow a distribution statistically close from the uniform distribution over $\mathbb{Z}_{\varphi(n)/4}$), set $h \leftarrow g^2$, compute $(g_0, g_1) \leftarrow (g \cdot h^a, g \cdot h^b)$, and send (h, g_0, g_1) to the CDH oracle. It is easy to observe that, given the CDH output $g_2 = h^{(a+1/2)\cdot(a+1/2)} = h^{ab} g^{a+b+1/2}$, one can recover a square root of g by computing $g_2 \cdot h^{ab} \cdot g^{-a-b}$.

2.2.2.2 The RSA Family of Assumptions

The RSA assumption is probably one of the most famous assumptions in cryptography. It was introduced in the seminal work of Rivest, Shamir, and Adleman [RSA78b]. It states, informally, that given an exponent e prime to $\varphi(n)$, it is hard for any probabilistic polynomial-time algorithm to find the e-th root modulo n of a random $y \xleftarrow{\$} \mathbb{Z}_n^*$. More formally, let P_n be the subset of $[\![1\,;n]\!]$ of elements prime to $\varphi(n)$. The RSA assumption does in fact refer to a class of assumptions, depending of the distribution $\mathtt{Dist}_n$ over P_n from which the exponent e is drawn. The experiment $\mathsf{Exp}^{\mathsf{rsa}}_{\mathscr{A}}$ is represented Figure 2.7.

Assumption 2.2.8. *($\mathtt{Dist}_n$-RSA Family of Assumptions) For any efficient algorithm $\mathscr{A}$, it holds that* $\mathsf{Succ}^{\mathsf{rsa}}(\mathscr{A}, \kappa, \mathtt{Dist}_n) = \mathrm{negl}(\kappa)$.

Various flavours of the RSA assumption are standard in the literature. In particular, the RSA assumption with a fixed small exponent (the most common being 65537) is widely used in practical implementations. In theoretical papers, it is common to consider the

$\mathsf{Exp}^{\mathsf{rsa}}_{\mathscr{A}}(1^\kappa)$:
$(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$
$x \xleftarrow{\$} \mathbb{Z}_n$
$(y,e) \leftarrow \mathscr{A}(n,x)$
if $y^e = x \bmod n$ **then return** 1
else return 0

Figure 2.8: Experiment $\mathsf{Exp}^{\mathsf{srsa}}_{\mathscr{A}}(1^\kappa)$ for the Strong-RSA assumption

RSA assumption for exponents picked from the uniform distribution over P_n (see [HW09] for example). In this work, we will use several flavours of the RSA assumption which are somewhat intermediate between these two standard variants: we will consider the RSA assumption for exponents picked from the uniform distribution over $[\![3\,;a]\!] \cap \mathsf{P}_n$ for a value a polynomial in κ (hence, we consider random small exponents), which we will denote a-RSA, and the RSA assumption for random κ-bit prime exponents.

Relation to the Factorization. It is clear that the hardness of RSA implies the hardness of the factorization, but the converse is not known. In fact, there are some evidences that we are unlikely to find a black-box reduction from factoring to RSA [BV98], although such a reduction is known if the adversary is restricted to be a straight-line program [Bro16].

Additional Properties. We outline some useful observations regarding the RSA family of assumptions.

Proposition 2.2.9. *For an RSA instance (n,e,y), finding $x \in \mathbb{Z}_n^*$ and e' prime to e such that $x^e = y^{e'} \bmod n$ is equivalent to finding an e-th root of y modulus n.*

Proof. Using Bézout relation $ue + ve' = 1$, then $x^{ve} = y^{ve'} = y^{1-ue} \bmod n$. So $y = (x^v y^u)^e \bmod n$. □

2.2.2.3 The Strong-RSA Assumption

The Strong-RSA assumption [BP97; FO97] is a variant of the RSA assumption that gives more freedom to the adversary: rather than drawing the exponent e from some specified distribution, it lets the choice of e to the adversary. The experiment$\mathsf{Exp}^{\mathsf{srsa}}_{\mathscr{A}}$ is represented Figure 2.8.

Assumption 2.2.10. *(Strong-RSA Assumption) For any efficient algorithm $\mathscr{A}$, it holds that $\mathsf{Succ}^{srsa}(\mathscr{A},\kappa) = \mathrm{negl}(\kappa)$.*

Relation to RSA and Factoring. It is clear that if Strong-RSA is hard, then both factoring and RSA are hard (for any distribution over the exponent). No reduction is known in the other direction. In fact, Strong-RSA appears clearly stronger than the RSA family of assumptions: while in any RSA assumption, the adversary $\mathscr{A}$ wins if he finds the only valid solution to the challenge, there are *exponentially many* valid solutions to a Strong-RSA challenge. Using the terminology of [GK16], both factoring and RSA are 1-search complexity assumption, while Strong-RSA is a t-search complexity assumption, for an exponentially large t. Although it remains a *falsifiable* assumption (i.e., the falseness of the assumption can be proven by exhibiting an algorithm that breaks it), t-search assumptions with large t are less desirable cryptographic assumptions. This is best said by quoting Goldwasser and Kalai [GK16]:

'Whereas the strong RSA assumption is considered quite reasonable in our community, the existence of exponentially many witnesses allows for assumptions that are overly tailored to cryptographic primitives.'

Such assumptions do not allow standard win-win results: either the assumption is true and some cryptographic construction is secure, or it is not, and we obtain a useful algorithm for solving some hard problem. Therefore, it is typical to try to avoid such assumptions in cryptography.

2.3 Cryptographic Primitives

In this section, we discuss cryptographic primitives that are relevant to our work.

2.3.1 One-Way Functions

Cryptography is based on the existence of tasks that can be efficiently *executed*, but that cannot be efficiently *abused*. One-way functions represent the most fundamental object of this kind, and as such constitute the basis of a variety of other primitives: a one-way function is a function that can be efficiently computed, but that cannot be efficiently inverted (where inverting means finding any valid preimage of a random image). More formally,

Definition 2.3.1. *(One-Way Function) A function $f : \{0,1\}^* \to \{0,1\}^*$ is* one-way *if it satisfies the following two conditions:*

1. *there exists a deterministic polynomial-time algorithm F so that for all input x in the domain of f, $F(x) = f(x)$ (in other words, f is efficiently computable);*
2. *for every PPT algorithm $\mathscr{A}$, every (positive) polynomial $p(\cdot)$, and all large enough n's,*

$$\Pr[\mathscr{A}(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

Note that this definition enlightens the importance of the parameter 1^n: without it, a function could appear one-way merely because it shrinks its input by a very large (say, exponentially large) factor, hence inverting the output cannot be done in time polynomial in the *output size*. We also remark that this definition corresponds to a flavor of one-way function usually known as *strong one-way function*, and is the most standard one. A strong one-way function guarantees that any PPT adversary has only a negligible probability of inverting the function; a weak one-way function, on the other hand, only guarantees that any PPT adversary has a *non-negligible probability of failing* to invert the function. In fact, weak one-way functions can be shown to imply strong one-way function (and the reverse direction is obvious), hence both definitions are equivalent.

2.3.2 Commitment Schemes

The notion of commitment is one of the most fundamental and widely used in cryptography. A commitment scheme allows a committer $\mathcal{C}$ holding a secret value s to send a *commitment*

$\mathsf{Exp}^{\mathsf{hiding}-0}_{\mathscr{A}}(1^\kappa)$:
$\mathsf{pp} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\kappa)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pp})$
$r \xleftarrow{\$} \mathcal{R}$
$(c, d) \leftarrow \Pi.\mathsf{Commit}(\mathsf{pp}, m_0; r)$
$b \leftarrow \mathscr{A}(\mathsf{pp}, c, \mathsf{st})$

$\mathsf{Exp}^{\mathsf{hiding}-0}_{\mathscr{A}}(1^\kappa)$:
$\mathsf{pp} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\kappa)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pp})$
$r \xleftarrow{\$} \mathcal{R}$
$(c, d) \leftarrow \Pi.\mathsf{Commit}(\mathsf{pp}, m_1; r)$
$b \leftarrow \mathscr{A}(\mathsf{pp}, c, \mathsf{st})$

Figure 2.9: Experiments $\mathsf{Exp}^{\mathsf{hiding}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathsf{hiding}-1}_{\mathscr{A}}(1^\kappa)$ for the hiding property of a commitment scheme Π

c of s to a verifier $\mathcal{V}$, and later on to *open* this commitment to reveal the value s. Such a commitment should *hide* the committed value s to the verifier, but *binds* the committer in opening only s. More formally,

Definition 2.3.2. *(Commitment Scheme) A commitment scheme* Π *is a triple of PPT algorithms* $(\Pi.\mathsf{Setup}, \Pi.\mathsf{Commit}, \Pi.\mathsf{Verify})$*, such that*

- $\Pi.\mathsf{Setup}(1^\kappa)$*, generates the public parameters* pp*, which also specifies the message space* $\mathcal{M}$*, the commitment space* $\mathcal{C}$*, the opening space* $\mathcal{D}$*, and the random source* $\mathcal{R}$*;*
- $\Pi.\mathsf{Commit}(\mathsf{pp}, m; r)$*, given the message* $m \in \mathcal{M}$ *and some random coins* $r \in \mathcal{R}$*, outputs a commitment-opening pair* (c, d)*,*
- $\Pi.\mathsf{Verify}(\mathsf{pp}, c, d, m)$*, outputs a bit* b *whose value depends on the validity of the opening* (m, d) *with respect to the commitment* c*,*

which satisfies the correctness, hiding, and binding properties defined below.

We now introduce the security properties of a commitment scheme, starting with the correctness property.

Definition 2.3.3. *(Correctness of a Commitment Scheme) A commitment scheme* Π *is* correct *if for any public parameters* $\mathsf{pp} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\kappa)$*, any message* $m \in \mathcal{M}$*, and any random coin* $r \in \mathcal{R}$*, for* $(c, d) \leftarrow \Pi.\mathsf{Commit}(\mathsf{pp}, m; r)$*, it holds that* $\Pi.\mathsf{Verify}(\mathsf{pp}, c, d, m) = 1$*.*

We define the hiding property of a commitment scheme. The experiments $\mathsf{Exp}^{\mathsf{hiding}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathsf{hiding}-1}_{\mathscr{A}}(1^\kappa)$ for the hiding property of a commitment scheme Π are represented Figure 2.9.

Definition 2.3.4. *(Hiding Property of a Commitment Scheme) A commitment scheme* Π *is* hiding *if for any PPT adversary* $\mathscr{A}$*, it holds that* $\mathsf{Adv}^{\mathit{hiding}}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$*.*

We define the binding property of a commitment scheme. The experiment $\mathsf{Exp}^{\mathsf{binding}}_{\mathscr{A}}(1^\kappa)$ for the binding property of a commitment scheme Π is represented Figure 2.10.

Definition 2.3.5. *(Binding Property of a Commitment Scheme) A commitment scheme* Π *is* binding *if for any PPT adversary* $\mathscr{A}$*, it holds that* $\mathsf{Succ}^{\mathit{binding}}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$*.*

$\mathsf{Exp}^{\mathsf{binding}}_{\mathscr{A}}(1^\kappa)$:
$\mathsf{pp} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\kappa)$
$(c, d, m_0, m_1) \xleftarrow{\$} \mathscr{A}(\mathsf{pp})$
if $\Pi.\mathsf{Verify}(\mathsf{pp}, c, d, m_0) = 1$ and $\Pi.\mathsf{Verify}(\mathsf{pp}, c, d, m_1) = 1$ **then return** 1
else return 0

Figure 2.10: Experiment $\mathsf{Exp}^{\mathsf{binding}}_{\mathscr{A}}(1^\kappa)$ for the binding property of a commitment scheme Π

Homomorphic Commitment Scheme. A commitment scheme can also be *homomorphic*, if for a group law $\oplus$ on the message space $\mathcal{M}$, from $(c_0, d_0) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_0; r_0)$ and $(c_1, d_1) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_1; r_1)$, one can generate c from c_0 and c_1 (and pp) as well as d from d_0 and d_1 (and pp) so that $\mathsf{Verify}(\mathsf{pp}, c, d, m_0 \oplus m_1) = 1$.

An Example: the Pedersen Commitment Scheme. A famous example of commitment scheme is the Pedersen commitment scheme [Ped92], which is perfectly hiding and whose binding property relies on the discrete logarithm assumption:

Example 2.3.6. *(Pedersen) The Pedersen commitment scheme is defined as follows:*

- $\mathsf{Setup}(1^\kappa)$*, generates the description of a group* $\mathbb{G}$ *of prime order* p *together with two generators* (g, h)*. We let* pp *denote* $(\mathbb{G}, p, g, h)$*;*
- $\mathsf{Commit}(\mathsf{pp}, m; r)$*, given the message* $m \in \mathbb{Z}_p$ *and some random coins* $r \in \mathbb{Z}_p$*, outputs a commitment-opening pair* $(c, d) \leftarrow (g^m h^r, r)$*,*
- $\mathsf{Verify}(\mathsf{pp}, c, d, m)$*, returns* 1 *iff* $g^m h^r = c$*.*

This commitment scheme is perfectly hiding, binding under the discrete logarithm assumption in $\mathbb{G}$*, and additively homomorphic.*

Proof. For the hiding property, notice that upon random choice of $r \in \mathbb{Z}_p$, for any $m \in \mathbb{Z}_p$, $g^m h^r$ is uniformly distributed over $\mathbb{G}$. For the binding property, given openings (r_0, r_1) for a commitment c to distinct messages (m_0, m_1), the relation $g^{m_0} h^{r_0} = g^{m_1} h^{r_1}$ leads to $h = g^{(m_0 - m_1)(r_1 - r_0)^{-1}}$, which gives the discrete logarithm of h in base g. Finally, it is clear that from $c_0 = g^{m_0} h^{r_0}$ and $c_1 = g^{m_1} h^{r_1}$, $r_0 + r_1$ is a valid opening of $c_0 c_1$ to $m_0 + m_1$, hence the homomorphic property. □

On the Relation Between Commitments and One-Way Functions. It has been established early that one-way functions can be used to construct computationally hiding, statistically binding commitment schemes [HILL99; Nao91]. More recently, one-way functions have also been shown to imply statistically hiding, computationally binding commitment schemes [NOV06; HR07].

Proposition 2.3.7. *If one-way functions exist, there are both statistically binding, computationally hiding commitment schemes, and statistically hiding, computationally binding commitment schemes.*

$\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-0}_{\mathscr{A}}(1^\kappa)$:
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\kappa)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pk})$
$r \xleftarrow{\$} \mathcal{R}$
$c \leftarrow \Pi.\mathsf{Encrypt}(\mathsf{pk}, m_0; r)$
$b \leftarrow \mathscr{A}(\mathsf{pk}, c, \mathsf{st})$

$\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-1}_{\mathscr{A}}(1^\kappa)$:
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\kappa)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pk})$
$r \xleftarrow{\$} \mathcal{R}$
$c \leftarrow \Pi.\mathsf{Encrypt}(\mathsf{pk}, m_1; r)$
$b \leftarrow \mathscr{A}(\mathsf{pk}, c, \mathsf{st})$

Figure 2.11: Experiments $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-1}_{\mathscr{A}}(1^\kappa)$ for the IND-CPA security property of a public-key encryption scheme Π

2.3.3 Public-Key Encryption Schemes

A public-key encryption scheme allows to encode a message using an encryption key, so that it is unfeasible to find out the message from its encoding (even given the encryption key), yet a decryption key allows to recover the message from the encoding. More formally,

Definition 2.3.8. *(Public-Key Encryption Scheme) A public-key encryption scheme* Π *is a triple of PPT algorithms* $(\Pi.\mathsf{KeyGen}, \Pi.\mathsf{Encrypt}, \Pi.\mathsf{Decrypt})$*, such that*

- $\Pi.\mathsf{KeyGen}(1^\kappa)$*, generates a pair* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *is the public key and* sk *is the private key. We assume that* pk *specifies the ciphertext space* $\mathcal{C}$*, the message space* $\mathcal{M}$*, and the random source* $\mathcal{R}$*;*
- $\Pi.\mathsf{Encrypt}(\mathsf{pk}, m; r)$*, given the message* $m \in \mathcal{M}$ *and some random coins* $r \in \mathcal{R}$*, outputs a ciphertext* c*;*
- $\Pi.\mathsf{Decrypt}(\mathsf{sk}, c)$*, output a message* $m \in \mathcal{M}$*;*

which satisfies the correctness and IND-CPA security properties defined below.

Definition 2.3.9. *(Correctness of an Encryption Scheme) A public-key encryption scheme* Π *is* correct *if for any pair* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\kappa)$*, any message* $m \in \mathcal{M}$*, and any random coin* $r \in \mathcal{R}$*, decryption is the reverse operation of encryption:* $\Pi.\mathsf{Decrypt}(\mathsf{sk}, \Pi.\mathsf{Encrypt}(\mathsf{pk}, m; r)) = m$*.*

The experiments $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-1}_{\mathscr{A}}(1^\kappa)$ for the IND-CPA security property of a public-key encryption scheme Π are represented Figure 2.11.

Definition 2.3.10. *(IND-CPA Security Property of a Public-Key Encryption Scheme) A public-key encryption scheme* Π *is* IND-CPA secure *if for any PPT adversary* $\mathscr{A}$*, it holds that* $\mathsf{Adv}^{\mathtt{ind\text{-}cpa}}(\mathscr{A}, \kappa) = \mathrm{negl}(\kappa)$*.*

Public-key encryption schemes can also be additively homomorphic, which is defined similarly as for commitment schemes.

An Example: the ElGamal Encryption Scheme. The ElGamal encryption scheme is a famous public-key encryption scheme introducted in [ElG85] whose IND-CPA security property reduces to the DDH assumption. In its original formulation, the plaintext m is a group element. We describe below a standard variant, where the message is encrypted in the exponent. This variant is additively homomorphic and decryption requires brute-forcing over the message space; the latter must therefore be sufficiently small.

Example 2.3.11. *(ElGamal) The ElGamal encryption scheme is defined as follows:*

- $\mathsf{KeyGen}(1^\kappa)$, *generates the description of a group* $\mathbb{G}$ *of prime order* p *together with a generators* g, *picks* $s \xleftarrow{\$} \mathbb{Z}_p$, *sets* $h \leftarrow g^s$, $\mathsf{pk} \leftarrow (\mathbb{G}, p, g, h)$, *and* $\mathsf{sk} \leftarrow (\mathbb{G}, p, g, s)$.
- $\mathsf{Encrypt}(\mathsf{pk}, m; r)$, *given the message* $m \in \mathbb{Z}_p$ *(from a message space of size bounded by a polynomial) and some random coins* $r \in \mathbb{Z}_p$, *parses* pk *as* $(\mathbb{G}, p, g, h)$ *and outputs a ciphertext* $c \leftarrow (g^r, g^m h^r)$,
- $\mathsf{Decrypt}(\mathsf{sk}, c)$, *parses* c *as* (c_0, c_1), sk *as* $(\mathbb{G}, p, g, s)$, *and outputs* $m \leftarrow \mathsf{dlog}_g(c_1/c_0^s)$.

This encryption scheme is perfectly correct, additively homomorphic, and its IND-CPA security reduces to the decisional Diffie-Hellman assumption (see Section 2.2.1.3).

Proof. Correctness and additive homomorphism are clear from the description. For IND-CPA security, given a bit b and a tuple (g, h, u, v), the challenger in the experiment $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-b}_{\mathscr{A}}(1^\kappa)$ sets (g, h) to be the public key of the scheme. Upon receiving (m_0, m_1) from $\mathscr{A}$, he computes c as (u, vg^{m_b}). If (g, h, u, v) is a random tuple, this perfectly hides m_b; otherwise, if (g, h, u, v) is a DDH tuple, this is a valid encryption of m_b. Therefore, from the answer of an adversary that wins the experiment with non-negligible probability, the challenger can guess whether (g, h, u, v) is a DDH tuple with non-negligible probability. □

2.3.4 Smooth Projective Hash Functions

Hash proof systems, also called smooth projective hash functions (SPHFs), have been introduced by Cramer and Shoup in [CS02] as a tool to build IND-CCA secure cryptosystems (i.e., cryptosystems which should remain secure even when the adversary is given access to a decryption oracle). Since then, SPHFs have found many more applications.

We recall that an NP-language $\mathscr{L} \subset \mathfrak{X}$ associated to a *relation* R is a set of the form $\mathscr{L} = \{x \in \mathfrak{X} \mid \exists w, R(x, w) = 1\}$, where R is a polynomial-time computable function. Informally speaking, an SPHF for a language $\mathscr{L} \subset \mathfrak{X}$ provides two ways to hash a word $x \in \mathfrak{X}$, either using a hashing key hk, or using a projection key hp together with a witness w for the statement $x \in \mathscr{L}$. The correctness requirement states that the two ways of hashing should return the same hash value for any $x \in \mathscr{L}$. The smoothness property, on the other hand, says that if $x \notin \mathscr{L}$, then the hash value computed with hk is statistically indistinguishable from a random value from the view point of any adversary, even given hp. More formally,

Definition 2.3.12. *(SPHF) A smooth projective hash function* S *for a* NP*-language* $\mathscr{L} \subset \mathfrak{X}$ *with hash space* $\mathcal{H}$ *is a 4-tuple of PPT algoritms* $(S.\mathsf{HashKG}, S.\mathsf{ProjKG}, S.\mathsf{Hash}, S.\mathsf{ProjHash})$ *such that:*

- $S.\mathsf{HashKG}(1^\kappa)$: *outputs a hashing key* hk*;*
- $S.\mathsf{ProjKG}(\mathsf{hk}, x)$: *on input* hk *and a word* $x \in \mathfrak{X}$, *output a projection key* hp*;*
- $S.\mathsf{Hash}(\mathsf{hk}, x)$: *on input* hk *and a word* $x \in \mathfrak{X}$, *outputs a hash value* $H \in \mathcal{H}$*;*
- $S.\mathsf{ProjHash}(\mathsf{hp}, x, w)$: *on input* hp, *a word* $x \in \mathscr{L}$, *and a witness* w *for the statement* $x \in \mathscr{L}$, *outputs a projective hash value* $\mathsf{proj}H \in \mathcal{H}$*;*

which satisfies the correctness and smoothness properties defined below.

Smooth projective hash functions are often used in conjunction with *hard-subset-membership languages*, which are NP languages $\mathscr{L} \subset \mathfrak{X}$ satisfying two properties: it is possible to efficiently sample pairs (x, w) where x is uniform over $\mathscr{L}$ and w is a valid witness for $x \in \mathscr{L}$ (with a PPT algorithm Sample), and no efficient algorithm can distinguish the distribution $\{x \xleftarrow{\$} \mathfrak{X}\}$ from the distribution $\{x \in \mathfrak{X} \mid (x, w) \xleftarrow{\$} \mathsf{Sample}(\mathscr{L})\}$. As an example, the language of DDH tuples is a hard-subset-membership language over $\mathbb{G}^4$, provided that the decisional Diffie-Hellman assumption holds over $\mathbb{G}$.

Note that the above definition allows the projection key hp to depend on the word x. This is a relaxation of the original definition of [CS02], where hp had to be independent of the word x, that was suggested in [GL06; KOY09]. The variant considered here is usually called GL-SPHF (for Gennaro-Lindell SPHF). We now define the correctness property of SPHFs.

Definition 2.3.13. *(Correctness of SPHFs) A smooth projective hash function S for a* NP*-language $\mathscr{L} \subset \mathfrak{X}$ is correct if for any* $\mathsf{hk} \xleftarrow{\$} S.\mathsf{HashKG}(1^\kappa)$*, any $x \in \mathscr{L}$ with witness w, and* $\mathsf{hp} \leftarrow S.\mathsf{ProjKG}(\mathsf{hk}, x)$*, it holds that*

$$S.\mathsf{Hash}(\mathsf{hk}, x) = S.\mathsf{ProjHash}(\mathsf{hp}, x, w)$$

We now define the smoothness property of SPHFs.

Definition 2.3.14. *(Smoothness of SPHFs) A smooth projective hash function S for a* NP*-language $\mathscr{L} \subset \mathfrak{X}$ is smooth if for any* $\mathsf{hk} \xleftarrow{\$} S.\mathsf{HashKG}(1^\kappa)$*, any $x \in \mathfrak{X} \setminus \mathscr{L}$, and* $\mathsf{hp} \leftarrow S.\mathsf{ProjKG}(\mathsf{hk}, x)$*, the distributions* $\{(x, \mathsf{hp}, H) \mid H \leftarrow S.\mathsf{Hash}(\mathsf{hk}, x)\}$ *and* $\{(x, \mathsf{hp}, H) \mid H \xleftarrow{\$} \mathcal{H}\}$ *are statistically indistinguishable.*

Note that this definition assumes that the word x is chosen before hp is generated (as, in particular, the latter can depend of it). We already mentioned the alternative definition of [CS02], in which hp is independent of x. With this alternative definition, two variants can be considered: the strongest, denoted KV-SPHF (for Katz-Vaikuntanathan SPHF) requires the smoothness to hold even if the word x is chosen adaptively by the adversary after seing hp, while the weakest, usually denoted CS-SPHF (for Cramer-Shoup SPHF) is the original definition of [CS02] and requires x to be generated before seing hp.

2.3.4.1 Generic Framework of SPHFs over Cyclic Groups

In this section, we briefly introduce an algebraic framework for constructing SPHFs on languages defined over cyclic groups. This framework was initially introduced in [BBC+13].

Languages. Let $\mathbb{G}$ be a cyclic group of prime order p and $\mathbb{Z}_p$ the field of integers modulo p. If we look at $\mathbb{G}$ and $\mathbb{Z}_p$ as the same ring $(\mathbb{G}, +, \bullet)$, where internal operations are on the scalars, many interesting languages can be represented as subspaces of the vector space $\mathbb{G}^n$, for some n. Here are some examples.

Example 2.3.15 (DDH or ElGamal ciphertexts of 0)**.** *Let g and h be two generators of $\mathbb{G}$. The language of DDH tuples in basis (g, h) is*

$$\mathscr{L} = \{(u, e) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p,\ u = g^r \text{ and } e = h^r\} \subseteq \mathbb{G}^2,$$

where r is the witness. It can be seen as the subspace of $\mathbb{G}^2$ generated by (g, h). We remark that this language can also be seen as the language of (additive) ElGamal ciphertexts of 0 for the public key $\mathsf{pk} = (g, h)$. □

Example 2.3.16 (DLin)**.** *Let g_1, g_2 and h be three generators of $\mathbb{G}$. The language of DLin tuples in basis (g_1, g_2, h) is*

$$\mathscr{L} = \{(u_1, u_2, e) \in \mathbb{G}^2 \mid \exists (r, s) \in \mathbb{Z}_p^2,\ u_1 = g_1^r,\ u_2 = g_2^s \text{ and } e = h^{r+s}\} \subseteq \mathbb{G}^3,$$

where r is the witness. It can be seen as the subspace of $\mathbb{G}^3$ generated by the rows of the following matrix:

$$\Gamma = \begin{pmatrix} g_1 & 1 & h \\ 1 & g_2 & h \end{pmatrix}.$$

□

Example 2.3.17 (ElGamal ciphertexts of a bit)**.** *Let us consider the language of ElGamal ciphertexts of 0 or 1, under the public key $\mathsf{pk} = (g, h)$:*

$$\mathscr{L} := \{(u, e) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_p, \exists b \in \{0, 1\},\ u = g^r \text{ and } e = h^r g^b\}.$$

Here $C = (u, e)$ cannot directly be seen as an element of some vector space. However, a word $C = (u, e) \in \mathbb{G}^2$ is in $\mathscr{L}$ if and only there exists $\vec{\lambda} = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{Z}_p^3$ such that:

$$\begin{aligned} u &= g^{\lambda_1}\ (= \lambda_1 \bullet g) & e &= h^{\lambda_1} g^{\lambda_2}\ (= \lambda_1 \bullet h + \lambda_2 \bullet g) \\ 1 &= u^{\lambda_2} g^{\lambda_3}\ (= \lambda_2 \bullet u + \lambda_3 \bullet g) & 1 &= (e/g)^{\lambda_2} h^{\lambda_3}\ (= \lambda_2 \bullet (e - g) + \lambda_3 \bullet h), \end{aligned}$$

because, if we write $C = (u, e) = (g^r, h^r g^b)$ (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 (right bottom) 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, if we introduce the notation $\vec{C} = \theta(C) := \begin{pmatrix} u & e & 1 & 1 \end{pmatrix} \in \mathbb{G}^4$, then the language $\mathscr{L}$ can be defined as the set of $C = (u, e)$ such that $\vec{C}$ is in the subspace of $\mathbb{G}^4$ generated by the rows of the following matrix

$$\Gamma := \begin{pmatrix} g & h & 1 & 1 \\ 1 & g & u & e/g \\ 1 & 1 & g & h \end{pmatrix}.$$

□

Example 2.3.18 (Conjunction of Languages)**.** *Let g_i and h_i (for $i = 1, 2$) be four generators of $\mathbb{G}$, and $\mathscr{L}_i$ be (as in Example 2.3.15) the languages of DDH tuples in bases (g_i, h_i) respectively. We are now interested in the language $\mathscr{L} = \mathscr{L}_1 \times \mathscr{L}_2 \subseteq \mathbb{G}^4$, which is thus the conjunction of $\mathscr{L}_1 \times \mathbb{G}^2$ and $\mathbb{G}^2 \times \mathscr{L}_2$: it can be seen as the subspace of $\mathbb{G}^4$ generated by the rows of the following matrix*

$$\Gamma := \begin{pmatrix} g_1 & h_1 & 1 & 1 \\ 1 & 1 & g_2 & h_2 \end{pmatrix}.$$

□

This can also be seen as the matrix, diagonal by blocks, with Γ_1 and Γ_2 the matrices for $\mathscr{L}_1$ and $\mathscr{L}_2$ respectively.

More formally, the generic framework for SPHFs in [BBC+13] considers the languages $\mathscr{L} \subseteq \mathfrak{X}$ defined as follows: There exist two functions θ and Γ from the set of words $\mathfrak{X}$ to the vector space $\mathbb{G}^n$ of dimension n, and to set $\mathbb{G}^{k \times n}$ of $k \times n$ matrices over $\mathbb{G}$, such that $C \in \mathscr{L}$ if and only if $\vec{C} := \theta(C)$ is a linear combination of the rows of $\Gamma(C)$. From a witness

w for a word C, it should be possible to compute such a linear combination as a row vector $\overrightarrow{\lambda} = (\lambda_i)_{i=1,\ldots,k} \in \mathbb{Z}_p^{1\times k}$:

$$\overrightarrow{\hat{C}} = \theta(C) = \overrightarrow{\lambda} \bullet \Gamma(C). \tag{2.1}$$

For the sake of simplicity, because of the equivalence between w and $\overrightarrow{\lambda}$, we will use them indifferently for the witness.

SPHFs. Let us now build an SPHF on such a language. A hashing key hk is just a random column vector $\mathsf{hk} \in \mathbb{Z}_p^n$, and the associated projection key is $\mathsf{hp} := \Gamma(C) \bullet \mathsf{hk}$. The hash value of a word C is then $H := \overrightarrow{\hat{C}} \bullet \mathsf{hk}$, and if $\overrightarrow{\lambda}$ is a witness for $C \in \mathscr{L}$, this hash value can also be computed as:

$$H = \overrightarrow{\hat{C}} \bullet \mathsf{hk} = \overrightarrow{\lambda} \bullet \Gamma(C) \bullet \mathsf{hk} = \overrightarrow{\lambda} \bullet \mathsf{hp} = \mathsf{proj}H,$$

which only depends on the witness $\overrightarrow{\lambda}$ and the projection key hp. On the other hand, if $C \notin \mathscr{L}$, then $\overrightarrow{\hat{C}}$ is linearly independent from the rows of $\Gamma(C)$. Hence, $H := \overrightarrow{\hat{C}} \bullet \mathsf{hk}$ looks random even given $\mathsf{hp} := \Gamma(C) \bullet \mathsf{hk}$, which is exactly the *smoothness* property.

Example 2.3.19. *The SPHF corresponding to the language in Example 2.3.17, is then defined by:*

$$\begin{aligned}
&\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{hk}_3, \mathsf{hk}_4)^\intercal \xleftarrow{\$} \mathbb{Z}_p^4\\
&\mathsf{hp} = \Gamma(C) \bullet \mathsf{hk} = (g^{\mathsf{hk}_1} h^{\mathsf{hk}_2}, g^{\mathsf{hk}_2} u^{\mathsf{hk}_3} (e/g)^{\mathsf{hk}_4}, g^{\mathsf{hk}_3} h^{\mathsf{hk}_4})\\
&H = \overrightarrow{\hat{C}} \bullet \mathsf{hk} = u^{\mathsf{hk}_1} e^{\mathsf{hk}_2} \qquad\qquad \mathsf{proj}H = \overrightarrow{\lambda} \bullet \mathsf{hp} = \mathsf{hp}_1^r \cdot \mathsf{hp}_2^b \cdot \mathsf{hp}_3^{-rb}.
\end{aligned}$$

For the sake of clarity, we will omit the C argument, and write Γ, instead of $\Gamma(C)$.

3

Zero-Knowledge Proofs and Arguments

The main focus of our work is the study of interactive proofs, that allow a prover to convince a computationally bounded verifier of the truth of a statement. More specifically, we will consider *zero-knowledge arguments*, in which the prover is also computationally bounded, and the proof is guaranteed (informally) to leak nothing except the truth of the statement. In this chapter, we recall the classical definitions of zero-knowledge proofs and argument. To provide the reader with an overview of the context of our work, we organize this chapter as a (non-comprehensive) introduction to zero-knowledge, and recall a variety of important results, together with chronological insights and concrete examples.

Contents

3.1 Interactive Proofs . . . 37
3.1.1 Definitions . . . 37
3.1.2 Historical Notes . . . 38
3.2 Interactive Zero-Knowledge Proofs . . . 39
3.2.1 Definitions . . . 39
3.2.2 Brief Survey of Known Results . . . 42
3.3 Interactive Zero-Knowledge Arguments . . . 43
3.3.1 Definitions . . . 43
3.3.2 Historical Notes . . . 44
3.4 Proofs and Arguments of Knowledge . . . 44
3.4.1 Definitions . . . 45
3.5 Σ-Protocols . . . 46
3.5.1 Definition . . . 46
3.6 The Common Reference String Model . . . 50
3.6.1 Trusted Setup Assumptions . . . 51
3.6.2 Proving Security of Zero-Knowledge Proof Systems . . . 51
3.6.3 Simulation Soundness . . . 53
3.7 Zero-Knowledge Arguments over the Integers . . . 53
3.7.1 Zero-Knowledge Proofs of Non-Algebraic Statements . . . 53
3.7.2 Range Proofs . . . 54

3.7.3 Zero-Knowledge Arguments from Diophantine Relations 54
3.8 Non-Interactive Zero-Knowledge Arguments 55
3.8.1 Definition and Security Properties 55
3.8.2 Brief Survey of Known Results 56
3.8.3 Fiat-Shamir Heuristic 57
3.8.4 Groth-Sahai Proofs 57

3.1 Interactive Proofs

As discussed in the introduction, interactive proofs capture a natural generalization of the class NP; in this section, we discuss further this relation and formally define these notions.

3.1.1 Definitions

We recall that the class NP corresponds to the class of languages for which a computationally unbounded teacher can generate a proof of membership π which can be verified by a polynomial-time student – this captures proofs that can be *efficiently* verified. More formally,

Definition 3.1.1. *(The Class NP) A language $\mathscr{L}$ is in the class NP if there exists a polynomial time algorithm $R_{\mathscr{L}}$ such that*

$$\mathscr{L} = \{x \mid \exists \pi, |\pi| = \text{poly}(|x|) \wedge R_{\mathscr{L}}(x, \pi) = 1\}$$

The proof π is usually called a *witness* for the statement $x \in \mathscr{L}$. By the above definition, NP contains all languages for which an unbounded prover can compute deterministic proofs, where a proof is viewed as a string of size polynomial in the word x.

An interactive proof relaxes these requirements in two directions: first, the parties are allowed to use random coins, and the output of a proof verification should only match the actual truth of the statement with some reasonable enough probability. Second, rather than seing the proof as a fixed string π checked by the student, the student is allowed to interact with his teacher, asking questions and receiving answers in an adaptive way.

Definition 3.1.2. *(Interactive Proof System) An n-round interactive proof system $(\mathcal{P}, \mathcal{V})$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for a language $\mathscr{L}$ is any pair of randomized algorithms (modeled as interactive Turing Machines) such that $\mathcal{V}$ runs in probabilistic polynomial-time and the following conditions hold:*

- ***Completeness.*** *$(\mathcal{P}, \mathcal{V})$ is complete, if for any $x \in \mathscr{L}$:*

$$\Pr\left[\,(\mathcal{P}, \mathcal{V})(x) = 1\,\right] \geq 2/3;$$

- ***Soundness.*** *$(\mathcal{P}, \mathcal{V})$ is sound, if for any $x \notin \mathscr{L}$, for any prover $\mathcal{P}'$:*

$$\Pr\left[\,(\mathcal{P}', \mathcal{V})(x) = 1\,\right] \leq 1/3;$$

We denote by $(\mathcal{P}, \mathcal{V})(x)$ the random variables (where randomness is taken over the coin tosses of the parties) representing the output of $\mathcal{V}$ after interacting with $\mathcal{P}$ on a common input x.

The choice of the constants 1/3 and 2/3 in Definition 3.1.2 is arbitrary: the constant 1/3 (resp. 2/3) can be amplified by sequential repetitions to be as close to 0 (resp. to 1) as desired.[1] Typically, the error probability will be made superpolynomially small in the security parameter. It is clear that every language $\mathscr{L}$ in NP has an interactive proof system,

[1] When considering interactive proof systems, the soundness error can also be made small via *parallel* repetition of the protocol [Bab85; Gol98], preserving the round efficiency. However, this is not true for every variant of interactive proofs – such as interactive arguments and zero-knowledge proofs – while sequential repetitions do always allow to reduce the soundness error.

where the interaction consists in a single flow from $\mathcal{P}$ to $\mathcal{V}$: the prover simply sends the membership witness w to the verifier, who outputs one if and only if $R_{\mathscr{L}}(x, w) = 1$.

Having defined interactive proof systems, it is natural to consider the class IP of all languages that have an interactive proof system.

Definition 3.1.3. *(Class IP) A language $\mathscr{L}$ is in the class IP if it has an interactive proof system with a polynomial number of rounds (in its input length).*

3.1.2 Historical Notes

Seminal Works. Interactive proofs systems were studied in two seminal papers by Babai [Bab85], and Goldwasser, Micali, and Rackoff [GMR85]. Both papers introduced and studied complexity classes where a computationally unbounded prover must convince a polynomially bounded receiver of the truth of a statement using rounds of interactions. The main difference between the notions studied in both works is with respect to the random coins of the verifier: in the work of Babai, the verifier was required to reveal to the prover all coins that he used during the computation. Such interactive proofs are referred to as *public coin* interactive proofs, as opposed to *private coin* interactive proofs, in which the verifier might keep its internal state hidden. The complexity classes corresponding to public coin interactive proofs were denoted $\mathsf{AM}[f(n)]$ by Babai, where AM stands for Arthur-Merlin, n is the input length, and $f(n)$ is the allowed number of rounds of interaction. The complexity classes corresponding to private coin interactive proofs were denoted $\mathsf{IP}[f(n)]$ by Goldwasser, Micali, and Rackoff.

Major Results. One year after their introduction, these complexity classes were proven to be essentially equivalent: any language recognized by a private coin interactive protocol with $f(n)$ rounds can be recognized by a public coin interactive protocol with at most $f(n) + 2$ rounds. In other words, the ability to hide its internal random coins does little to help the verifier. This result was proven by Goldwasser and Sipser [GS86].

One of the most natural questions to ask about interactive proofs is whether these relaxations of the standard model of proofs, which is captured by the class NP, really helps recognizing more languages – namely, whether IP is strictly more powerful than NP. The most celebrated result in the field of interactive proofs is the proof that $\mathsf{IP} = \mathsf{PSPACE}$. The class PSPACE contains all languages that can be recognized by an algorithm that uses polynomial *space*, but which is allowed *unbounded running time*. PSPACE is believed to be strictly more powerful than NP; in particular, it contains the entire polynomial hierarchy. The proof that $\mathsf{IP} = \mathsf{PSPACE}$ arose from a sequence of works, culminating with the work of Lund, Fortnow, Karloff, and Nisan [LFKN92] and of Shamir [Sha92], who introduced a new proof technique, called arithmetization, that proved extremely fruitful.

Additional Observations. The class IP relaxes NP in two directions, by introducing randomness (and allowing errors), and adding interactions. One might wonder to what extent both relaxations are necessary.

Regarding randomness, it is known that if no randomness is allowed, the class of interactive proof systems collapses back to NP. In fact, it suffices to restrict soundness to be perfect (i.e., for any $x \notin \mathscr{L}$, for any prover $\mathcal{P}'$, $\Pr[(\mathcal{P}', \mathcal{V})(x) = 1] = 0$) for this collapse to happen; on the other hand, restricting completeness to be perfect does not reduce the expressivity of interactive proof systems (any language having an interactive proof system has one with

perfect completeness) [FGM89]. Conversely, relaxing further the requirements by allowing unbounded error (completeness must hold with any probability strictly greater than 1/2, and soundness with any probability strictly lower) does also not change the expressivity of interactive proofs; the resulting class is still equal to PSPACE.

Regarding interaction, the non-interactive version of IP (where verification is still randomized) is the class MA defined in [Bab85]; it contains NP, but the converse is not known.

3.2 Interactive Zero-Knowledge Proofs

Zero-knowledge proofs were introduced in the seminal work of Goldwasser, Micali, and Rackoff [GMR89]. They play a central role in cryptography; their study and their applications to secure computation are the main motivations of this thesis. Informally, a zero-knowledge proof is an interactive proof systems in which it is additionally required that the verifier should not learn anything from his interaction with the prover, beyond the truth of the statement. A typical usecase of zero-knowledge proofs is to ask all participants of some interactive protocol to prove that they behaved honestly, without revealing their private values.

3.2.1 Definitions

The main issue with defining zero-knowledge is that it apparently requires to first define what knowledge is, and what it does mean to gain no additional information. The elegant solution introduced in [GMR89] is to consider that a (potentially malicious) verifier $\mathcal{V}^*$ gains no new information from interacting with a prover $\mathcal{P}$ on a common input x if everything this verifier can compute after interacting with $\mathcal{P}$ can be computed directly from the common input x by an efficient algorithm. Proving zero-knowledge is therefore done by exhibiting such an efficient algorithm; it is called a *simulator* for the zero-knowledge proof. A natural efficiency requirement would be to assume that the simulator runs in probabilistic polynomial-time; it turns out that this is too restrictive and the definition of [GMR89] relaxes this requirement by letting the simulator run in *expected* polynomial time.

3.2.1.1 Perfect Zero-Knowledge

We start by defining perfect zero-knowledge proof systems, and discuss useful relaxations of this notion afterward.

Definition 3.2.1. *(Perfect Zero-Knowledge Proof System) An n-round perfect zero-knowledge proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathscr{L}$ is an n-round interactive proof system for $\mathscr{L}$ such that for every probabilistic polynomial time $\mathcal{V}^*$ there exists a probabilistic simulator $\mathcal{S}im$ running in* expected polynomial time *such that for every $x \in \mathscr{L}$,*

$$(\mathcal{P}, \mathcal{V}^*)(x) \equiv \mathcal{S}im(x)$$

Alternatively, the simulator can run in strict polynomial time if it is allowed to output a special symbol $\perp$ on input x with probability at most 1/2. The above definition only asks the simulator to compute the same output than $\mathcal{V}^*$; a seemingly stronger definition would be to let the simulator output the *entire view* of $\mathcal{V}^*$ during its interaction with $\mathcal{P}$. In facts, this does not change the definition, as a simulator must exist for every adversarial verifier $\mathcal{V}^*$, and

in particular for all verifiers that output their entire view; however, this alternative definition is convenient to work with. Let us denote $\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ the view of $\mathcal{V}^*$ when interacting with $\mathcal{P}$ on common input x, i.e., the sequence of all its local configurations. Without loss of generality, we can assume that $\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ consists of the internal random tape of $\mathcal{V}^*$, together with the sequence of all messages he received from $\mathcal{P}$ (as all its local configurations can be deterministically computed from this). Having defined perfect zero-knowledge proofs, we can introduce the class of languages they capture:

Definition 3.2.2. *(The Class* PZK*) A language $\mathscr{L}$ is in the class* PZK *if it has a perfect zero-knowledge proof system with a polynomial number of rounds (in its input length).*

3.2.1.2 Statistical Zero-Knowledge

A natural relaxation of the above definition is to require instead that for all x, the statistical distance between $\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ and $\mathcal{S}im(x)$ should be negligible in $|x|$. To simplify notations, we write $X \overset{\mathsf{stat}}{\equiv} Y$ to say that X is statistically indistinguishable from Y, and $X \overset{\mathsf{comp}}{\equiv} Y$ to say that X is computationally indistinguishable from Y (i.e., any PPT adversary has negligible advantage in distinguishing X from Y).

Definition 3.2.3. *(Statistical Zero-Knowledge Proof System) An n-round statistical zero-knowledge proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathscr{L}$ is an n-round interactive proof system for $\mathscr{L}$ such that for every probabilistic polynomial time $\mathcal{V}^*$ there exists a probabilistic simulator $\mathcal{S}im$ running in* expected polynomial time *such that for every $x \in \mathscr{L}$,*

$$\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(x) \overset{\mathsf{stat}}{\equiv} \mathcal{S}im(x)$$

Definition 3.2.4. *(The Class* SZK*) A language $\mathscr{L}$ is in the class* SZK *if it has a statistical zero-knowledge proof system with a polynomial number of rounds (in its input length).*

3.2.1.3 Computational Zero-Knowledge

One can relax further the definition of zero-knowledge proof systems by requiring only that the view of $\mathcal{V}^*$ and the output of the simulator must be computationally indistinguishable. The core feature of computational zero-knowledge proof systems is that, while providing meaningful zero-knowledge guarantees, they are very expressive: assuming the existence of one-way function, all languages in NP have a computational zero-knowledge proof systems. No such inclusion is known for PZK or SZK.

Definition 3.2.5. *(Computational Zero-Knowledge Proof System) An n-round computational zero-knowledge proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathscr{L}$ is an n-round interactive proof system for $\mathscr{L}$ such that for every probabilistic polynomial time $\mathcal{V}^*$ there exists a probabilistic simulator $\mathcal{S}im$ running in* expected polynomial time *such that for every $x \in \mathscr{L}$,*

$$\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(x) \overset{\mathsf{comp}}{\equiv} \mathcal{S}im(x)$$

Definition 3.2.6. *(The Class* CZK*) A language $\mathscr{L}$ is in the class* CZK *if it has a computational zero-knowledge proof system with a polynomial number of rounds (in its input length).*

The following theorem was proven in [GMW86], it is the cornerstone of a large number of results in cryptography (and in particular in secure computation):

Theorem 3.2.7. *Assuming the existence of one-way functions,* $\mathsf{NP} \subseteq \mathsf{CZK}$.

Proof. (Sketch) Assume that the parties have performed the setup of a computationally hiding, statistically binding commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$. By Proposition 2.3.7, such a scheme exists under the assumption that one-way functions exist.

Observe that to build a computational zero-knowledge proof system for all of NP, it suffices to build such a system for any NP-complete language, as all languages of NP can be reduced to it, using e.g., Karp reductions. We therefore focus on the following NP-complete problem:

Problem 3.2.8. *(Graph 3-Coloring) Given a graph* $G = ([n], E)$*, determines whether there exists an assignment* $\mathsf{Color} : [n] \mapsto \{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$*, which associates a color to each vertex of the graph, such that no pair of adjacent vertices has the same color.*

We now describe a computational zero-knowledge proof system for graph 3-coloring ($\mathsf{3COL}$). Let $G = ([n], E)$ be the common input to $(\mathcal{P}, \mathcal{V})$. We assume that $\mathcal{P}$ is given a 3-coloring col of G.

1. $\mathcal{P}$ picks a uniformly random permutation π of the color set $\{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$, and computes $\mathsf{col}' = \pi \circ \mathsf{col}$. For each vertex $v \in [n]$, $\mathcal{P}$ computes a commitment-opening pair $(c_v, d_v) \xleftarrow{\$} \mathsf{Commit}(\mathsf{col}'(v))$ and sends c_v to $\mathcal{V}$.
2. $\mathcal{V}$ picks a uniformly random edge $e = (u, v) \in E$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ sends (Color_u, d_u) and (Color_v, d_v) to $\mathcal{V}$. The latter accepts if and only if $\mathsf{Color}_u \neq \mathsf{Color}_v$, and $\mathsf{Verify}(c_u, d_u, \mathsf{Color}_u) = \mathsf{Verify}(c_v, d_v, \mathsf{Color}_v) = 1$.

Completeness follows easily by observing that when col is a valid 3-coloring of G, then so is col'. For soundness, if G is not 3-colorable, then it must necessarily hold that there exists an edge $(u, v) \in E$ such that $\mathsf{Color}_u = \mathsf{Color}_v$; by the binding property of the commitment scheme, if the verifier asked this edge, $\mathcal{P}$ cannot open (c_u, c_v) to different colors, hence $\mathcal{V}$ will reject with probability at least $1/|E|$. While this gives a very large soundness error, the soundness can always be amplified by sequential repetitions of the protocol; typically, sequentially repeating the protocol $\lambda \cdot (|E| + 1)$ times, for some security parameter lambda, ensures that the soundness error is bounded by

$$(1 - 1/|E|)^{\lambda \cdot (|E|+1)} < 3^{-\lambda}.$$

For zero-knowledge, we must exhibit a simulator $\mathcal{Sim}$ which, given the code of some verifier $\mathcal{V}^*$, produces a transcript indistinguishable from $\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(G)$ without knowing a 3-coloring of G. We sketch a description of such a simulator: for every $v \in [n]$, $\mathcal{Sim}$ will pick $\mathsf{Color}'_v \xleftarrow{\$} \{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$ and compute $(c_v, d_v) \xleftarrow{\$} \mathsf{Commit}(\mathsf{Color}'_v)$. Then, $\mathcal{Sim}$ writes c_v on the input-message tape of $\mathcal{V}^*$ for every $v \in [n]$, and runs it until it outputs a query $e \in E$ (treating any invalid query as some predetermined fixed edge). If e corresponds to an edge (u, v) such that $\mathsf{Color}'_u \neq \mathsf{Color}'_v$, $\mathcal{Sim}$ terminates with transcript $(c_1, \cdots, c_n, e = (u, v), \mathsf{Color}'_u, d_u, \mathsf{Color}'_v, d_v)$; otherwise, it restarts the protocol, selecting a new uniformly random tape for $\mathcal{V}^*$. Observe that if $\mathcal{V}^*$ was picking $e = (u, v)$ independently of $(c_1, \cdots, c_n)$, it would hold that $\mathsf{Color}'_u \neq \mathsf{Color}'_v$ with probability $2/3$; it can be shown that this remains essentially true when $\mathcal{V}^*$ is given

$(c_1, \cdots, c_n)$ by the hiding property of the commitment scheme. Similarly, when $\mathcal{S}im$ terminates, its output $(c_1, \cdots, c_n, e = (u, v), \mathsf{Color}'_u, d_u, \mathsf{Color}'_v, d_v)$ is computationally indistinguishable from $\text{VIEW}^{\mathcal{P}}_{\mathcal{V}^*}(G)$. □

For a fully detailed proof of Theorem 3.2.7, see Goldreich's book [Gol06]. Note that this result was later extended to the stronger result $\mathsf{IP} = \mathsf{CZK}$ (see [IY88; BGG+90]).

3.2.1.4 Honest-Verifier Zero-Knowledge.

The definition of zero-knowledge proof systems asks for the existence of an algorithm that can simulate the view of any verifier $\mathcal{V}^*$. Honest-verifier zero-knowledge proof systems are interactive proof systems with a weaker notion of zero-knowledge, called honest-verifier zero-knowledge (HVZK), which only asks for the existence of a simulator for a *single* verifier, which is the honest verifier prescribed by the specification of the protocol.

Definition 3.2.9. *(Honest-Verifier Zero-Knowledge Proof System) An n-round (perfect, statistical, computational) honest-verifier zero-knowledge proof system $(\mathcal{P}, \mathcal{V})$ for a language $\mathscr{L}$ is an n-round interactive proof system for $\mathscr{L}$ such that there exists a probabilistic simulator $\mathcal{S}im$ running in* expected polynomial time *such that for every $x \in \mathscr{L}$,*

$$\text{VIEW}^{\mathcal{P}}_{\mathcal{V}}(x) \equiv \mathcal{S}im(x)$$

where $\equiv$ denotes equality, statistical indistinguishability, or computational indistinguishability.

From HVZK to ZK. At first sight, the honest-verifier zero-knowledge notion might seem too weak to provide meaningfull security guarantees in cryptographic protocols. However, perhaps surprisingly, HVZK proofs can be shown to capture the essence of the challenge of building zero-knowledge proofs. Indeed, as shown by Goldreich, Sahai, and Vadhan in [GSV98], a language $\mathscr{L}$ has a statistical honest-verifier zero-knowledge proof system if and only if it belongs to SZK. A similar statement was established for public-coin computational zero-knowledge proof systems. Thus, in essence, honest-verifier zero-knowledge proof systems and zero-knowledge proof systems capture the same languages.

This equivalence is not a purely theoretical observation: it turns out that, when considering practical zero-knowledge proofs in a classical model known as the common reference string model (which will be discussed later on), the conversion from a HVZK proof to a ZK proof can be done very efficiently, at a small, constant additive cost in communication and computation (see e.g. [Gro04; GMY06]).

3.2.2 Brief Survey of Known Results

We have seen above that $\mathsf{NP} \subseteq \mathsf{CZK}$ assuming one-way functions. In this section, we briefly discuss other aspect of the complexity-theoretic study of zero-knowledge proof systems.

Regarding the classes PZK and SZK, it is known [For87; AH91] that $\mathsf{PZK} \subseteq \mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$, where coAM is the class of languages whose complementary is in AM. This inclusion implies that it is unlikely that $\mathsf{NP} \subseteq \mathsf{SZK}$: this would imply that $\mathsf{AM} \cap \mathsf{coAM}$ contains NP-complete problems, hence that $\mathsf{AM} = \mathsf{coAM}$, which would have surprising consequences, such as the collapse of the polynomial hierarchy. Other complexity-theoretic results on SZK were established by Okamoto [Oka96] who established that $\mathsf{SZK} = \mathsf{coSZK}$ (SZK is closed by complement), and that every language in SZK also has a *public coin* statistical

zero-knowledge proof system (where all coins of the verifier are revealed to the prover during the computation).

Regarding the class CZK, a natural question is to ask whether one-way functions are necessary to prove $\mathsf{NP} \subseteq \mathsf{CZK}$. The first steps toward answering this question were made in [OW93], and were extended in [Vad04; OV07] to give a full characterization of languages in CZK as having a "part" in SZK, and a part from which one-way functions can be constructed.

3.3 Interactive Zero-Knowledge Arguments

We have seen that it is unlikely that $\mathsf{NP} \subseteq \mathsf{SZK}$. The class CZK can be seen as a relaxation of zero-knowledge proof systems, in which we allow the zero-knowledge property to hold only computationally, while maintaining a statistical soundness property. As we observed, this relaxation suffices to capture every language in NP.

In this section, we explore an alternative notion, called interactive zero-knowledge arguments, which can be seen as a relaxation of zero-knowledge proofs dual to CZK: the zero-knowledge property of a zero-knowledge argument is required to hold statistically, while the knowledge-extraction property must only hold computationally. This relaxation is of a different nature than the previous one, as for it to make sense, we must restrict our attention to *computationally bounded provers*, while zero-knowledge proof systems can be described with respect to bounded or unbounded provers. Therefore, the natural setting for zero-knowledge arguments is to focus on languages in NP, and on *efficient* prover which are assumed to hold a membership witness for the statement as auxiliary input.

3.3.1 Definitions

Definition 3.3.1. *(Statistical Zero-Knowledge Argument System) An n-round interactive zero-knowledge argument system $(\mathcal{P}, \mathcal{V})$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for a language $\mathscr{L}$ is any pair of probabilistic polynomial-time algorithms such that the following conditions hold:*

- ***Completeness.*** *$(\mathcal{P}, \mathcal{V})$ is complete, if for any $x \in \mathscr{L}$ with a membership witness w:*

$$\Pr\left[\,(\mathcal{P}(w), \mathcal{V})(x) = 1\,\right] \geq 2/3;$$

- ***Computational Soundness.*** *$(\mathcal{P}, \mathcal{V})$ is sound, if for any $x \notin \mathscr{L}$, for any probabilistic polynomial-time prover $\mathcal{P}'$:*

$$\Pr\left[\,(\mathcal{P}', \mathcal{V})(x) = 1\,\right] \leq 1/3;$$

- ***Statistical Zero-Knowledge.*** *$(\mathcal{P}, \mathcal{V})$ is zero-knowledge, if for every probabilistic polynomial time $\mathcal{V}^*$ there exists a probabilistic simulator $\mathcal{S}im$ running in expected polynomial time such that for every $x \in \mathscr{L}$,*

$$\textsc{View}^{\mathcal{P}}_{\mathcal{V}^*}(x) \overset{\mathsf{stat}}{\equiv} \mathcal{S}im(x)$$

Note that perfect zero-knowledge argument systems and computational zero-knowledge argument systems can be defined in a similar fashion.

Definition 3.3.2. *(The Classes* PZKA,SZKA,CZKA*) A language $\mathscr{L}$ is in the class* SZKA *(resp.* PZKA*,* CZKA*) if it has a statistical (resp. perfect, computational) zero-knowledge argument system with a polynomial number of rounds (in its input length).*

Using the same methodology than in the proof of Theorem 3.2.7, one can prove that every language in NP has a statistical zero-knowledge argument system. The proof goes on by replacing the statistically binding, computationally hiding commitment scheme with a statistically hiding, computationally binding commitment scheme. As the latter can also be built from any one-way function [HR07; NOV06], we have:

Theorem 3.3.3. *Assuming the existence of one-way functions,* $\mathsf{NP} \subseteq \mathsf{SZKA}$.

3.3.2 Historical Notes

Zero-knowledge argument were introduced in [BCC88]. They proved to be a very powerful relaxation of zero-knowledge proof systems. In particular, it is possible to construct zero-knowledge argument systems with very strong *succinctness* requirements, where the communication complexity (and sometimes the work of the verifier) can be made sublinear in (or even independent of) the witness size [Kil92; IKO07; BCC+16]. In comparison, zero-knowledge proofs (or even standard interactive proofs) with sublinear communication would have surprising consequences [GH98; GVW02].

3.4 Proofs and Arguments of Knowledge

Zero-knowledge proofs, and their relaxed version, zero-knowledge arguments, allow to prove statements of the form $x \in \mathscr{L}$ (i.e., membership statements). Restricting our attention to NP-languages, such statements can be phrased as existential statements, of the form $\exists w, R_{\mathscr{L}}(x, w) = 1$. Proofs of knowledge strengthen the security guarantee given by classical zero-knowledge proofs. While a zero-knowledge proof suffices to convince the verifier of the *existence* of a witness w for the statement, a proof of knowledge additionally proves that the prover *knows* such a witness.

Several remarks are in order here. First, observe that when considering unbounded prover, this distinction does not make sense, as an unbounded prover can always compute a witness if there exists one. However, when restricting our attention to computationally bounded provers, the distinction makes sense, as a witness can potentially exist for a statement, even though the limited prover might not be able to compute it.

Second, we have to define what it means for a prover to *know* such a witness. Informally, this is done as follows: we say that a party, modeled as a Turing machine, knows a value is the machine can be easily modified so as to output it. More specifically, we will say that an (efficient) algorithm A knows a value w if we can build another efficient algorithm which, given access to A (e.g. by getting the code of A), can output w. Such an algorithm is called an *extractor* for A. Intuitively, this allows to define proofs of knowledge for a statement $x \in \mathscr{L}$ as follows: the soundness property is replaced by a knowledge-extraction property, which states that for every efficient algorithm $\mathcal{P}^*$ such that $(\mathcal{P}^*, \mathcal{V})(x) = 1$, there exists an efficient extractor Ext which, given access to $\mathcal{P}^*$, can compute a witness w such that $R_{\mathscr{L}}(x, w) = 1$. By efficient, we mean that the running time of the extractor should be inversely related to the success probability of $\mathcal{P}^*$.

Third, an important property of proofs of knowledge is that they can make sense even for statements that are trivial from an existential point of view, i.e., for trivial languages for which a membership witness always exists, but can be hard to compute. We illustrate this with a classical example:

Example 3.4.1. *Let $\mathscr{L}_{\mathsf{dlog}}(\mathbb{G}, g)$ denote, for a cyclic group $(\mathbb{G}, \cdot)$ with a generator g, the following language:*

$$\mathscr{L}_{\mathsf{dlog}}(\mathbb{G}, g) = \{h \in \mathbb{G} \mid \exists x \in \mathbb{Z}, g^x = h\}$$

As g is a generator of $\mathbb{G}$, this is a trivial language: all elements of $\mathbb{G}$ belong to $\mathscr{L}_{\mathsf{dlog}}$ (in other words, $\mathbb{G} = \mathscr{L}_{\mathsf{dlog}}$). However, although it holds that for all $h \in \mathbb{G}$, there is an integer x such that $h = g^x$, *computing* such an integer x can be computationally infeasible (see the discussion on the discrete logarithm assumption, Section 2.2.1). Therefore, while asking a prover to show the existence of the discrete logarithm of some word h is meaningless, convincing a verifier that a prover *knows* the discrete logarithm of h in base g gives him a non-trivial information.

As an example of a typical use case of zero-knowledge arguments of knowledge, consider the issue of authenticating a server: to allow for secure communication with the clients, a server releases his public key pk, and stores the corresponding secret key sk (in Example 3.4.1, pk could be a group element h, and sk an integer x such that $g^x = h$). Before interacting with the server, a client might want to be sure that he is talking to the right server, and not with some potentially malicious individual. To do so, the client typically asks his opponent to perform a zero-knowledge proof that he knows the secret key sk corresponding to pk; a successful proof authenticates the server.

3.4.1 Definitions

Before defining zero-knowledge proofs of knowledge, let us introduce some notations. Recall that A^B indicates that A is given oracle access to B.

Definition 3.4.2. *(Next-Message Function) For any algorithm A, we denote by $\mathsf{nm}_{x,w;r}[A]$ the* next-message function *of A, i.e., the algorithm that on input a list m of messages, outputs the next message sent by A after receiving these messages for a common input x, an auxiliary input w, and a random tape r.*

The next-message function allows to formalize the fact that the extractor will be given a fine-grained oracle access to the prover algorithm in the knowledge-extraction procedure.

Definition 3.4.3. *(Zero-Knowledge Proof of Knowledge) An n-round interactive (perfect, statistical, computational) zero-knowledge proof of knowledge $(\mathcal{P}, \mathcal{V})$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, for a language $\mathscr{L}$ with relation $R_{\mathscr{L}}$, is any pair of algorithms such that $\mathcal{V}$ runs in probabilistic polynomial time and the following conditions hold:*

- ***Completeness.*** *$(\mathcal{P}, \mathcal{V})$ is complete, if for any $x \in \mathscr{L}$ with a membership witness w:*

$$\Pr\left[\,(\mathcal{P}(w), \mathcal{V})(x) = 1\,\right] \geq 2/3;$$

- ***Knowledge Extraction.*** *$(\mathcal{P}, \mathcal{V})$ is knowledge-extractable with knowledge error κ, if there exists an efficient algorithm Ext and a polynomial p such that for any input x, for*

any prover $\mathcal{P}^*$, the oracle algorithm $\mathsf{Ext}^{\mathsf{nm}_{x,w;r}[\mathcal{P}^*]}$ runs in expected polynomial time and satisfies

$$\Pr[w' \leftarrow \mathsf{Ext}^{\mathsf{nm}_{x,w;r}[\mathcal{P}^*]} \ : \ R_{\mathscr{L}}(x,w') = 1] \geq \frac{\varepsilon - \kappa}{p(|x|)}$$

where ε denotes the probability that $\mathcal{V}$ accepts when interacting with $\mathcal{P}^*$ on common input x.

- ***Zero-Knowledge**. $(\mathcal{P}, \mathcal{V})$ is (perfectly, statistically, computationally) zero-knowledge, if for every probabilistic polynomial time $\mathcal{V}^*$ there exists a probabilistic simulator $\mathcal{S}im$ running in expected polynomial time such that for every $x \in \mathscr{L}$,*

$$\textsc{View}^{\mathcal{P}}_{\mathcal{V}^*}(x) \equiv \mathcal{S}im(x)$$

where $\equiv$ denotes equality, statistical indistinguishability, or computational indistinguishability.

The parameter κ specifies the error made by Ext when extracting a witness. The knowledge error can be made exponentially small by sequential repetitions of the proof system, see [Gol04].

Definition 3.4.4. *(The Classes PZKPoK, SZKPoK, CZKPoK) A language $\mathscr{L}$ is in the class SZKPoK (resp. PZKPoK, CZKPoK) if it has a statistical (resp. perfect, computational) zero-knowledge proof of knowledge system with a negligible knowledge error and a polynomial number of rounds (in its input length).*

The zero-knowledge proof system defined in the proof of Theorem 3.2.7, on common input $G = ([n], E)$, can be proven to be a zero-knowledge proof of knowledge of a 3-coloring of G with knowledge error $1 - 1/|E|$. Therefore, we get:

Theorem 3.4.5. *Assuming the existence of one-way functions, $\mathsf{NP} \subseteq \mathsf{CZKPoK}$.*

3.5 Σ-Protocols

The previous sections focused on discussions and abstract descriptions of the security properties of zero-knowledge proof systems. The aim of this section is to provide more concrete examples, by considering a specific class of zero-knowledge proof systems to which most efficient zero-knowledge protocols from the literature belong: Σ-protocols.

3.5.1 Definition

A Σ-protocol is an honest-verifier zero-knowledge proof of knowledge, with a particular three-move structure. While they are only honest-verifier zero-knowledge, standard techniques (e.g. [Gro04; GMY06]) can be used to turn any Σ-protocol into full-fledged zero-knowledge proofs of knowledge, at the cost of an additional round of interaction (plus a small additive cost in communication).

Definition 3.5.1. *(Σ-Protocol) A Σ-protocol for a language $\mathscr{L}$ is a public-coin three-move honest-verifier zero-knowledge proof of knowledge, that has the following structure:*

1. *$\mathcal{P}$ sends to $\mathcal{V}$ some* commitments *values r,*

Protocol Π_{dlog}

Common Input: the description of a prime-order group $\mathbb{G}$ of (exponentially large) order p with a generator g, and a group element h.

Prover Witness: A value $x \in \mathbb{Z}_p$ such that $g^x = h$.

Protocol:

1. $\mathcal{P}$: pick $r \xleftarrow{\$} \mathbb{Z}_p$, send $\rho \leftarrow g^r$.
2. $\mathcal{V}$: pick $e \xleftarrow{\$} \mathbb{Z}_p$, send e.
3. $\mathcal{P}$: send $d \leftarrow e \cdot x + r \bmod p$

Verification: $\mathcal{V}$ accepts iff $g^d = h^e \rho$.

Figure 3.1: The Schnorr Σ-protocol for proving knowledge of a discrete logarithm

2. *$\mathcal{V}$ sends to $\mathcal{P}$ a uniformly random* challenge *e,*

3. *$\mathcal{P}$ sends to $\mathcal{V}$ an* answer *$f(w, r, e)$ where f is some public function, and w is the witness held by $\mathcal{P}$*

The attentive reader might have already noticed that the zero-knowledge proof system described in the proof of Theorem 3.2.7 satisfies this three move structure, and indeed, this protocol is a Σ-protocol (although we have only sketched a proof of soundness, it can be proven to be knowledge-extractable as well). Below, we will provide further examples of Σ-protocols.

3.5.1.1 First Example: the Schnorr Protocol

In Section 3.4, we illustrated proofs of knowledge with Example 3.4.1, mentioning the possibility to prove knowledge of the discrete logarithm of some group element h in some base g, where g is the generator of some group $\mathbb{G}$. We now elaborate on this example by describing a Σ-protocol for proving knowledge of a discrete logarithm. The protocol is given Figure 3.1. It was first described in [Sch90]. It is commonly used as an authentication protocol: given a public value h, the prover authentifies himself by proving his knowledge of the secret value x associated to this public value (i.e., x is such that $g^x = h$ for a fixed generator g).

Rewinding. The standard solution to prove security of Σ-protocols is to use a technique called *rewinding*. The simulator will run the code of the prover, feeding it with the verifier inputs it requires, and then rewind it to some previous state so as to feed it with different inputs. Doing so, he will be able to get several outputs of the prover with respect to different verifier inputs, starting from some common state of the prover. Intuitively, this allows the simulator to cancel out some randomness that had been introduced by the prover to mask his witness and ensures that the proof will remain zero-knowledge.

Rewinding is also widely used to prove the zero-knowledge property against a potentially malicious verifier $\mathcal{V}^*$. Here, the simulator will rewind the verifier many times, until he is able to generate an accepting transcript with respect to a run of this verifier.

Security Analysis (Sketch). We show that the protocol Π_{dlog} given Figure 3.1 is perfectly complete, knowledge-extractable, and honest-verifier zero-knowledge. Perfect completeness follows immediately by inspection: if $d = ex + r \bmod p$, $g = h^x$ and $\rho = h^r$, then $g^d = h^e\rho$.

For honest-verifier zero-knowledge, let $\mathcal{Sim}$ be a simulator which is given the common input $(\mathbb{G}, g, h)$ and the code of $\mathcal{V}$. $\mathcal{Sim}$ selects a uniformly random tape for $\mathcal{V}$ and runs it with this random tape on a random input message $\rho \xleftarrow{\$} \mathbb{G}$. Once $\mathcal{V}$ outputs a challenge e, $\mathcal{Sim}$ restarts the protocol, feeding $\mathcal{V}$ with the same random tape and setting the input message ρ to $g^r h^{-e}$ for a uniformly random r. Note that ρ is distributed exactly as in an honest execution of the protocol. After $\mathcal{V}$ outputs the challenge e (as $\mathcal{V}$ is honest, it always draw e honestly, using only the coins of his random tape, hence this challenge is the same than the one extracted by $\mathcal{Sim}$ in the previous run of $\mathcal{V}$), $\mathcal{Sim}$ answers with $d \leftarrow r$; observe that the equation $g^d = h^e\rho$ is always satisfied when $d = r$ and $\rho = g^r h^{-e}$, and that the answer is distributed exactly as in an honest run of Π_{dlog}, hence the honest-verifier zero-knowledge property.

For knowledge-extraction, let $\mathcal{P}^*$ be a prover that produces an accepting answer with non-negligible probability ε, and let $\mathcal{Sim}'$ be a simulator which is given the code of $\mathcal{P}^*$ as input. Once $\mathcal{P}^*$ outputs the first flow ρ, $\mathcal{Sim}'$ writes a random $e \xleftarrow{\$} \mathbb{Z}_p$ on its message input tape, and get an answer d. Then, $\mathcal{Sim}'$ rewinds $\mathcal{P}^*$ to step 2 of Π_{dlog}, feeding it with a new random challenge $e' \xleftarrow{\$} \mathbb{Z}_p$, and getting a new answer d'. Observe that if both (d, d') are accepting answers, it holds that $g^d = h^e\rho$ and $g^{d'} = h^{e'}\rho$, which gives $g^{d-d'} = h^{e-e'} = g^{x\cdot(e-e')}$. In this case, $\mathcal{Sim}'$ can obtain x by computing $(d - d')(e - e')^{-1} \bmod p$ (as $e' \neq e$ with overwhelming probability).

3.5.1.2 Second Example: Disjunction of Languages

We further elaborate on the example of Section 3.5.1.1 by showing that the techniques it involves do extend to more complex statements, such as proof of knowledge of a discrete logarithm in two bases, or disjunction of statements. Let us fix a prime-order group $\mathbb{G}$ with a generator g, and consider the following statement for words $(h, u, v, u', v') \in \mathbb{G}^5$:

$$\text{“I know } x \text{ such that } g^x = h, \text{ or I know } y \text{ such that } u^y = u' \text{ and } v^y = v'\text{.”}$$

Let us call statement 1 the left part of the above statement, and statement 2 its right part. Statement 1 is exactly the statement in the Schnorr protocol. Statement 2 corresponds to a proof of knowledge of a witness for the language of DDH tuples over $\mathbb{G}$ (i.e., tuples of the form (u, v, u^y, v^y)). Note that the latter is also a meaningful membership statement, as the language of DDH tuples is a non-trivial language (assuming the hardness of the DDH assumption, it is a hard-subset-membership language), unlike statement 1. Section 3.5.1.1 gives a Σ-protocol for statement 1; we describe a Σ-protocol for statement 2 in Figure 3.2. Note that this protocol is a very natural generalization of the Schnorr protocol; it is straightforward to extend the security proof of the Schnorr protocol into a security proof for the protocol of Figure 3.2.

Disjunction of Statements. Observe that the zero-knowledge property of the protocols Π_{dlog} and Π_{DDH} essentially stems from the following observation: if the prover knows the challenge e in advance, he can make the verification succeed even if he does not know a witness for the statement (see the security analysis in 3.5.1.1). This exact observation is the key to the standard method for proving disjunctions of statements, which was introduced in [CDS94]. We outline the method on Figure 3.3.

Protocol Π_{DDH}

Common Input: the description of a prime-order group $\mathbb{G}$ of (exponentially large) order p, and four group element (u, v, u', v').

Prover Witness: A value $y \in \mathbb{Z}_p$ such that $u^y = u'$ and $v^y = v'$.

Protocol:

1. $\mathcal{P}$: pick $r \xleftarrow{\$} \mathbb{Z}_p$, send $(\rho, \sigma) \leftarrow (u^r, v^r)$.
2. $\mathcal{V}$: pick $e \xleftarrow{\$} \mathbb{Z}_p$, send e.
3. $\mathcal{P}$: send $d \leftarrow e \cdot y + r \bmod p$

Verification: $\mathcal{V}$ accepts iff $u^d = (u')^e\rho$ and $v^d = (v')^e\sigma$.

Figure 3.2: Σ-protocol for proving knowledge of a witness for a DDH tuple

The intuition behind this protocol is that $\mathcal{P}$ will chose in advance one of the two challenges for (Π_0, Π_1) – the challenge corresponding the statement for which he does not know a witness. The two challenges used by $\mathcal{V}$ are (e_0, e_1) such that $e_0 + e_1 = e$, where e is a random challenge picked by $\mathcal{V}$ in step 2. This ensures that $\mathcal{P}$ can choose in advance one of (e_0, e_1), but has absolutely no information on the remaining challenge before e is sent. Hence, $\mathcal{P}$ can simulate one of the two proofs, but is forced to honestly play the other one. Zero-knowledge is implied by the zero-knowledge property of Π_0, Π_1, as the simulated proofs are indistinguishable from honest proofs. Knowledge extraction also follows from the knowledge extraction property of Π_0, Π_1 and ensures that a simulator can extract a witness for one of the two statements from any successful prover. As it also has the required structure, the protocol Π_S is a Σ-protocol. Applying this method to Π_{dlog} and Π_{DDH} immediately leads to a protocol for the statement given at the beginning of this section.

Protocol Π_S

Given a statement $S = S_0 \vee S_1$, which is a disjunction of two statements S_0, S_1, let (Π_0, Π_1) denote Σ-protocols for (S_0, S_1). The protocol Π_S for S proceeds as follows:

Common Input: the common input of Π_0 and Π_1

Prover Witness: $\mathcal{P}$ know a witness w_b to the statement S_b for some $b \in \{0, 1\}$; w_b is also a witness for the disjunction statement S.

Protocol:

1. $\mathcal{P}$: pick a random challenge $e_{1-b} \xleftarrow{\$} \mathbb{Z}_p$. Compute the first message m_b of the protocol Π_b honestly, and use the zero-knowledge simulator of Π_{1-b} to build a simulated first message m_{1-b}, using the challenge e_{1-b}. Send (m_0, m_1).
2. $\mathcal{V}$: pick $e \xleftarrow{\$} \mathbb{Z}_p$, send e.
3. $\mathcal{P}$: set $e_b \leftarrow e - e_{1-b} \bmod p$. Compute the last message m'_b of the protocol Π_b with challenge e_b from the verifier, and simulate the last message m'_{1-b} of Π_{1-b} using the zero-knowledge simulator with challenge e_{1-b}. Send (e_0, m'_0, m'_1).

Verification: $\mathcal{V}$ sets $e_1 \leftarrow e - e_0 \bmod p$, and perform the verification procedure of Π_0 with transcript (m_0, e_0, m'_0), and Π_1 with transcript (m_1, e_1, m'_1).

Figure 3.3: Σ-protocol for proving the disjunction of statements $S = S_0 \vee S_1$

3.6 The Common Reference String Model

All the results of the previous sections relied on interactive protocols with strong security guarantees without making any trust assumption whatsoever. This is known as the plain model, and it provides the highest real-world security guarantees in an adversarial context. However, the absence of any form of trust strongly narrows the range of feasibility results: several desirable properties, either related to the security or to the efficiency of interactive proof systems, are provably *unachievable* in the plain model. Consider for example the important question of building zero-knowledge proofs with a small number of rounds of interaction. We know that there is no hope of building a zero-knowledge proof system in the plain model with a single round of interaction for non-trivial languages [GO94], and strong limitations are also known for two rounds of interaction [GO94; BLV03]. Regarding security, a highly desirable property is that of composability: a secure proof system should remain secure even if arbitrarily many instances of the system are run concurrently as parts of a larger protocol. However, by the seminal work of Canetti [Can01], we know that *universally composable* zero-knowledge proof systems exist only for trivial languages.

Numerous other limitations are known for zero-knowledge proof systems in which the simulator makes only a black-box use of the verifier; for example, constant-round zero-knowledge proof systems secure under concurrent composition are impossible to achieve in the plain model with black-box simulators [CKPR01]. Consequently, any zero-knowledge proof system overcoming these limitations in the plain model must make use of non-black-box techniques, which are often inefficient in practice.

Functionality $\mathcal{F}_{\mathsf{crs}}$

$\mathcal{F}_{\mathsf{crs}}$ is parametrized by a distribution $\mathcal{D}$ and maintains a dictionary Dict of stored values, which is initially empty. When activated on an input (input, sid) for a session identifier sid, $\mathcal{F}_{\mathsf{crs}}$ retrieves $\mathsf{Dict}[sid]$.

- If $\mathsf{Dict}[sid]$ is empty, $\mathcal{F}_{\mathsf{crs}}$ picks $\mathsf{crs} \xleftarrow{\$} \mathcal{D}$, returns crs to the activating party, and stores $\mathsf{Dict}[sid] \leftarrow \mathsf{crs}$.
- If $\mathsf{Dict}[sid] = \mathsf{crs}$, returns crs to the activating party.

Figure 3.4: The common reference string ideal functionality [CF01]

3.6.1 Trusted Setup Assumptions

In light of these strong limitations, a natural question to ask is whether there exists *minimal* trust assumptions one could make that lead to a model in which *practically efficient* zero-knowledge proof systems with *strong security guarantees* can be built. In this work, we will consider zero-knowledge proof systems in a model known as the common reference string (crs) model, which was introduced by Damgård in [Dam00]. In this model, the parties are given access to a common string that has been honestly drawn from some prescribed distribution in a setup phase by a trusted dealer. More formally, the crs model enhances the plain model by giving both players access (via perfectly secure authenticated channels) to an ideal functionality $\mathcal{F}_{\mathsf{crs}}$, represented Figure 3.4. While weaker models have been proposed (such as the common random string model, in which crs comes from the uniform distribution, or the registered public-key model), the common reference string model has proven very convenient to use for constructing a large variety of efficient primitives with strong security requirements.

3.6.2 Proving Security of Zero-Knowledge Proof Systems

At an intuitive level, most impossibility results in the plain model stem from the fact that soundness and zero-knowledge appear somewhat contradictory. Consider for simplicity the case of zero-knowledge proofs of knowledge. The zero-knowledge property ensures that nothing leaks from the transcript of the protocol, while soundness ensures that a simulator can extract a valid witness from the prover. If the verifier could simply run the simulator algorithm and extract the witness of the prover, the proof could not possibly be zero-knowledge. What makes such proofs possible is that the simulator is given some additional power that the verifier does not have. Therefore, it is crucial to analyze exactly what kind of extra power can be given to the simulator. A natural solution is to give the simulator the *code* of the prover, as this code is not available to the verifier. However, this raises the issue of how to extract the relevant information from this code, which could possibly be non-trivial (think for example of an obfuscated code). Below, we outline two standard method for proving security of zero-knowledge proof systems. The first one, the rewinding technique, will be extensively used in Chapter 5 and Chapter 6. The second one is specific to the common reference string model but is convenient to prove the security of the system in a composable setting; we will use it in Chapter 4.

3.6.2.1 Rewinding

The rewinding strategy, which was informally introduced and illustrated in Section 3.5.1.1, is the most standard method to prove security of zero-knowledge protocols in the plain model. It is conceptually simple, and only requires to give the simulator black-box access to the next-message function of his opponent. However, this method fails in various setting – most notably, rewinding often fails in multiparty settings where several instances of the protocols can be composed together, as the number of necessary rewinding can grow exponentially in these situations, preventing the simulator from running in (expected) polynomial time. In such situations, one must rely on alternative proof strategies.

3.6.2.2 Disjunction With a Hard Subset-Membership Problem

Rewinding has the advantage of not requiring any setup assumption. However, if one is willing to assume such setup assumption, as we will do in this work, alternative proof methods are available. Recall that the crs model assumes that all players have access to some reference string drawn from a known distribution $\mathcal{D}$. This suggests another way of giving the simulator the extra power he needs over the honest prover to simulate the protocol: the simulator will simulate not only the prover, but also the crs functionality, which will allow him to generate an alternative crs from a different distribution $\mathcal{D}'$. This alternative crs should allow him to generate valid proofs on arbitrary statements. Of course, the distribution $\mathcal{D}'$ should be indistinguishable from $\mathcal{D}$, otherwise the verifier could distinguish the simulation from a real execution.

Intuition. More specifically, we outline below a standard method that we will use in this work. Let the common reference string contain the description of an hard-subset-membership language $\mathscr{L}$: for random x, it should be computationally infeasible to find out whether $x \in \mathscr{L}$ holds. The distribution $\mathcal{D}$ outputs a description of $\mathscr{L}$ together with a random $x \notin \mathscr{L}$. Now, to prove a statement S, the prover will instead prove the statement $S' = S \vee (x \in \mathscr{L})$. For honestly generated reference strings, S' gives exactly the same guarantees than S does. On the other hand, the simulator will modify the crs distribution and generate instead a random $x \in \mathscr{L}$, keeping the associated witness w. The modified crs is computationally indistinguishable from an honestly generated crs, by the hard-subset-membership property of $\mathscr{L}$. However, all statements of the form $S' = S \vee (x \in \mathscr{L})$ now become trivially true, and the witness w is a valid witness for these statements, giving the simulator a way to prove them. For this method to work, the underlying proof system must only guarantee *witness indistinguishability*, meaning that the protocol should not leak information on which witness was used by the prover, when several witnesses are available. This notion is weaker than zero-knowledge, and witness-indistinguishable proof systems are typically easier to build than zero-knowledge proof systems.

Example. Consider the statement "I know the discrete logarithm of h in base g" for some elements (g, h) of a prime-order group $\mathbb{G}$, as considered in Section 3.5.1.1. Let us build a zero-knowledge proof system in the common reference string model for this statement as follows: the common reference string contains the description of $\mathbb{G}$, a generator g, and a four-tuple $(u, v, u', v') \in \mathbb{G}$ of uniformly random group elements. The protocol is constructed as the disjunction of the two Σ-protocols Π_{dlog} (on the word (g, h)) and Π_{DDH} (on the word (u, v, u', v')), as described in Section 3.5.1.2. The resulting protocol is therefore a Σ-protocol for the statement "I know x such that $g^x = h$, or I know a witness proving that (u, v, u', v')

is a DDH tuple".

For an honestly generated crs, (u, v, u', v') is not a DDH tuple with overwhelming probability, hence the above protocol ensures that the prover knows the discrete logarithm of h. However, to simulate the prover for the zero-knowledge property, the simulator can modify the crs distribution so that it outputs a tuple (u, v, u^y, v^y) instead, for some random (u, v) and a random exponent y of his choice. Under the DDH assumption, this is indistinguishable from an honestly generated crs, but the trapdoor y is also a witness for the statement "(u, v, u', v') is a DDH tuple", hence the simulator can play the role of the prover with this witness. Observe that he does not need to rewind the verifier to do so, which is a desirable property to analyze the behavior of the protocol in a concurrent setting.

3.6.3 Simulation Soundness

We have seen above that the common reference string model allows to design zero-knowledge proofs that remain secure when composed with other cryptographic protocols. However, in most scenarios that involve the composition of zero-knowledge proofs, their soundness property does not suffice anymore to ensure the security of the entire protocol: indeed, when proving the security of the protocol, the simulator might produce simulated proofs (possibly on false statements). The soundness property does not guarantee that an adversary could not break the security of the protocol *given access to such simulated proofs*. Informally, simulation soundness requires that soundness still holds when the adversary is given access to a simulation oracle. Formally defining simulation soundness requires defining a zero-knowledge proof system with an explicit simulator algorithm, and thus can depend on the exact type of proof system that is considered. We do not attempt to give a formal definition here, but we will later formally define simulation soundness for the new type of zero-knowledge proof system that we introduce in Chapter 4.

3.7 Zero-Knowledge Arguments over the Integers

The protocols Π_{dlog} of Section 3.5.1.1 and Π_{DDH} of Section 3.5.1.2 can be naturally extended to prove a large variety of statements, such as arbitrary algebraic relations between values, committed (e.g., with a Pedersen commitment, see Section 2.3.2) or encrypted (e.g., with the ElGamal encryption scheme, see Section 2.3.3). Such algebraic statements arise naturally in many scenarios.

3.7.1 Zero-Knowledge Proofs of Non-Algebraic Statements

The above method also allows for proving arbitrary non-algebraic statements $x \in \mathscr{L}$: the prover commits to every bit of the witness w, proves with a disjunction proof that each commitment commits to either 0 or 1, and proves that the polynomial relation $R_{\mathscr{L}}(x, w)$ between the public x and the committed bits of w evaluates to 1. However, this method is quite inefficient in general; in particular, it requires to exchange at least a number of group elements proportional to the length of the witness. To address this issue, a number of solutions have been suggested, such as garbled-circuit-based zero-knowledge proofs for statements expressed by boolean circuits [JKO13; FNO15], which only use symmetric-key operations for each gate of the circuit, or zero-knowledge arguments with sublinear communication based on generalized Pedersen commitments [Gro09; Gro11].

An alternative approach to the above have been suggested by Lipmaa in [Lip03], and stems from ideas that can be traced to the work of Boudot [Bou00]. The observation is that several non-algebraic statements that naturally arise in applications can be efficiently expressed as Diophantine relations. Therefore, such statements can be efficiently proven if we can commit and prove relations between *integer values*.

3.7.2 Range Proofs

Consider a user who has committed some private values, and is asked to prove that they satisfy a certain relation. In many scenarios, this relation cannot be efficiently expressed algebraically. A common example is the case of range proofs: the prover is asked to show that a committed value belongs to some public range. This situation occurs e.g., in protocols for e-voting, or in anonymous cryptocurrencies. Standard zero-knowledge proof systems for such statements require to see the committed input as a bitstring, and to prove that its bits satisfy some polynomial relation, resulting in a blowup in communication and computation.

The work of Boudot [Bou00] and Lipmaa [Lip03] suggest the following alternative approach: suppose that we have at our disposal an *integer commitment scheme*, that allows to commit to an arbitrary $m \in \mathbb{Z}$, together with a zero-knowledge proof system that allows to prove *integer algebraic relations* between committed integers. Then, membership to a large variety of languages (looking ahead, languages of words that satisfy a Diophantine relations) can be expressed by proving such integer algebraic relations. For example, by a famous result of Lagrange, an integer m belongs to a range $[\![a\,;b]\!]$ if and only if there exists four integers (m_1, m_2, m_3, m_4) such that $(m-a)(b-m) = \sum_i m_i^2$. Therefore, to prove that a committed value m belongs to $[\![a\,;b]\!]$, the prover simply computes the appropriate values (m_1, m_2, m_3, m_4) (using the Rabin-Shallit algorithm [RS86]) and proves that the above relation holds. Below, we generalize this observation to statements that can be expressed by Diophantine relations.

3.7.3 Zero-Knowledge Arguments from Diophantine Relations

A *Diophantine set* $S \subseteq \mathbb{Z}^k$ is a set of vectors over $\mathbb{Z}^k$ defined by a (multivariate) *representing polynomial* $P_S(X, W)$ with $X = (X_1, \cdots, X_k)$ and $W = (Y_1, \cdots, Y_\ell)$, i.e., a set of the form $S = \{\vec{x} \in \mathbb{Z}^k \mid \exists \vec{w} \in \mathbb{Z}^\ell, P_S(\vec{x}, \vec{w}) = 0\}$ for some polynomial P_S. It was shown in [DPR61] that any recursively enumerable set is Diophantine. An interesting class for cryptographic applications is the class $\mathbf{D}$ of Diophantine sets S such that each $\vec{x} \in S$ has at least one witness $\vec{w}$ satisfying $||\vec{w}||_1 \leq (||\vec{x}||_1)^{O(1)}$. It is widely conjectured that $\mathbf{D} = \mathsf{NP}$, as $\mathbf{D}$ contains several NP-complete problems, and it was shown in [Pol03] that if $\mathsf{co\text{-}NLOGTIME} \subseteq \mathbf{D}$, then $\mathbf{D} = \mathsf{NP}$. The class $\mathbf{D}$ was introduced in [AM76] and its cryptographic relevance was pointed out in [Lip03]. For example, the set $\mathbb{Z}_+$ of positive integers is in $\mathbf{D}$, as by a well-known result of Lagrange, it can be defined as $\mathbb{Z}_+ = \{x \in \mathbb{Z} \mid \exists (w_1, w_2, w_3, w_4) \in \mathbb{Z}^4, x - (w_1^2 + w_2^2 + w_3^2 + w_4^2) = 0\}$. In addition, each w_i is of bounded size $||w_i|| \leq ||x||$.

Lipmaa [Lip03] has shown that zero-knowledge arguments of membership to a set $S \in \mathbf{D}$, with representing polynomial P over k-vector inputs and ℓ-vector witnesses, can be constructed using an integer commitment scheme, such as [DF02]. The size of the argument (the communication between $\mathcal{P}$ and $\mathcal{V}$) depends on k, ℓ, and $\deg(P)$, the degree of P. As noted in [Lip03], intervals, unions of intervals, exponential relations (i.e., set of tuples (x, y, z) such that $z = x^y$) and gcd relation (i.e., set of tuples (x, y, z) such that $z = \gcd(x, y)$) are all in $\mathbf{D}$, with parameters (k, ℓ and $\deg(P)$) small enough for cryptographic applications.

3.8 Non-Interactive Zero-Knowledge Arguments

As we have seen previously, interactive proofs can be understood as a relaxation of the standard non-interactive proofs (captured by the class NP), where we allow interaction (as well as random coins) between the verifier and the prover. Zero-knowledge proofs are a randomized interactive proof systems satisfying a specific zero-knowledge property. A natural question is to ask whether this zero-knowledge property can also be satisfied by non-interactive randomized proofs. Such systems are called *non-interactive zero-knowledge proof systems* (NIZK).

This question is also very interesting from a practical point of view: in the real world, interactivity means exchanging information over some network, which raises some latency issues. A highly interactive protocol can be inefficient in a setting where the interacting parties are far away – for example, if two parties are performing a protocol between San Fransisco and London, there is a 100ms roundtrip. Furthermore, while computation will always improve with more computational power, and communication with additional bandwidth, interactivity is inherently limited by the speed of light (e.g., we cannot hope to exchange messages in less that 37ms between San Fransisco and London). Therefore, the more computers improve, the more interactivity becomes a major concern for efficiency.

Eventually, we mention that another motivations for NIZK proofs are their applications to numerous cryptographic primitives. As this works mainly targets applications related to secure computation, we do not discusses this in detail.

3.8.1 Definition and Security Properties

Definition 3.8.1. *(Publicly Verifiable Non-Interactive Zero-Knowledge Proof System) A non-interactive zero-knowledge (NIZK) proof system between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ for a family of languages $\{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ is a triple of probabilistic polynomial-time algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ such that*

- $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa)$*, outputs a common reference string* crs*,*
- $\pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w)$*, on input the crs* crs*, a word x, and a witness w, outputs a proof π,*
- $b \leftarrow \mathcal{V}(\mathsf{crs}, x, \pi)$*, on input the crs* crs*, a word x, and a proof π, outputs $b \in \{0,1\}$,*

which satisfies the completeness, zero-knowledge, and soundness properties defined below.

A few remarks are in order. In the above definition of NIZK proof systems, the verifier algorithm takes only public informations as input in addition to the proof. This gives rise to the notion of *publicly verifiable* NIZK proof system. Alternatively, the Setup can generate a secret verification key vk, which is used by the verification algorithm; this variant is known as *designated-verifier* NIZK proof system. We only consider publicly verifiable NIZKs here.

Definition 3.8.2. *(Perfect Completeness) A NIZK proof system $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for a family of languages $\{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations $R_{\mathscr{L}_{\mathsf{crs}}}$ satisfies the perfect completeness property if for $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, for every $x \in \mathscr{L}_{\mathsf{crs}}$ and every witness w such that $R_{\mathscr{L}_{\mathsf{crs}}}(x, w) = 1$,*

$$\Pr[\pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w) \ : \ \mathcal{V}(\mathsf{crs}, x, \pi) = 1] = 1$$

We now define the zero-knowledge property. The definition we adopt is stronger than needed, but is satisfied by known constructions of efficient NIZK proof systems, and makes the NIZKs more amenable to composition.

Definition 3.8.3. *(Zero-Knowledge) A NIZK proof system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for a family of languages* $\{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* $R_{\mathscr{L}_{\mathsf{crs}}}$ *satisfies the (composable) zero-knowledge property if there exists a* simulated setup *algorithm* Setup' *and a probabilistic polynomial-time simulator* $\mathcal{S}im$ *such that*

- $\mathsf{Setup}'(1^\kappa)$ *outputs a pair* $(\mathsf{crs}', \mathcal{T})$*, where* $\mathcal{T}$ *is a* trapdoor*;*
- $\mathcal{S}im(\mathsf{crs}', \mathcal{T}, x)$*, on input a pair* $(\mathsf{crs}', \mathcal{T})$ *and a word* x*, outputs a simulated proof* π'*;*

which satisfy the following properties:

- *the distributions* $\{\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\kappa) \ : \ \mathsf{crs}\}$ *and* $\{(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}'(1^\kappa) \ : \ \mathsf{crs}\}$ *are indistinguishable, and*
- *for any* $(\mathsf{crs}', \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}'(1^\kappa)$*, any word* $x \in \mathscr{L}_{\mathsf{crs}'}$ *with witness* w*, the distributions* $\{\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}', x, w) \ : \ \pi\}$ *and* $\{\pi' \xleftarrow{\$} \mathcal{S}im(\mathsf{crs}', \mathcal{T}, x) \ : \ \pi'\}$ *are indistinguishable.*

We will adopt a comparable formalism for defining our new variant of zero-knowledge proof systems in Chapter 4. In addition, we will require the following property: the distributions generated by Setup and Setup' should be *statistically* indistinguishable. This is a useful property in settings where many proofs can be composed.

Definition 3.8.4. *(Non-Adaptive Soundness) A NIZK proof system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for a family of languages* $\{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ *with relations* $R_{\mathscr{L}_{\mathsf{crs}}}$ *satisfies the non-adaptive soundness property if for any* $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, every* $x \notin \mathscr{L}_{\mathsf{crs}}$*, and every prover* $\mathcal{P}^*$*,*

$$\Pr[\pi \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \ : \ \mathcal{V}(\mathsf{crs}, x, \pi) = 1] = \mathrm{negl}(\lambda)$$

In the above definition, the word $x \notin \mathscr{L}_{\mathsf{crs}}$ is chosen non-adaptively, before the public parameter are generated. A stronger security property is the *adaptive soundness* property, which allows x to be adversarially picked after the common reference string is fixed.

3.8.2 Brief Survey of Known Results

Non-interactive zero-knowledge proofs have been first introduced in [BFM88]. NIZK proof systems from general assumptions (doubly enhanced trapdoor permutations, or certified trapdoor permutations) have been first introduced in [FLS90]. Our definitions above assume a setup algorithm that generates a common reference string. One can wonder whether NIZKs could be defined in the plain model, without having to assume a trusted setup. Unfortunately, it was shown in [Ore87] that NIZKs in the plain model can exist only for trivial languages, namely, those that are contained in BPP. The common reference string model is not the only one that was proposed for NIZKs; NIZK proof systems have also been defined in the stronger *preprocessing model* [DMP90], or in a different *secret-key model* [CD04].

The class NISZK of languages that admit a NIZK proof system with statistical zero-knowledge has been proven in [GSV99] to be equal to SZK if NISZK is closed by complement. Proving the latter is a well-established open problem.

3.8.3 Fiat-Shamir Heuristic

The Fiat-Shamir heuristic [FS87] is a heuristic method to convert Σ-protocols (see Section 3.5) into non-interactive zero-knowledge proofs. It proceeds as follows: to prove the membership of a word x to a language $\mathscr{L}$ the prover $\mathcal{P}$ first compute the first flow (the commitments) of a Σ-protocol for this statement. Let c denote this first flow. Then, $\mathcal{P}$ sets $e \leftarrow H(x, c)$, where H is some hash function, and computes the last flow of the Σ-protocol, using e as the challenge.

While this approach leads to very efficient NIZKs, it cannot be proven to work under any standard assumption related to hash functions. Instead, the above methodology can be proven to work if the hash function is modeled as a truly random function. This idealized model is known as the random oracle model. Unfortunately, truly random functions are objects of exponential size, and cannot be realized efficiently. In fact, some (contrived) protocols can be proven secure in the random oracle model, but are trivially insecure when instantiated with any concrete hash function [BR93]. Therefore, this abstraction is best seen as a heuristic indication of security.

3.8.4 Groth-Sahai Proofs

For a long time, two types of NIZK proof systems were available: efficient but heuristically secure proof systems in the random oracle model, and inefficient proof systems in the hidden bit model [FLS90], which can be instantiated in the standard model, under well-studied assumptions. This changed with the arrival of pairing-based cryptography, from which a fruitful line of work (starting with the work of Groth, Ostrovsky, and Sahai [GOS06b; GOS06a]) introduced increasingly more efficient NIZK proof systems in the standard model. This line of work culminated with the framework of Groth-Sahai proofs [GS08], which identified a restricted yet very powerful class of languages for which efficient pairing-based NIZK could be designed, with security based on essentially any standard assumption on pairing-friendly groups. This framework (which was subsequently optimized in [GSW10; BFI+10; EG14]) paved the road to numerous practical applications. In the next chapter, we aim at providing an alternative solution to one of those applications, round-efficient two-party computation.

4

Implicit Zero-Knowledge Arguments

It is my experience that proofs involving matrices can be shortened by 50% if one throws the matrices out.

– Emil Artin, *Geometric Algebra*

4.1 Introduction

Zero-Knowledge Arguments have found numerous applications in cryptography, most notably to simplify protocol design as in the setting of secure two-party computation [Yao86; GMW87b; GMW87a], and as a tool for building cryptographic primitives with strong security guarantees such as encryption secure against chosen-ciphertext attacks [NY90; DDN91]. In this chapter, we focus on the use of zero-knowledge arguments as used in efficient two-party protocols for enforcing semi-honest behavior. We are particularly interested in round-efficient two-party protocols, as network latency and round-trip times can be a major efficiency bottleneck, for instance, when a user wants to securely compute on data that is outsourced to the cloud.

4.1.1 Enforcing Honest Behavior in Two-Party Computation

The study of zero-knowledge as a tool to enforce semi-honest behavior in secure two-party computation was initiated in [GMW87b]. While this seminal work essentially established the theoretical feasability of the method, the developements of efficient zero-knowledge proof systems in the past two decades have led to natural and practically efficient compilers for a large variety of two-party computation protocols, where honest behavior can be captured by membership to algebraic languages. We briefly mention two of the most natural alternatives below.

- Σ-protocols lead to efficient protocols regarding both computation and communication, and can be based on virtually any group-based assumption, such as the discrete logarithm assumption. However, an n-round semi-honest protocol compiled into a actively secure protocol via Σ-protocols will in general have $3n$ rounds, causing a blowup in round efficiency. The Fiat-Shamir transform (see Section 3.8.3) overcomes this issues in the random oracle model, but as random oracles cannot be instantiated in the real world, this can only be seen as a heuristic indication of security [BR93; FS87].

- Alternatively, one can rely on pairing-based non-interactive zero-knowledge proof systems *a la* Groth-Sahai [GS08], see Section 3.8.4. The use of elliptic curves with a pairing makes this alternative less efficient regarding computation (pairings are expensive operations, and exponentiations are up to three times slower on curves with pairings than on curves without pairings), but it remains fairly practical. It allows for provably secure compilation of n-round semi-honest protocols into n-rounds actively secure protocols, in the common reference string model, under pairing-based cryptographic assumptions.

In this chapter, we introduce a new primitive called *implicit zero-knowledge argument* (iZK) that stands philosophically as an intermediate notion between interactive zero-knowledge proofs (such as Σ-protocols) and (designated-verifier) non-interactive zero-knowledge proofs, when used as a tool to compile semi-honest two-party computation protocols into actively secure two-party computation protocols.

The intuition of iZKs is captured by the following observation: standard methods for compiling semi-honest protocols into actively secure protocols require the parties to verify, using zero-knowledge proofs, that their opponents behaved honestly before sending their next message. In contrast, iZK allow the parties to send their next flow masked, so that their opponents will be able to remove the mask if and only if they behaved honestly.

This guarantees that private informations remain hidden to malicious parties, but does not explicitely inform the sending party on the honesty of his opponents – hence the implicit zero-knowledge flavor. It turns out that this approach gives rise to low-interactivity protocols, almost matching the round-efficiency offered by NIZK, while being realizable from a wider class of assumptions (essentially the same class of assumptions on which Σ-protocols can be based). As a byproduct, our construction does not require pairings, which results in important savings in both communication and computation when compared to pairing-based non-interactive zero-knowledge proofs.

Summing up, iZKs allow to compile semi-honest two-party protocols into actively secure protocols under a wide variety of standard assumptions, and lead to efficient protocols regarding both communication and computation. It is also almost as round-efficient as NIZKs: an n-round protocol is compiled into an $(n+2)$-round protocol in general (or $(n+1)$-round in many natural scenarios).

Application to Covert Two-Party Computation. Subsequent to our work, implicit zero-knowledge arguments have also proven valuable tools to construct two-party computation protocols enjoying a very strong security property, known as covertness, which states (informally) that no information should leak from the transcript of a protocol – not even the fact that the parties were taking part to the protocol, and not carrying normal conversations. A covert analogue of Yao's garbled circuit based on implicit zero-knowledge arguments has been proposed in [Jar16a].

4.1.2 On Round-Efficiency

We point out that, contrary to some common belief, there is no straightforward way to reduce the number of rounds of zero-knowledge proofs "à la Schnorr" [Sch90] by performing the first steps (commitment and challenges) in a preprocessing phase, so that each proof only takes one flow subsequently. Indeed, as noticed by Bernhard-Pereira-Warinschi in [BPW12], the statement of the proof has to be chosen before seeing the challenges, unless the proof becomes unsound.

In addition to being an interesting theoretical problem, improving the round efficiency is also very important in practice. If we consider a protocol between a client in Europe, and a cloud provider in the US, for example, we expect a latency of at least 100ms (and even worse if the client is connected with 3g or via satellite, which may induce a latency of up to 1s [Bro13]). Concretely, using Curve25519 elliptic curve of Bernstein [Ber06] (for 128 bits of security, and 256-bit group elements) with a 10Mbps Internet link and 100ms latency, 100ms corresponds to sending 1 flow, or 40,000 group elements, or computing 1,000 exponentiations at 2GHz on one core of current AMD64 microprocessor[1], hence 4,000 exponentiations on a 4-core microprocessor[2]. As a final remark on latency, while speed of networks keeps increasing as technology improves, latency between two (far away) places on earth is strongly limited by the speed of light: there is no hope to get a latency less than 28ms between London and San Francisco, for example.

[1] According to [ECR], an exponentiation takes about 200,000 cycles.
[2] Assuming exponentiations can be made in parallel, which is the case for our iZKs.

4.1.3 Contributions of this Chapter

As already outlined, we introduce in this chapter the notion of *implicit Zero-Knowledge Arguments* or iZK and simulation-sound variants thereof or SSiZK, lightweight alternatives to (simulation-sound) zero-knowledge arguments for enforcing semi-honest behavior in two-party protocols. Then, we construct efficient two-flow iZK and SSiZK protocols for a large class of languages under the (plain) DDH assumption in cyclic groups without random oracles; this is the main technical contribution of our work. Our SSiZK construction from iZK is very efficient and incurs only a small additive overhead. Finally, we present several applications of iZK to the design of efficient secure two-party computation, where iZK can be used in place of interactive zero-knowledge arguments to obtain more round-efficient protocols.

While our iZK protocols require an additional flow compared to NIZK, we note that eliminating the use of pairings and random oracles offers both theoretical and practical benefits. From a theoretical stand-point, the DDH assumption in cyclic groups is a weaker assumption than the DDH-like assumptions used in Groth-Sahai pairing-based NIZK [GS08], and we also avoid the theoretical pitfalls associated with instantiating the random oracle methodology [CGH04; BBP04]. From a practical stand-point, we can instantiate our DDH-based protocols over a larger class of groups. Concrete examples include Bernstein's Curve25519 [Ber06] which admit very efficient group exponentiations, but do not support an efficient pairing and are less likely to be susceptible to recent breakthroughs in discrete log attacks [BGJT14; GKZ14]. By using more efficient groups and avoiding the use of pairing operations, we also gain notable improvements in computational efficiency over Groth-Sahai proofs. Moreover, additional efficiency improvements come from the structure of iZK which makes them *efficiently batchable*. Conversely, Groth-Sahai NIZK cannot be efficiently batched and do not admit efficient SS-NIZK (for non-linear equations).

4.1.4 New Notion: Implicit Zero-Knowledge Arguments

iZK is a two-party protocol executed between a prover and a verifier, at the end of which both parties should output an ephemeral key. The idea is that the key will be used to encrypt subsequent messages and to protect the privacy of a verifier against a cheating prover. Completeness states that if both parties start with a statement in the language, then both parties output the same key K. Soundness states that if the statement is outside the language, then the verifier's ephemeral output key is hidden from the cheating prover. Note that the verifier may not learn whether his key is the same as the prover's and would not be able to detect whether the prover is cheating, hence the soundness guarantee is *implicit*. This is in contrast to a standard ZK argument, where the verifier would "explicitly" abort when interacting with a cheating prover. Finally, zero-knowledge stipulates that for statements in the language, we can efficiently simulate (without the witness) the joint distribution of the transcript between an honest prover and a malicious verifier, together with the honest prover's ephemeral output key K. Including K in the output of the simulator ensures that the malicious verifier does not gain additional knowledge about the witness when honest prover uses K in subsequent interaction, as will be the case when iZK is used as part of a bigger protocol.

More precisely, iZKs are key encapsulation mechanisms in which the public key ipk is associated with a word x and a language $i\mathscr{L}$. In our case, x is the flow[3] and $i\mathscr{L}$ the language

[3] In our formalization, actually, it is the flow together all the previous flows. But we just say it is the flow to

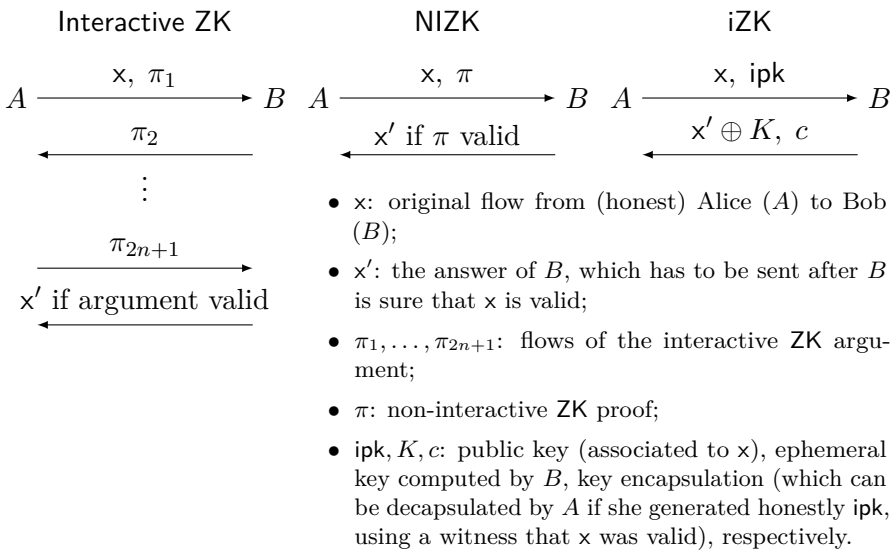


Figure 4.1: Enforcing semi-honest behavior of Alice (A)

of valid flows. If x is in $\mathsf{i}\mathscr{L}$, knowing a witness proving so (namely, random coins used to generate the flow) enables anyone to generate ipk together with a secret key isk, using a key generation algorithm iKG. But, if x is not in $\mathsf{i}\mathscr{L}$, there is no polynomial-time way to generate a public key ipk for which it is possible to decrypt the associated ciphertexts (*soundness*).

To ensure semi-honest behavior, as depicted in Figure 4.1, each time a player sends a flow x, he also sends a public key ipk generated by iKG and keeps the associated secret key isk. To answer back, the other user generates a key encapsulation c for ipk and x, of a random ephemeral key K. He can then use K to encrypt (using symmetric encryption or pseudo-random generators and one-time pad) all the subsequent flows he sends to the first player. For this transformation to be secure, we also need to be sure that c (and the ability to decapsulate K for any ipk) leaks no information about random coins used to generate the flow (or, more generally, the witness of x). This is ensured by the *zero-knowledge* property, which states there must exist a trapdoor (for some common reference string) enabling to generate a public key ipk and a trapdoor key itk (using a trapdoor key algorithm iTKG), so that ipk looks like a classical public key and itk allows to decapsulate any ciphertext for ipk.

4.1.5 Overview of our iZK and SSiZK Constructions

We proceed to provide an overview of our two-flow iZK protocols; this is the main technical contribution of our work. Our main tool is Hash Proof Systems or Smooth Projective Hash Functions (SPHFs) [CS02]. We observe that SPHFs are essentially "honest-verifier" iZK; our main technical challenge is to boost this weak honest-verifier into full-fledged zero knowledge, without using pairings or random oracles. Preliminaries on SPHFs can be found in Section 2.3.4, but to recall briefly, a smooth projective hash function on a language $\mathscr{L}$ is a hash function whose evaluation on a word $C \in \mathscr{L}$ can be computed in two ways, either by using a *hashing key* hk or by using the associated *projection key* hp. When $C \notin \mathscr{L}$, however, the hash of C cannot be computed from hp; actually, when $C \notin \mathscr{L}$, the hash of C computed

simplify explanations.

with hk is statistically indistinguishable from a random value from the point of view of any individual knowing the projection key hp only. In this chapter, as in [GL06], we consider a weak form of SPHFs, where the projection key hp can depend on C.

On Building iZK from SPHF. Concretely, if we have an SPHF for some language $\mathscr{L}$, we can set the public key ipk to be empty ($\perp$), the secret key isk to be the witness w, the ciphertext c to be the projection key hp, and the encapsulated ephemeral key K would be the hash value. (Similar connections between SPHF and zero knowledge were made in [GL03; GL06; BPV12; ABB+13].) The resulting iZK would be correct and sound, the soundness coming from the smoothness of the SPHF: if the word C is not in $\mathscr{L}$, even given the ciphertext $c = \mathsf{hp}$, the hash value K looks random. However, it would not necessarily be zero-knowledge for two reasons: a malicious verifier could generate a malformed projection key, for which the projected hash value of a word depends on the witness, and there seems to be no trapdoor enabling to compute the hash value K from only $c = \mathsf{hp}$.

These two issues could be solved using either Trapdoor SPHF [BBC+13] or NIZK of knowledge of hk. But both methods require pairings or random oracle, if instantiated on cyclic or bilinear groups. Instead we construct it as follows:

First, suppose that a projection key is well-formed (i.e., there exists a corresponding hashing key). Then, there exists an *unbounded* zero-knowledge simulator that "extracts" a corresponding hashing key and computes the hash value. To boost this into full-fledged zero knowledge with an efficient simulator, we rely on the "OR trick" from [FLS90]. We add a random 4-tuple (g', h', u', e') to the CRS, and build an SPHF for the augmented language $C \in \mathscr{L}$ or (g', h', u', e') is a DDH tuple. In the normal setup, (g', h', u', e') is not a DDH tuple with overwhelming probability, so the soundness property is preserved. In the trapdoor setup, $(g', h', u', e') := (g', h', g'^r, h'^r)$ is a random DDH tuple, and the zero-knowledge simulator uses the witness r to compute the hash value.

Second, to ensure that the projection key is well-formed, we use a second SPHF. The idea for building the second SPHF is as follows: in most SPHF schemes, proving that a projected key hp is valid corresponds to proving that it lies in the column span of some matrix Γ (where all of the linear algebra is carried out in the exponent). Now pick a random vector tk: if hp lies in the span of Γ, then $\mathsf{hp}^\intercal\mathsf{tk}$ is completely determined given $\Gamma^\intercal\mathsf{tk}$; otherwise, it is completely random. The former yields the projective property and the latter yields smoothness, for the SPHF with hashing key hk and projection key $\mathsf{tp} = \Gamma^\intercal\mathsf{tk}$. Since the second SPHF is built using the transpose $\Gamma^\intercal$ of the original matrix Γ (defining the language $\mathscr{L}$), we refer to it as a "transpose SPHF". As it turns out, the second fix could ruin soundness of the ensuing iZK protocol: a cheating prover could pick a malformed $\Gamma^\intercal\mathsf{tk}$, and then the hash value $\mathsf{hp}^\intercal\mathsf{tk}$ computed by the verifier could leak additional information about his witness hk for hp, thereby ruining smoothness. To protect against the leakage, we would inject additional randomness into hk so that smoothness holds even in the presence of leakage from the hash value $\mathsf{hp}^\intercal\mathsf{tk}$. This idea is inspired by the 2-universality technique introduced in a very different context of chosen-ciphertext security [CS02].

Finally, to get simulation-soundness (i.e., soundness even if the adversary can see fake or simulated proofs), we rely on an additional "OR trick" (mixed up with an idea of Malkin et al. [MTVY11]): we build an SPHF for the augmented language $C \in \mathscr{L}$, or (g', h', u', e') is a DDH tuple (as before), or $(g'', h'', \mathcal{W}_1(C), \mathcal{W}_2(C))$ is not a DDH tuple (with $\mathcal{W}_k$ a Waters function [Wat05], $\mathcal{W}_k(m) = v_{k,0} \prod_{i=1}^{|m|} v_{k,i}^{m_i}$, when $m = m_1 \| \dots \| m_{|m|}$ is a bitstring, the $v_{k,0}, \dots, v_{k,|m|}$ are random group elements, and C is seen as a bitstring, for $k = 1, 2$).

In the security proof, with non-negligible probability, $(g'', h'', \mathcal{W}_1(C), \mathcal{W}_2(C))$ is a non-DDH tuple for simulated proofs, and a DDH tuple for the soundness challenge, which proves simulation-soundness.

Organization. First, we formally introduce the notion of *implicit zero-knowledge proofs* (iZK) in Section 4.2. Second, in Section 4.3, we discuss some difficulties related to the construction of iZK from SPHF and provide an intuition of our method to overcome these difficulties. Next, we show how to construct iZK and SSiZK from SPHF over cyclic groups for any language handled by the generic framework [BBC+13], which encompasses most, if not all, known SPHFs over cyclic groups. Then in Section 4.3.8, we extend our construction of iZK from SPHF to handle larger classes of languages described by computational structures such as circuits or branching programs. Third, in Section 4.4, we indeed show a concrete application of our iZK constructions: the most efficient 3-round two-party protocol computing inner product in the UC framework with static corruption so far. We analyze our construction and provide a detailed comparison with the Groth-Sahai methodology [GS08] and the approach based on zero-knowledge proofs "à la Schnorr" [Sch90] in Section 4.4.3. In addition, as proof of concept, we show in Section 4.4.1 that iZK can be used instead of ZK arguments to generically convert any protocol secure in the semi-honest model into a protocol secure in the malicious model. This conversion follows the generic transformation of Goldreich, Micali and Wigderson (GMW) in their seminal papers [GMW87b; GMW87a]. While applying directly the original transformation with Schnorr-like ZK protocols blows up the number of rounds by a multiplicative factor of at least three (even in the common reference string model), our conversion only adds a small constant number of rounds.

4.1.6 Related Work

SPHFs were introduced by Cramer and Shoup in [CS02] in order to achieve IND-CCA security from IND-CPA encryption schemes, which led to the first efficient IND-CCA encryption scheme provably secure in the standard model under the DDH assumption [CS98]. They can intuitively be seen as a kind of implicit designated-verifier proofs of membership to some language [ACP09; BPV12]. The connection between zero-knowledge protocols and SPHF was uncovered in [GL03; GL06] with password-authenticated key exchange protocols, in [BPV12] with blind signatures, and in [ABB+13] with oblivious transfer.

Using the "OR trick" with SPHF is reminiscent of [ABP15]. However, the methods used in this are very different from the one in [ABP15], as we do not use pairings, but consider weaker form of SPHF on the other hand.

A recent line of work has focused on the cut-and-choose approach for transforming security from semi-honest to malicious models [IKLP06; LP07; LP11; sS11; sS13; Lin13; HKE13] as an alternative to the use of zero-knowledge arguments. Indeed, substantial progress has been made towards practical protocols via this approach, as applied to Yao's garbled circuits. However, the state-of-the-art still incurs a large computation and communication multiplicative overhead that is equal to the security parameter. We note that Yao's garbled circuits do not efficiently generalize to arithmetic computations, and that our approach would yield better concrete efficiency for natural functions F that admit compact representations by arithmetic branching programs. In particular, Yao's garbled circuits cannot take advantage of the structure in languages handled by the Groth-Sahai methodology [GS08], and namely the ones defined by multi-exponentiations: even in the latter case, Groth-Sahai technique

requires pairings, while we will be able to avoid them.

The idea of using implicit proofs (without the zero-knowledge requirement) as a lightweight alternative to zero-knowledge proofs also appeared in an earlier work of Aiello, Ishai and Reingold [AIR01], where it was already observed that they could be used to build round-efficient protocols. They realize implicit proofs using conditional disclosure of secrets [GIKM98]. The latter, together with witness encryption [GGSW13] and SPHFs, only provide a weak "honest-verifier zero-knowledge" guarantee.

In a completely different context, a primitive called zero-knowledge to garbled circuits has been introduced in [CGOS07]. The latter is essentially an implicit zero-knowledge proof (in the plain model). However, it was built using very general tools (and relies on expensive Karp reductions to NP-complete problems), without computational efficiency in mind, to show the theoretical feasibility of a strong form of secure computation called covert multiparty computation. It was observed later in [Jar16b] that iZK can also be used to achieve this security notion efficiently. Very recently, zero-knowledge to garbled circuits have been used to obtain new results (in the plain model) regarding the round-efficiency of general multiparty computation [ACJ17].

Recently, Jarecki introduced the concept of conditional key encapsulation mechanism [Jar14], which is related to iZK as it adds a "zero-knowledge flavor" to SPHFs by allowing witness extraction. The construction is a combination of SPHF and zero-knowledge proofs "à la Schnorr". Contrary to iZK, it does not aim at reducing the interactivity of the resulting protocol, but ensures its covertness.

Witness encryption was introduced by Garg et al. in [GGSW13]. It enables to encrypt a message M for a word C and a language $\mathscr{L}$ into a ciphertext c, so that any user knowing a witness w that $C \in \mathscr{L}$ can decrypt c. Similarly to SPHFs, witness encryption also only has this "honest-verifier zero-knowledge" flavor: it does not enable to decrypt ciphertext for words $C \notin \mathscr{L}$, with a trapdoor. That is why, as SPHF, witness encryption cannot be used to construct directly iZK.

4.2 Definition of Implicit Zero-Knowledge Arguments

4.2.1 Definition

Let $(\mathsf{i}\mathscr{L}_{\mathsf{crs}})_{\mathsf{crs}}$ be a family of NP languages, indexed by a common reference string crs, and defined by a (polynomial time) witness relation $\mathsf{i}\mathcal{R}_{\mathsf{crs}}$, namely $\mathsf{i}\mathscr{L} = \{\mathsf{x} \in \mathsf{i}\mathcal{X}_{\mathsf{crs}} \mid \exists \mathsf{iw},\ \mathsf{i}\mathcal{R}_{\mathsf{crs}}(\mathsf{x}, \mathsf{iw}) = 1\}$, where $(\mathsf{i}\mathcal{X}_{\mathsf{crs}})_{\mathsf{crs}}$ is a family of sets. The reference string crs is generated by some polynomial-time algorithm $\mathsf{Setup}_{\mathsf{crs}}$ taking as input the unary representation of the security parameter κ. For the sake of simplicity, crs is often implicit.

To achieve stronger properties (namely simulation-soundness in Section 4.3.5), we sometimes also assume that $\mathsf{Setup}_{\mathsf{crs}}$ can output some additional information or trapdoor $\mathcal{T}_{\mathsf{crs}}$. This trapdoor should enable to check, in polynomial time, whether a given word x is in $\mathsf{i}\mathscr{L}$ or not. It is only used in security proofs, and is never used by the iZK algorithms.

An iZK is defined by the following polynomial-time algorithms:

- $\mathsf{icrs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{crs})$ generates the (normal) common reference string (CRS) icrs (which implicitly contains crs). The resulting CRS provides statistical soundness;

- $(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs})$[4] generates the (trapdoor) common reference string icrs together with a trapdoor $\mathsf{i}\mathcal{T}$. The resulting CRS provides statistical zero-knowledge;
- $(\mathsf{ipk}, \mathsf{isk}) \xleftarrow{\$} \mathsf{iKG}^{\ell}(\mathsf{icrs}, \mathsf{x}, \mathsf{iw})$ generates a public/secret key pair, associated to a word $\mathsf{x} \in \mathsf{i}\mathscr{L}$ and a label $\ell \in \{0,1\}^*$, with witness iw;
- $(\mathsf{ipk}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}^{\ell}(\mathsf{icrs}, \mathsf{i}\mathcal{T}, \mathsf{x})$ generates a public/trapdoor key pair, associated to a word $\mathsf{x} \in \mathfrak{X}$ and a label $\ell \in \{0,1\}^*$;
- $(c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{icrs}, \mathsf{ipk}, \mathsf{x})$ outputs a ciphertext c of a value K (an ephemeral key), for the public key ipk, the word x, and the label $\ell \in \{0,1\}^*$;
- $K \leftarrow \mathsf{iDec}^{\ell}(\mathsf{icrs}, \mathsf{isk}, c)$ decrypts the ciphertext c for the label $\ell \in \{0,1\}^*$, and outputs the ephemeral key K;
- $K \leftarrow \mathsf{iTDec}^{\ell}(\mathsf{icrs}, \mathsf{itk}, c)$ decrypts the ciphertext c for the label $\ell \in \{0,1\}^*$, and outputs the ephemeral key K.

The three last algorithms can be seen as key encapsulation and decapsulation algorithms. Labels ℓ are only used for SSiZK and are often omitted. The CRS icrs is often omitted, for the sake of simplicity.

Normally, the algorithms iKG and iDec are used by the user who wants to (implicitly) prove that some word x is in $\mathsf{i}\mathscr{L}$ (and we often call this user the prover), while the algorithm iEnc is used by the user who wants to (implicitly) verify this (and we often call this user the verifier), as shown in Figures 4.1 and 4.3. The algorithms iTKG and iTDec are usually only used in proofs, to generate simulated or fake implicit proofs (for the zero-knowledge property).

4.2.2 Security Requirements

An iZK satisfies the four following properties (for any $(\mathsf{crs}, \mathcal{T}_{\mathsf{crs}}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{crs}}(1^{\kappa})$):

- **Correctness.** The encryption is the reverse operation of the decryption, with or without a trapdoor: for any $\mathsf{icrs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{crs})$ or with a trapdoor, for any $(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs})$, and for any $\mathsf{x} \in \mathfrak{X}$ and any $\ell \in \{0,1\}^*$,
  - if $\mathsf{x} \in \mathsf{i}\mathscr{L}$ with witness iw, $(\mathsf{ipk}, \mathsf{isk}) \xleftarrow{\$} \mathsf{iKG}^{\ell}(\mathsf{icrs}, \mathsf{x}, \mathsf{iw})$, and $(c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{ipk}, \mathsf{x})$, then we have $K = \mathsf{iDec}^{\ell}(\mathsf{isk}, c)$;
  - if $(\mathsf{ipk}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}^{\ell}(\mathsf{i}\mathcal{T}, \mathsf{x})$ and $(c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{ipk}, \mathsf{x})$, then we have $K = \mathsf{iTDec}^{\ell}(\mathsf{itk}, c)$.
- **Setup Indistinguishability.** A polynomial-time adversary cannot distinguish a normal CRS generated by iSetup from a trapdoor CRS generated by $\mathsf{iTSetup}$. More formally, no PPT can distinguish, with non-negligible advantage, the two distributions:

$$\{\mathsf{icrs} \mid \mathsf{icrs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{crs})\} \qquad \{\mathsf{icrs} \mid (\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs})\}.$$

[4] When the CRS is word-dependent, i.e., when the trapdoor $\mathsf{i}\mathcal{T}$ does only work for one word x^* previously chosen, there is a second argument: $(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs}, \mathsf{x}^*)$. Security notions are then slightly different. See details in Section 4.3.7.2.

$\mathsf{Exp}^{\mathtt{iZK\text{-}zk}\text{-}b}(\mathscr{A}, \mathsf{crs}, \kappa)$
$(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs})$
$(\ell, \mathsf{x}^*, \mathsf{iw}, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{icrs}, \mathsf{i}\mathcal{T})$
if $\mathsf{i}\mathcal{R}(\mathsf{x}^*, \mathsf{iw}) = 0$ **then return** random bit
if $b = 0$ **then** $(\mathsf{ipk}, \mathsf{isk}) \xleftarrow{\$} \mathsf{iKG}^{\ell}(\mathsf{icrs}, \mathsf{x}^*, \mathsf{iw}^*)$
else $(\mathsf{ipk}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}^{\ell}(\mathsf{i}\mathcal{T}, \mathsf{x}^*)$
$(c, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{st}, \mathsf{icrs}, \mathsf{i}\mathcal{T}, \mathsf{ipk})$
if $b = 0$ **then** $K \leftarrow \mathsf{iDec}^{\ell}(\mathsf{isk}, c)$
else $K \leftarrow \mathsf{iTDec}^{\ell}(\mathsf{itk}, c)$
return $\mathscr{A}(\mathsf{st}, K)$

$\mathsf{Exp}^{\mathtt{iZK\text{-}ss}\text{-}b}(\mathscr{A}, \mathsf{crs}, \kappa)$
$(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs})$
$(\ell^*, \mathsf{x}^*, \mathsf{ipk}^*, \mathsf{st}) \xleftarrow{\$} \mathscr{A}^{\mathcal{O}}(\mathsf{icrs})$
$(c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{ipk}^*, \mathsf{x}^*)$
if $b = 0$ **then** $K' \leftarrow K$
else $K' \xleftarrow{\$} \Pi$
$b' \xleftarrow{\$} \mathscr{A}^{\mathcal{O}}(\mathsf{st}, c, K')$
if $\exists \mathsf{itk}$, $(\ell^*, \mathsf{x}^*, \mathsf{ipk}^*, \mathsf{itk}) \in L \cup L'$ **then**
return random bit
if $\mathsf{x}^* \in \mathsf{i}\mathscr{L}$ **then return** random bit
return b'

Figure 4.2: Experiments $\mathsf{Exp}^{\mathtt{iZK\text{-}zk}\text{-}b}$ for zero-knowledge of iZK, and $\mathsf{Exp}^{\mathtt{iZK\text{-}ss}\text{-}b}$ for simulation-soundness of SSiZK

- **Soundness.** When the CRS is generated as $\mathsf{icrs} \xleftarrow{\$} \mathsf{iSetup}(\mathsf{crs})$, and when $\mathsf{x} \notin \mathscr{L}$, the distribution of K is statistically indistinguishable from the uniform distribution, even given c. More formally, if Π is the set of all the possible values of K, for any bitstring ipk, for any word $\mathsf{x} \notin \mathsf{i}\mathscr{L}$, for any label $\ell \in \{0,1\}^*$, the two distributions:

$$\{(c, K) \mid (c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{ipk}, \mathsf{x})\} \qquad \{(c, K') \mid (c, K) \xleftarrow{\$} \mathsf{iEnc}^{\ell}(\mathsf{ipk}, \mathsf{x}); K' \xleftarrow{\$} \Pi\}$$

  are statistically indistinguishable (iEnc may output $(\perp, K)$ when the public key ipk is not well formed).

- **Zero-Knowledge.** For any label $\ell \in \{0,1\}^*$, when the CRS is generated using $(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}^{\ell}(\mathsf{crs})$, for any message $\mathsf{x}^* \in \mathsf{i}\mathscr{L}$ with the witness iw^*, the public key ipk and the decapsulated key K corresponding to a ciphertext c chosen by the adversary, either using isk or the trapdoor itk, should be indistinguishable, even given the trapdoor $\mathsf{i}\mathcal{T}$. More formally, we consider the experiment $\mathsf{Exp}^{\mathtt{iZK\text{-}zk}\text{-}b}$ in Figure 4.2. The iZK is (statistically) zero-knowledge if the advantage of any adversary $\mathscr{A}$ (not necessarily polynomial-time) for this experiment is negligible.

We defined our security notion with a "composable" security flavor, as Groth and Sahai in [GS08]: soundness and zero-knowledge are statistical properties, the only computational property is the setup indistinguishability property. This is slightly stronger than what is needed, but is satisfied by our constructions and often easier to use.

We also consider stronger iZK, called simulation-sound iZK or SSiZK, which satisfies the following additional property:

- **Simulation Soundness.** The soundness holds (computationally) even when the adversary can see simulated public keys and decryption with these keys. More formally, we consider the experiment $\mathsf{Exp}^{\mathtt{iZK\text{-}ss}\text{-}b}$ in Figure 4.2, where the oracle $\mathcal{O}$, and the lists L and L' are defined as follows:
  - on input (ℓ, x), $\mathcal{O}$ generates $(\mathsf{ipk}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}(\mathsf{icrs}, \mathsf{i}\mathcal{T}, \mathsf{x})$, stores $(\ell, \mathsf{x}, \mathsf{ipk}, \mathsf{itk})$ in a list L, and outputs ipk;

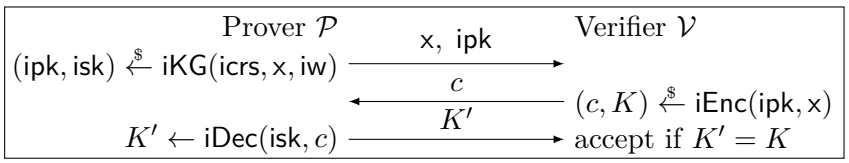


Figure 4.3: Three-round zero-knowledge from iZK for a word $\mathsf{x} \in \mathsf{i}\mathscr{L}$ and a witness iw

– on input (ipk, c), $\mathcal{O}$ retrieves the record $(\ell, \mathsf{x}, \mathsf{ipk}, \mathsf{itk})$ from L (and aborts if no such record exists), removes it from L, and adds it to L', computes $K \leftarrow \mathsf{iTDec}^{\ell}(\mathsf{icrs}, \mathsf{itk}, c)$, and outputs K.

The iZK is (statistically) simulation-sound if the advantage of any adversary $\mathscr{A}$ (not necessarily polynomial-time) for this experiment is negligible.

Remark 4.2.1. *An iZK for some language $\mathsf{i}\mathscr{L}$ directly leads to a 3-round zero-knowledge arguments for $\mathsf{i}\mathscr{L}$. The construction is depicted in Figure 4.3 and the proof is provided in Section 4.2.2. If the iZK is additionally simulation-sound, the resulting zero-knowledge argument is also simulation-sound.*

Proof of Remark 4.2.1 (Sketch). We prove that the construction of ZK from iZK given in Remark 4.2.1 is correct. The completeness and the soundness of the zero-knowledge protocol from iZK directly follows from the completeness and the soundness of the underlying iZK; the zero-knowledge is straightforward too: the existence of a simulator is ensured because a simulator is explicitly given by the underlying iZK. The simulator simply uses the trapdoor instead of the witness, and the proof of perfect simulation directly follows from the zero-knowledge property of the underlying iZK.

4.3 Construction of Implicit Zero-Knowledge Arguments

In this section, we construct implicit zero-knowledge arguments, building on the generic framework of SPHFs (introducted in [BBC+13] for the particular case of cyclic groups, and generalized in Benhamouda's thesis [Ben16]), where the projection key hp can depend on the word, as it is at the core of our construction of iZK. The framework is recalled and illustrated in Section 2.3.4.1. First, we explain in more details the limitations of SPHFs and the fact they cannot directly be used to construct iZK (we actually exhibit a concrete attack). Second, we show how to overcome these limitations to build iZK and SSiZK.

4.3.1 Limitations of Smooth Projective Hash Functions

At a first glance, as explained in the introduction, it may look possible to construct an iZK from an SPHF for the same language $\mathscr{L} = \mathsf{i}\mathscr{L}$ as follows:

- $\mathsf{iSetup}(\mathsf{crs})$ and $\mathsf{iTSetup}(\mathsf{crs})$ outputs the empty CRS $\mathsf{icrs} := \perp$;
- $\mathsf{iKG}(\mathsf{icrs}, \mathsf{x}, \mathsf{iw})$ outputs an empty public key $\mathsf{ipk} := \perp$ together with the secret key $\mathsf{isk} := (\mathsf{x}, \mathsf{iw})$;

- $\mathsf{iEnc}(\mathsf{ipk}, \mathsf{x})$ generates a random hashing key $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{crs}, \mathsf{x})$ and outputs the ciphertext $c := \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{crs}, \mathsf{x})$ together with the ephemeral key $K := H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{crs}, \mathsf{x})$;
- $\mathsf{iDec}(\mathsf{isk}, c)$ outputs the ephemeral key $K := \mathsf{proj}H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{crs}, \mathsf{x}, \mathsf{iw})$.

This construction is sound: if $\mathsf{x} \notin \mathscr{L}$, given only $c = \mathsf{hp}$, the smoothness ensures that $K = H$ looks random. Unfortunately, there seems to be no way to compute K from only c, or in other words, there does not seem to exist algorithms iTKG and iTDec.

Example 2.3.19 is not Zero-Knowledge. Actually, with the SPHF from Example 2.3.19, no such algorithm iTKG or iTDec (verifying the zero-knowledge property) exists. It is even worse than that: a malicious verifier may get information about the witness, even if he just has a feedback whether the prover could use the correct hash value or not (and get the masked value or not), in a protocol such as the one in Figure 4.1. A malicious verifier can indeed generate a ciphertext $c = \mathsf{hp}$, by generating hp_1 honestly but by picking hp_2 and hp_3 uniformly at random. Now, a honest prover will compute $\mathsf{proj}H = \mathsf{hp}_1^r \mathsf{hp}_2^b \mathsf{hp}_3^{-rb}$, to get back the ephemeral key (using iDec). When C is an encryption of $b = 1$, this value is random and independent of H, as hp_2 and hp_3 have been chosen at random, while when $b = 0$, this value is the correct $\mathsf{proj}H$ and is equal to H. Thus the projected hash value $\mathsf{proj}H$, which is the ephemeral output key by the honest prover, reveals some information about b, part of the witness.

If we want to avoid such an attack, the prover has to make sure that the hp he received was built correctly. Intuitively, this sounds exactly like the kind of verifications we could make with an SPHF: we could simply build an SPHF on the language of the "correctly built" hp. Then the prover could send a projection key for this new SPHF and ask the verifier to XOR the original hash value H with the hash value of this new SPHF. However, things are not that easy: first this does not solve the limitation due to the security proof (the impossibility of computing H for $\mathsf{x} \notin \mathsf{i}\mathscr{L}$) and second, in the SPHF in Example 2.3.19, all projection keys are valid (since Γ is full-rank, for any hp, there exists necessarily a hk such that $\mathsf{hp} = \Gamma \bullet \mathsf{hk}$).

4.3.2 iZK Construction

Let us consider an SPHF defined as in Section 2.3.4.1 for a language $\mathsf{i}\mathscr{L} = \mathscr{L}$. In this section, we show how to design, step by step, an iZK for $\mathsf{i}\mathscr{L}$ from this SPHF, following the overview in Section 4.1.5. At the end, we provide a summary of the construction and a complete proof. We illustrate our construction on the language of ElGamal ciphertexts of bits (Examples 2.3.17 and 2.3.19), and refer to this language as "our example". We suppose a cyclic group $\mathbb{G}$ of prime order p is fixed, and that DDH is hard in $\mathbb{G}$[5].

We have seen the limitations of directly using the original SPHF are actually twofold. First, SPHFs do not provide a way to compute the hash value of a word outside the language, with just a projection key for which the hashing key is not known. Second, nothing ensures that a projection key has really been derived from an actually known hashing key, and in such a bad case, the projected hash value may leak some information about the word C (and the witness).

To better explain our construction, we first show how to overcome the first limitation. Thereafter, we will show how our approach additionally allows to check the validity of the

[5]The construction can be trivially extended to DLIN, or any MDDH assumption [EHK+13] though.

projection keys (with a non-trivial validity meaning). It will indeed be quite important to notice that the projection keys coming from our construction (according to one of the setups) will not necessarily be valid (with a corresponding hashing key), as the corresponding matrix Γ will not always be full rank, contrary to the projection keys of the SPHF in Example 2.3.19. Hence, the language of the valid projection keys will make sense in this setting.

Adding the Trapdoor. The CRS of our construction is a tuple $\mathsf{icrs} = (g', h', u' = g'^{r'}, e' = h'^{s'}) \in \mathbb{G}^4$, with g', h' two random generators of $\mathbb{G}$, and

- r', s' two random distinct scalars in $\mathbb{Z}_p$, for the normal CRS generated by iSetup, so that (g', h', u', e') is not a DDH tuple;
- $r' = s'$ a random scalar in $\mathbb{Z}_p$, for the trapdoor CRS generated by iTSetup, with $\mathsf{i}\mathcal{T} = r'$ the trapdoor, so that (g', h', u', e') is a DDH tuple.

Then, we build an SPHF for the augmented language $\mathscr{L}_t$ defined as follows: a word $C_t = (C, u', e')$ is in $\mathscr{L}_t$ if and only if either C is in the original language $\mathscr{L}$ or (u', e') is a DDH tuple. This new language $\mathscr{L}_t$ can be seen as the disjunction of the original language $\mathscr{L}$ and of the DDH language in basis (g', h'). Construction of disjunctions of SPHFs were proposed in [ABP15] but require pairings. In this thesis, we use an alternative more efficient construction without pairing[6]. Let us show it on our example, with $C_t = (C, u', e')$. We set $\vec{\hat{C}}_t := (g'^{-1}, 1, 1, 1, 1, 1, 1)$ and $\Gamma_t(C_t) \in \mathbb{G}^{(k+3)\times(n+3)}$ as

$$\Gamma_t(C_t) := \left(\begin{array}{ccc|c} \multicolumn{3}{c|}{1} & \Gamma(C) \\ \hline g' & 1 & 1 & \vec{\hat{C}} = \theta(C) \\ \hline 1 & g' & h' & 1 \ \ldots \ 1 \\ \hline g' & u' & e' & 1 \ \ldots \ 1 \end{array}\right) = \left(\begin{array}{ccc|cccc} 1 & 1 & 1 & g & h & 1 & 1 \\ 1 & 1 & 1 & 1 & g & u & e/g \\ 1 & 1 & 1 & 1 & 1 & g & h \\ \hline g' & 1 & 1 & u & e & 1 & 1 \\ \hline 1 & g' & h' & 1 & 1 & 1 & 1 \\ \hline g' & u' & e' & 1 & 1 & 1 & 1 \end{array}\right). \tag{4.1}$$

Let us show the language corresponding to Γ_t and $\vec{\hat{C}}_t$ is indeed $\mathscr{L}_t$: Due to the first column of Γ_t and the first element of $\vec{\hat{C}}_t$, if $\vec{\hat{C}}_t$ is a linear combination of rows of Γ_t with coefficients $\vec{\lambda_t}$ (i.e., $\vec{\hat{C}}_t = \vec{\lambda_t} \bullet \Gamma_t$), one has $\lambda_{t,4} + \lambda_{t,6} = -1$, and thus at least $\lambda_{t,4}$ or $\lambda_{t,6}$ is not equal to zero.

- If $\lambda_{t,6} \neq 0$, looking at the second and the third columns of Γ_t gives that:

  $$\lambda_{t,5} \bullet (g',\ h') + \lambda_{t,6} \bullet (u',\ e') = (1, 1) \quad \text{equivalent to} \quad (u', e') = (g'^{\lambda_{t,5}/\lambda_{t,6}},\ h'^{\lambda_{t,5}/\lambda_{t,6}}),$$

  or in other words (u', e') is a DDH tuple in basis (g', h');
- if $\lambda_{t,4} \neq 0$, looking at the last four columns of Γ_t gives that: $\lambda_{t,4} \bullet \vec{\hat{C}} = \lambda_{t,4} \bullet (u, e, 1, 1)$ is a linear combination of rows of Γ, hence $\vec{\hat{C}}$ too. As a consequence, by definition of $\mathscr{L}$, $C \in \mathscr{L}$.

[6] Contrary to [ABP15] however, our matrix Γ_t depends on the words C_t, which is why we get this more efficient construction.

Now, whatever the way the CRS is generated (whether (u', e') is a DDH tuple or not), it is always possible to compute $\mathsf{proj}H$ as follows, for a word $C \in \mathscr{L}$ with witnesses r and b:

$$\mathsf{proj}H = \overrightarrow{\lambda_t} \bullet \mathsf{hp} \qquad \overrightarrow{\lambda_t} = (\overrightarrow{\lambda}, -1, 0, 0) = (r, b, -rb, -1, 0, 0)$$

When the CRS is generated with the normal setup, as shown above, this is actually the only way to compute $\mathsf{proj}H$, since (u', e') is not a DDH tuple and so $\overrightarrow{C}_t$ is linearly dependent of the rows of Γ_t if and only if $C \in \mathscr{L}$. On the opposite, when the CRS is generated by the trapdoor setup with trapdoor r', we can also compute $\mathsf{proj}H$ using the witness r': $\mathsf{proj}H = \overrightarrow{\lambda'_t} \bullet \mathsf{hp}$ with $\overrightarrow{\lambda'_t} = (0, 0, 0, 0, r', -1)$.

However, the latter way to compute $\mathsf{proj}H$ gives the same result as the former way, only if $\mathsf{hp}_{t,5}$ and $\mathsf{hp}_{t,6}$ involve the correct value for hk_1. A malicious verifier could decide to choose random $\mathsf{hp}_{t,5}$ and $\mathsf{hp}_{t,6}$, which would make $\overrightarrow{\lambda'_t} \bullet \mathsf{hp}$ look random and independent of the real hash value!

Ensuring the Validity of Projection Keys. The above construction and trapdoor would provide zero-knowledge if we could ensure that the projection keys hp (generated by a potentially malicious verifier) is valid, so that, intuitively, $\mathsf{hp}_{t,5}$ and $\mathsf{hp}_{t,6}$ involve the correct value of hk_1. Using a zero-knowledge proof (that hp derives from some hashing key hk) for that purpose would annihilate all our efforts to avoid adding rounds and to work under plain DDH (interactive ZK proofs introduce more rounds, and Groth-Sahai [GS08] NIZK would require assumptions on bilinear groups). So we are left with doing the validity check again with SPHFs.

Fortunately, the language of valid projection keys hp can be handled by the generic framework, since a valid projection key hp is such that: $\mathsf{hp} = \Gamma_t \bullet \mathsf{hk}$, or in other words, if we transpose everything $\mathsf{hp}^\intercal = \mathsf{hk}^\intercal \bullet \Gamma_t^\intercal$. This is exactly the same as in Equation (2.1), with $\overrightarrow{C} \leftrightarrow \mathsf{hp}^\intercal$, $\Gamma \leftrightarrow \Gamma_t^\intercal$ and witness $\overrightarrow{\lambda} \leftrightarrow \mathsf{hk}^\intercal$. So we can now define a smooth projective hash function on that language, where the projection key is called transposed projection key tp, the hashing key is called transposed hashing key tk, the hash value is called transposed hash value $\mathsf{t}H$ and the projected hash value is called transposed projected hash value $\mathsf{tproj}H$.

Finally, we could define an iZK, similarly to the one in Section 4.3.1, except, ipk contains a transposed projection key tp (generated by the prover from a random transposed hashing key tk), and c contains the associated transposed projected hash value $\mathsf{tproj}H$ in addition to hp, so that the prover can check using tk that hp is valid by verifying whether $\mathsf{tproj}H = \mathsf{t}H$ or not.

An Additional Step. Unfortunately, we are not done yet, as the above modification breaks the soundness property! Indeed, in this last construction, the prover now learns an additional information about the hash value H: $\mathsf{tproj}H = \mathsf{hk}^\intercal\mathsf{tp}$, which does depend on the secret key hk. He could therefore choose $\mathsf{tp} = \overrightarrow{C}_t^\intercal$, so that $\mathsf{tproj}H = \mathsf{hk}^\intercal\overrightarrow{C}_t^\intercal = \overrightarrow{C}_t\mathsf{hk}$ is the hash value $H = K$ of C under hk.

We can fix this by ensuring that the prover will not know the extended word $\overrightarrow{C}_t$ on which the SPHF will be based when he sends tp, by making $\overrightarrow{C}_t$ and K depend on a random scalar $\zeta \in \mathbb{Z}_p$ chosen by the verifier (and included in c).

Detailed Construction. Let us now formally show how to build an iZK from any SPHF built from the generic framework of [BBC+13], following the previous ideas. We recall that we consider a language $\mathscr{L} = \mathsf{i}\mathscr{L}$, such that a word $\mathsf{x} = C$ is in $\mathsf{i}\mathscr{L}$, if and only if $\overrightarrow{C} = \theta(C)$ is

a linear combination of the rows of some matrix $\Gamma \in \mathbb{G}^{k\times n}$ (which may depend on C). The coefficients of this linear combination are entries of a row vector $\vec{\lambda} \in \mathbb{Z}_p^{1\times k}$: $\vec{C} = \vec{\lambda} \bullet \Gamma$, where $\vec{\lambda} = \vec{\lambda}(\mathsf{iw})$ can be computed from the witness iw for x.

The setup algorithms $\mathsf{iSetup}(\mathsf{crs})$ and $\mathsf{iTSetup}(\mathsf{crs})$ are defined as above (page 71). We define an extended language using the generic framework:

$$\begin{aligned} \theta_t(\mathsf{x},\zeta) = \vec{\hat{C}}_t = (g'^{-1}, 1, \ldots, 1) &\qquad \in \mathbb{G}^{1\times(n+3)} \\ \Gamma_t(\mathsf{x}) &\qquad \in \mathbb{G}^{(k+3)\times(n+3)}, \end{aligned}$$

where $\Gamma_t(\mathsf{x})$ is the matrix of Equation (4.1), and ζ is a random scalar used to ensure the prover cannot guess the word $\vec{\hat{C}}_t$ which will be used, and so cannot choose $\mathsf{tp} = \vec{\hat{C}}_t$. We write:

$$\begin{aligned} \vec{\lambda_t}(\mathsf{iw}) &= (\vec{\lambda}(\mathsf{iw}), -1, 0, 0) \\ \vec{\lambda_t}(\mathsf{i}\mathcal{T}) &= (0, \ldots, 0, r', -1, 0, \ldots, 0) \qquad \text{with } \mathsf{i}\mathcal{T} = r', \end{aligned}$$

so that:

$$\vec{\hat{C}}_t = \begin{cases} \vec{\lambda_t}(\mathsf{iw}) \bullet \Gamma_t(\mathsf{x}) & \text{if } (g', h', u', e') \text{ is a } \mathsf{DDH} \text{ tuple, with witness } \mathsf{iw} \\ \vec{\lambda_t}(\mathsf{i}\mathcal{T}) \bullet \Gamma_t(\mathsf{x}) & \text{if } \mathsf{x} \in \mathsf{i}\mathscr{L} \text{ with witness } \mathsf{i}\mathcal{T}. \end{cases}$$

The resulting iZK construction is depicted in Figure 4.4. This is a slightly more efficient construction than the one we sketched previously, where the prover does not test anymore explicitly $\mathsf{tproj}H$, but $\mathsf{tproj}H$ (or $\mathsf{t}H$) is used to mask K. Thus, $\mathsf{tproj}H$ no more needs to be included in c.

4.3.3 Notes

This construction was originally presented in [BCPW15]. The construction described in this section improves over the original construction by a factor 2, by avoiding the 2-universality method and directly using ζ to randomize K. For the sake of concreteness, the extended matrix Γ_t is explicitely defined. In fact, this extended matrix corresponds to the matrix for the disjunction of two languages, the original language and the language of DDH tuples. The construction can be generalized to handle disjunction with an arbitrary hard-subset-membership language, which allows to base it on a large variety of other assumptions. This generalized construction (also taking the factor two improvement into account) is presented in section 6.3.3.2 of our co-author's thesis [Ben16]. The security requirements are also slightly relaxed in [Ben16] to allow for computational soundness and computational zero-knowledge; here, we kept the original definition (soundness and zero-knowledge are statistical, only setup indistinguishability is computational) which is more restricted, but has a "composable" flavor comparable to the non-interactive zero-knowledge proofs of Groth and Sahai [GS08].

4.3.4 Proof of Security

Correctness. Straightforward.

Setup Indistinguishability. The only difference between iSetup and $\mathsf{iTSetup}$ is that in the former (g', h', u', e') is a random tuple, while in the later (g', h', u', e') is a DDH tuple. Hence the setup indistinguishability holds under plain DDH in $\mathbb{G}$.

$\underline{\mathsf{iSetup}(\mathsf{crs})}$	$\underline{\mathsf{iTSetup}(\mathsf{crs})}$
$(g', h') \xleftarrow{\$} \mathbb{G}^{*2}$ $(r', s') \xleftarrow{\$} \mathbb{Z}_p^2 \setminus \{(a, a) \mid a \in \mathbb{Z}_p\}$ $(u', e') \leftarrow (g'^{r'}, h'^{s'}) \in \mathbb{G}^2$ $\mathsf{icrs} \leftarrow (g', h', u', e')$ **return** icrs	$(g', h') \xleftarrow{\$} \mathbb{G}^{*2}$ $r' \xleftarrow{\$} \mathbb{Z}_p$ $(u', e') \leftarrow (g'^{r'}, h'^{r'}) \in \mathbb{G}^2$ $\mathsf{icrs} \leftarrow (g', h', u', e')$; $\mathsf{i}\mathcal{T} \leftarrow r'$ **return** $(\mathsf{icrs}, \mathsf{i}\mathcal{T})$
$\underline{\mathsf{iKG}(\mathsf{icrs}, \mathsf{x}, \mathsf{iw})}$	$\underline{\mathsf{iTKG}(\mathsf{icrs}, \mathsf{x}, \mathsf{i}\mathcal{T})}$
$\mathsf{tk} \xleftarrow{\$} \mathbb{Z}_p^{k+3}$ $\mathsf{ipk} := \mathsf{tp} \leftarrow \Gamma_t(\mathsf{x})^\intercal \bullet \mathsf{tk} \in \mathbb{G}^{n+3}$ $\mathsf{isk} := (\mathsf{x}, \mathsf{tk}, \mathsf{iw})$ **return** $(\mathsf{ipk}, \mathsf{isk})$	$\mathsf{tk} \xleftarrow{\$} \mathbb{Z}_p^{k+3}$ $\mathsf{ipk} := \mathsf{tp} \leftarrow \Gamma_t(\mathsf{x})^\intercal \bullet \mathsf{tk} \in \mathbb{G}^{n+3}$ $\mathsf{itk} := (\mathsf{x}, \mathsf{tk}, \mathsf{i}\mathcal{T})$ **return** $(\mathsf{ipk}, \mathsf{itk})$
$\underline{\mathsf{iEnc}(\mathsf{icrs}, \mathsf{ipk}, \mathsf{x})}$	$H \leftarrow \theta_t(\mathsf{x}) \bullet \mathsf{hk} \in \mathbb{Z}_p$ $K \leftarrow H^\zeta \cdot \mathsf{tproj}H \in \mathbb{G}$ $c := (\zeta, \mathsf{hp})$ **return** (K, c)
$\mathsf{tp} \leftarrow \mathsf{ipk}$; $\mathsf{hk} \xleftarrow{\$} \mathbb{Z}_p^{n+3}$; $\zeta \xleftarrow{\$} \mathbb{Z}_p$ $\mathsf{hp} \leftarrow \Gamma_t(\mathsf{x}) \bullet \mathsf{hk} \in \mathbb{Z}_p^{k+3}$ $\mathsf{tproj}H \leftarrow \mathsf{hk}^\intercal \bullet \mathsf{tp} \in \mathbb{G}$	
$\underline{\mathsf{iDec}(\mathsf{icrs}, \mathsf{isk}, c)}$	$\underline{\mathsf{iTDec}(\mathsf{icrs}, \mathsf{itk}, c)}$
$(\mathsf{x}, \mathsf{tk}, \mathsf{iw}) \leftarrow \mathsf{isk}$ $(\zeta, \mathsf{hp}) \leftarrow c$ $\mathsf{t}H \leftarrow \mathsf{hp}^\intercal \bullet \mathsf{tk} \in \mathbb{Z}_p$ $\mathsf{proj}H \leftarrow \overrightarrow{\lambda_t}(\mathsf{iw}) \bullet \mathsf{hp} \in \mathbb{G}$ **return** $K := \mathsf{proj}H^\zeta \cdot \mathsf{t}H \in \mathbb{G}$	$(\mathsf{x}, \mathsf{tk}, \mathsf{i}\mathcal{T}) \leftarrow \mathsf{itk}$ $(\zeta, \mathsf{hp}) \leftarrow c$ $\mathsf{t}H \leftarrow \mathsf{hp}^\intercal \bullet \mathsf{tk} \in \mathbb{Z}_p$ $\mathsf{trap}H := \overrightarrow{\lambda_t}(\mathsf{i}\mathcal{T}) \bullet \mathsf{hp} \in \mathbb{G}$ **return** $K := \mathsf{trap}H^\zeta \cdot \mathsf{t}H \in \mathbb{G}$

Figure 4.4: Construction of iZK

Statistical Soundness. Let us consider a CRS $\mathsf{icrs} = (\mathsf{crs}, g', h', u', e')$ generated by $\mathsf{iSetup}(\mathsf{crs})$. We need to show that, for any $C = \mathsf{x} \notin \mathscr{L} = \mathsf{i}\mathscr{L}$ (i.e., such that $\overrightarrow{C}$ is linearly independent of rows of Γ) and any $\mathsf{iZK} = \mathsf{tp} \in \mathbb{G}^{n+3}$, the distribution of $K = H^\zeta \cdot \mathsf{tproj}H$ is statistically close to uniform over $\mathbb{G}$, even given $c = (\zeta, \mathsf{hp})$.

As $\mathsf{x} \notin \mathscr{L}$ and (g', h', u', e') is not a DDH tuple, it holds that $\theta_t(\mathsf{x})$ does not belong to the span of the columns of $\Gamma_t(\mathsf{x})$. We start by showing that there is at most a single value ζ such that $(\zeta \bullet \theta_t(\mathsf{x})) \cdot \mathsf{ipk}^\intercal \in \mathsf{ColSpan}(\Gamma_t(\mathsf{x}))$. Let (ζ_1, ζ_2) be two values such that

$$(\zeta_1 \bullet \theta_t(\mathsf{x})) \cdot \mathsf{ipk}^\intercal \in \mathsf{ColSpan}(\Gamma_t(\mathsf{x})) \qquad (\zeta_2 \bullet \theta_t(\mathsf{x})) \cdot \mathsf{ipk}^\intercal \in \mathsf{ColSpan}(\Gamma_t(\mathsf{x}))$$

This implies $(\zeta_1 - \zeta_2) \bullet \theta_t(\mathsf{x}) \in \mathsf{ColSpan}(\Gamma_t(\mathsf{x}))$. As $\theta_t(\mathsf{x}) \notin \mathsf{ColSpan}(\Gamma_t(\mathsf{x}))$, this equation necessarily implies $\zeta_1 = \zeta_2$. Suppose now that ζ is chosen such that $(\zeta \bullet \theta_t(\mathsf{x})) \cdot \mathsf{ipk}^\intercal \notin \mathsf{ColSpan}(\Gamma_t(\mathsf{x}))$, which happens with overwhelming probability $1 - 1/p$. Then, observe that

$$K = H^\zeta \cdot \mathsf{tproj}H = (\theta_t(\mathsf{x}) \bullet \mathsf{hk})^\zeta \cdot (\mathsf{hk}^\intercal \bullet \mathsf{tp}) = \mathsf{hk}^\intercal \bullet ((\zeta \bullet \theta_t^\intercal(\mathsf{x})) \cdot \mathsf{tp}),$$

as $H = H^\intercal = \theta_t(\mathsf{x}) \bullet \mathsf{hk} = \mathsf{hk}^\intercal \bullet \theta_t^\intercal(\mathsf{x})$. Therefore, the value K can be seen as the hash value of $(\zeta \bullet \theta_t^\intercal(\mathsf{x})) \cdot \mathsf{tp}$, hence it is uniformly random from the view of the adversary.

Perfect Zero-Knowledge. Let $\mathsf{x}^* \in \mathsf{i}\mathscr{L} = \mathscr{L}$ be a word with witness iw^*. For the zero-knowledge property, we (the challenger playing the role of the prover) generates a public key $\mathsf{ipk} = \mathsf{tp}$, where tp is a projection key, associated to a random hashing key tk, for the language

of valid hp's. Then, the adversary (playing the role of the verifier) sends a ciphertext $c(\zeta, \mathsf{hp})$. There are two cases:

- either there exists $\mathsf{hk} \in \mathbb{Z}_p^{n+3}$ such that $\mathsf{hp} = \Gamma_t \bullet \mathsf{hk}$. In this case, we have

$$\begin{aligned}\mathsf{proj}H := \overrightarrow{\lambda_t}(\mathsf{iw}^*) \bullet \mathsf{hp} = \overrightarrow{\lambda_t}(\mathsf{iw}^*) \bullet \Gamma_t \bullet \mathsf{hk} = \overrightarrow{\hat{C}_t} \bullet \mathsf{hk} \\ = \overrightarrow{\lambda_t}(\mathsf{i}\mathcal{T}) \bullet \Gamma_t \bullet \mathsf{hk} = \overrightarrow{\lambda_t}(\mathsf{i}\mathcal{T}) \bullet \mathsf{hp} := \mathsf{trap}H\end{aligned}$$

  (this property actually can be seen as coming from the correctness of the SPHF with projection key hp);

- or, there does not exist $\mathsf{hk} \in \mathbb{Z}_p^{n+3}$ such that $\mathsf{hp} = \Gamma_t \bullet \mathsf{hk}$. In this case, hp is not valid and $\mathsf{t}H = \Gamma_t^\intercal \bullet \mathsf{tk}$ (with $\mathsf{tk} \in \mathbb{Z}_p^{k+3}$) looks uniformly random for the adversary (before he sees $\mathsf{proj}H^\zeta \cdot \mathsf{t}H$ or $\mathsf{trap}H^\zeta \cdot \mathsf{t}H$ in the game), since the only information he sees about tk is $\mathsf{tp} = \Gamma_t^\intercal \bullet \mathsf{tk}$, but hp is linearly independent of rows of $\Gamma_t^\intercal$. This property on $\mathsf{t}H$ can actually be seen as the smoothness property of the SPHF with projection key tp. Then $\mathsf{proj}H^\zeta \cdot \mathsf{t}H$ and $\mathsf{trap}H^\zeta \cdot \mathsf{t}H$ look both uniformly random to the adversary, and cannot be distinguished.

Therefore, our construction is perfect zero-knowledge.

4.3.5 SSiZK Construction

Our SSiZK construction is similar to our iZK construction, except that, in addition both iSetup and $\mathsf{iTSetup}$ add to the CRS icrs a tuple $(v_{k,i})_{i=0,\ldots,2\kappa}^{k=1,2}$ of group elements constructed as follows: for $i = 0$ to 2κ (with κ the security parameter): $r'_i \xleftarrow{\$} \mathbb{Z}_p, v_{1,i} \leftarrow g'^{r'_i}, v_{2,i} \leftarrow h'^{r'_i}$. We also define the two Waters functions [Wat05] $\mathcal{W}_k : \{0,1\}^{2\kappa} \to \mathbb{G}$, as $\mathcal{W}_k(m) = v_{k,0} \prod_{i=1}^{2\kappa} v_{k,i}^{m_i}$, for any bitstring $m = m_1 \| \ldots \| m_{2\kappa} \in \{0,1\}^{2\kappa}$. Finally, the CRS is also supposed to contain a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{2\kappa}$ drawn from a collision-resistant hash function family $\mathcal{HF}$.

Next, the language $\mathscr{L}_t$ is further extended by adding 3 rows and 2 columns (all equal to 1 except on the 3 new rows) to $\Gamma_t(\mathsf{x})$, where the 3 new rows are:

$$\left(\begin{array}{c|c|c|ccc|cc} 1 & 1 & 1 & 1 & \ldots & 1 & g' & h' \\ 1 & 1 & 1 & 1 & \ldots & 1 & u'' & e'' \\ \hline g' & 1 & 1 & 1 & \ldots & 1 & g' & 1 \end{array}\right) \in \mathbb{G}^{3\times(n+5)},$$

with $u'' = \mathcal{W}_1(\mathcal{H}(\ell, \mathsf{x}))$ and $e'' = \mathcal{W}_2(\mathcal{H}(\ell, \mathsf{x}))$. The vector $\overrightarrow{\hat{C}_t}$ becomes $\overrightarrow{\hat{C}_t} = (g^{-1}, 1, \ldots, 1)$ (it is the same except for the number of 1's). The security proof of the construction is given below. It requires that $\mathsf{Setup}_{\mathsf{crs}}$ also outputs some additional information or trapdoor $\mathcal{T}_{\mathsf{crs}}$, which enables to check, in polynomial time, whether a given word x is in $\mathsf{i}\mathscr{L}$ or not.

4.3.6 Proof of Security

We first provide an informal overview of the security proof. Correctness, setup indistinguishability, and zero-knowledge are straightforward. Soundness follows from the fact that (g', h', u'', e'') is a DDH-tuple, when parameters are generated by iSetup (and also $\mathsf{iTSetup}$ actually), and so $(g', 1)$ is never in the subspace generated by (g', h') and (u'', e'') (as $h' \neq 1$),

hence the corresponding language $\mathscr{L}_t$ is the same as for our iZK construction. Finally, to prove simulation-soundness, we use the programmability of the Waters function [HK12] and change the generation of the group elements $(v_{k,i})$ so that for the challenge proof (generated by the adversary) (g', h', u'', e'') is not a DDH-tuple, while for the simulated proofs it is a DDH-tuple. Then, we can change the setup to iSetup, while still being able to simulate proofs. But in this setting, the word $\vec{\hat{C}}_t$ for the challenge proof is no more in $\mathscr{L}_t$, and smoothness implies simulation-soundness.

Details. Let us first write down the complete matrices $\vec{\hat{C}}_t$ and $\Gamma_t(\mathsf{x})$:

$$\theta_t(\mathsf{x}, \zeta) = \vec{\hat{C}}_t = (g'^{-1}, 1, \ldots, 1) \qquad \in \mathbb{G}^{1\times(n+5)}$$

$$\Gamma_t(\ell, \mathsf{x}) = \left(\begin{array}{cc|ccc|cc} \multicolumn{2}{c|}{1} & \multicolumn{3}{c|}{\Gamma(\mathsf{x})} & \multicolumn{2}{c}{1} \\ \hline g' & 1 & \multicolumn{3}{c|}{\vec{\hat{C}}} & 1 & 1 \\ \hline 1 & g' & 1 & \ldots & 1 & 1 & 1 \\ g' & u' & 1 & \ldots & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & \ldots & 1 & g' & h' \\ 1 & 1 & 1 & \ldots & 1 & u'' & e'' \\ g' & 1 & 1 & \ldots & 1 & g' & 1 \end{array}\right) \qquad \in \mathbb{G}^{(k+6)\times(n+5)}$$

$$\vec{\lambda}_t(\mathsf{iw}) = (\vec{\lambda}(\mathsf{iw}), -1, 0, 0, 0, 0, 0)$$

$$\vec{\lambda}_t(\mathsf{i}\mathcal{T}) = (0, \ldots, 0, r', -1, 0, 0, 0, 0, \ldots, 0) \qquad \text{with } \mathsf{i}\mathcal{T} = r',$$

with $u'' = \mathcal{W}_1(\mathcal{H}(\ell, \mathsf{x}))$ and $e'' = \mathcal{W}_2(\mathcal{H}(\ell, \mathsf{x}))$.

Proof.

Correctness. Straightforward.

Setup Indistinguishability. The only elements added to the CRS $(v_{1,i}, v_{2,i})_i$ have exactly the same distribution when generated by iSetup and iTSetup. So it is equivalent to the setup indistinguishability of our iZK construction (see Section 4.3.4), and is implied by the DDH assumption.

Zero-Knowledge. The proof is exactly the same as for our iZK construction (see Section 4.3.4).

Soundness. Both iSetup and iTSetup output a CRS icrs, such that $(g', h', v_{1,i}, v_{2,i})$ is a DDH tuple, and so is (g', h', u'', e''). From the definition of $\vec{\hat{C}}_t$ and Γ_t, a word $(\mathsf{x}, g', h', u', e', u'', e'')$ is in the extended language corresponding to $\vec{\hat{C}}_t$ and Γ_t if and only if $\mathsf{x} \in \mathsf{i}\mathscr{L}$, or (g', h', u', e') is a DDH tuple, or $(g', 1)$ is in the subspace generated by (g', h') and (u'', e''). But the latter subspace is exactly the subspace generated by (g', h') (as (g', h', u'', e'') is a DDH tuple). Hence, $(g', 1)$ is never in that subspace (as g' and h' are supposed to be generators), and the last case of the disjunction is never satisfied.

Therefore, the extended language is actually the same as for our iZK construction, and the soundness can be proved in the same way as in Section 4.3.4.

Simulation-Soundness. Let us now prove the simulation-soundness by exhibiting a sequence of indistinguishable games. An overview of the proof is given in Section 4.3.5.

We consider an adversary $\mathscr{A}$ against the simulation soundness. In each game $\mathbf{G}_i$, we start by picking a random bit b, run some experiment, and output some bit b'. We denote by Adv_i the advantage of the adversary in the game $\mathbf{G}_i$:

$$\mathsf{Adv}_i = 2 \cdot \Pr[b' = b] - 1.$$

Finally, we write negl any negligible quantity in κ.

We recall that we suppose that $\mathsf{Setup}_{\mathsf{crs}}$ also outputs some additional information or trapdoor $\mathcal{T}_{\mathsf{crs}}$, which enables to check, in polynomial time, whether a given word x is in $\mathfrak{i}\mathscr{L}$ or not. This enables to perform the test $\mathsf{x}^* \in \mathfrak{i}\mathscr{L}^*$ (at the end of the experiment $\mathsf{Exp}^{\mathtt{iZK\text{-}ss\text{-}}b}(\mathscr{A}, \mathsf{crs}, \kappa)$) in polynomial time.

Game $\mathbf{G}_0$: In this first game, we pick a random bit b, run the experiment $\mathsf{Exp}^{\mathtt{iZK\text{-}ss\text{-}}b}(\mathscr{A}, \mathsf{crs}, \kappa)$, and outputs the bit b' (output by the experiment). We use the trapdoor $\mathcal{T}_{\mathsf{crs}}$ to test whether $x^* \in \mathfrak{i}\mathscr{L}$ or not (at the end of the experiment). The advantage Adv_0 is exactly the advantage of the adversary $\mathscr{A}$ in the simulation soundness experiments.

Game $\mathbf{G}_1$: In this game, instead of picking DDH tuples $(g', h', v_{1,i}, v_{2,i})$ in $\mathsf{iTSetup}$, we pick $v_{1,i}$ and $v_{2,i}$ uniformly at random in $\mathbb{G}$. Under the DDH assumption, $\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \mathrm{negl}$.

Game $\mathbf{G}_2$: Similarly to the proof in [LPJY14], in this game, we pick $g'', h'' \xleftarrow{\$} \mathbb{G}$, and set, for $i = 0, \ldots, 2\kappa$:

$$r'_i \xleftarrow{\$} \mathbb{Z}_p \qquad r''_i \xleftarrow{\$} \mathbb{Z}_p \tag{4.2}$$

$$v_{1,i} \leftarrow g'^{r'_i} \cdot g''^{r''_i} \cdot g'^{\rho'_i} \qquad v_{2,i} \leftarrow h'^{r'_i} \cdot h''^{r''_i}, \tag{4.3}$$

with $\rho'_0 = \mu\zeta' - \rho_0$, $\rho'_i = -\rho_i$ (for $i = 1, \ldots, 2\kappa$), $\mu \xleftarrow{\$} \{0, \ldots, 2\kappa\}$, $r'_i, r''_i \xleftarrow{\$} \mathbb{Z}_p$, $\rho_i \xleftarrow{\$} \{0, \ldots, \zeta'\}$, for $i = 0, \ldots, 2\kappa$, with $\zeta' = 2(q+1)$ and q the number of simulated proofs (i.e., queries (ℓ, x) to oracle $\mathcal{O}$). This game is perfectly indistinguishable from the previous one, as the distribution of the $v_{k,i}$'s is exactly the same: $\mathsf{Adv}_1 = \mathsf{Adv}_2$.

Game $\mathbf{G}_3$: In this game, we abort if for some query (ℓ, x) to $\mathcal{O}$, $\rho'_0 + \sum_{i=i}^{2\kappa} m_i \rho'_i = 0$, with $m = m_1 \| \ldots \| m_{2\kappa} = \mathcal{H}(\ell, \mathsf{x}) \in \{0,1\}^{2\kappa}$; or if for $m^* = m_1^* \| \ldots \| m_{2\kappa}^* = \mathcal{H}(\ell^*, \mathsf{x}^*) \in \{0,1\}^{2\kappa}$, $\rho'_0 + \sum_{i=1}^{2\kappa} m_i^* \rho'_i \neq 0$. Using the same analysis as in [Wat05; BR09; LPJY14]: $\mathsf{Adv}_2^2/(27(q+1)(2\kappa+1)) \leq \mathsf{Adv}_3$.

Game $\mathbf{G}_4$: In this game, we choose g'', h'' so that (g', h', g'', h'') is a random DDH tuple (instead of a random tuple as before). Under the DDH assumption, $\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \mathrm{negl}$.

Game $\mathbf{G}_5$: In this game, we set, for $i = 0, \ldots, 2\kappa$:

$$r'_i \xleftarrow{\$} \mathbb{Z}_p \qquad v_{1,i} \leftarrow g'^{r'_i} \cdot g'^{\rho'_i} \qquad v_{2,i} \leftarrow h'^{r'_i}, \tag{4.4}$$

with ρ_i defined as in $\mathbf{G}_2$. This game is perfectly indistinguishable from the previous one, as the distribution of the $v_{k,i}$'s is exactly the same: $\mathsf{Adv}_4 = \mathsf{Adv}_5$.

Game $\mathbf{G}_6$: In this game, for any query (ℓ, x) to $\mathcal{O}$, we generate $\mathsf{ipk} = \mathsf{tp}$ as usual, but for a subsequent query $(\mathsf{ipk} = \mathsf{tp}, c = (\zeta, \mathsf{hp}))$ to $\mathcal{O}$, we compute $\mathsf{trap}H$ (in iTDec) as $\mathsf{trap}H = \vec{\lambda}' \bullet \mathsf{hp}$ instead of $\mathsf{trap}H = \vec{\lambda}_t(\mathsf{i}\mathcal{T}) \bullet \mathsf{hp}$, where

$$\vec{\lambda}' = \left(0, \ldots, 0, -\frac{r'_0 + \sum_{i=1}^{2\kappa} r'_i}{\alpha}, \frac{1}{\alpha}, -1, 0, \ldots, 0\right),$$

and

$$m = \mathcal{H}(\ell, \mathsf{x}) \qquad\qquad \alpha = \rho'_0 + \sum_{i=1}^{n} m_i \rho'_i.$$

This vector $\vec{\lambda}'$ is well defined as $\alpha \neq 0$ from the abort condition in $\mathbf{G}_3$. Furthermore:

$$\begin{aligned}
&\left(\frac{r'_0+\sum_{i=1}^{2\kappa} r'_i}{\alpha} \quad \frac{1}{\alpha}\right) \bullet \begin{pmatrix} g' & h' \\ \mathcal{W}_1(m) = v_{1,0} \prod_{i=1}^{2\kappa} v_{1,i}^{m_i} & \mathcal{W}_2(m) = v_{2,0} \prod_{2=1}^{2\kappa} v_{1,i}^{m_i} \end{pmatrix} \\
&= \begin{pmatrix} g'^{(-r'_0-\sum_{i=1}^{2\kappa} m_i r'_i)/\alpha} g'^{(r'_0+\sum_{i=1}^{2\kappa} m_i r'_i+\rho'_0+\sum_{i=1}^{2\kappa} \rho'_i)/\alpha} \\ h'^{(-r'_0-\sum_{i=1}^{2\kappa} m_i r'_i)/\alpha} h'^{(r'_0+\sum_{i=1}^{2\kappa} m_i r'_i)/\alpha} \end{pmatrix}^{\mathsf{T}} \\
&= \left(g'^{(\rho'_0+\sum_{i=1}^{2\kappa} \rho'_i))/\alpha} \quad h'^0\right) = \begin{pmatrix} g' & 1 \end{pmatrix}
\end{aligned}$$

so that

$$\vec{\lambda}' \bullet \Gamma_t = \vec{C}_t.$$

Finally, a proof similar as the one for the zero-knowledge property of our iZK construction (see Section 4.3.4) shows that $\mathsf{Adv}_5 \leq \mathsf{Adv}_6 + \mathrm{negl}$.

Game $\mathbf{G}_7$: In this game, we generate the CRS using iSetup (i.e., (g', h', u', e') is now a random tuple instead of a DDH tuple). This is possible as $\mathsf{i}\mathcal{T}$ was not used in the previous game. Under the DDH assumption, $\mathsf{Adv}_6 \leq \mathsf{Adv}_7 + \mathrm{negl}$.

In this last game, we remark that $\vec{C}^*_t$ (corresponding to the challenge ℓ^*, x^*) is linearly independent of rows of $\Gamma_t(\ell^*, \mathsf{x}^*)$, as $\mathsf{x}^* \notin \mathsf{i}\mathscr{L}$, (g', h', u', e') is not a DDH tuple, and $(g', h', \mathcal{W}_1(\ell^*, \mathsf{x}^*), \mathcal{W}_2(\ell^*, \mathsf{x}^*))$ is a DDH tuple. Then, similarly as in the soundness proof above, we get that $\mathsf{Adv}_7 = \mathrm{negl}$ (statistically).

4.3.7 More Efficient iZK Constructions

In this section, we describe several ways to get slightly more efficient constructions of iZK at the cost of some (very reasonable) additional requirements.

4.3.7.1 Reducing the Size of the Ciphertext Using Entropy Extractors

In the generic framework constructed in Section 4.3, the ciphertext c of the iZK contains a random integer ζ. However, the actual requirement on ζ is quite simple: we want to ensure that the adversary will not be able to guess it before we send it. If the adversary was able to guess ζ, then he could have sent a $\mathsf{tproj}H$ corresponding to a linear combination of the lines of the matrix, and then $\mathsf{tproj}H$ would contain additional information about the secret key, breaking the zero-knowledge property of the iZK. To ensure that the adversary will not guess the ζ in advance, it is not necessary to send the ζ among with the other elements of

$\mathsf{Exp}^{\mathtt{iZK\text{-}zk\text{-}}b}(\mathscr{A}, \mathsf{crs}, \kappa)$
$(\mathsf{x}^*, w, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{crs})$ // only for word-dependent CRS
$\mathsf{x}^* \leftarrow \perp$ // only for re-usable CRS
$(\mathsf{icrs}, \mathsf{i}\mathcal{T}) \xleftarrow{\$} \mathsf{iTSetup}(\mathsf{crs}, \mathsf{x}^*)$
$(\ell, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{crs})$ // only for word-dependent CRS
$(\ell, \mathsf{x}^*, w, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{st}, \mathsf{icrs}, \mathsf{i}\mathcal{T})$ // only for re-usable CRS
if $\mathcal{R}(\mathsf{x}^*, w, \mathsf{st}) = 0$ **then return** 0
if $b = 0$ **then**
  $(\mathsf{ipk}, \mathsf{isk}) \xleftarrow{\$} \mathsf{iKG}^{\ell}(\mathsf{icrs}, \mathsf{x}^*, w)$
else
  $(\mathsf{ipk}, \mathsf{itk}) \xleftarrow{\$} \mathsf{iTKG}^{\ell}(\mathsf{i}\mathcal{T}, \mathsf{x}^*)$
$(c, \mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{st}, \mathsf{icrs}, \mathsf{i}\mathcal{T}, \mathsf{ipk})$
if $b = 0$ **then**
  $K \leftarrow \mathsf{iDec}^{\ell}(\mathsf{isk}, c)$
else
  $K \leftarrow \mathsf{iTDec}^{\ell}(\mathsf{itk}, c)$
return $\mathscr{A}(\mathsf{st}, K)$

Figure 4.5: Experiments $\mathsf{Exp}^{\mathtt{iZK\text{-}zk\text{-}}b}$ for zero-knowledge of iZK

the ciphertext c, as it already contains a lot of entropy: one can add the description of an entropy extractor Ext in the CRS, and the value ζ will be directly computed as $\zeta = \mathsf{Ext}(\mathsf{hp})$. This saves one element in c.

4.3.7.2 More Efficient Construction with Word-Dependent CRS

In Section 4.3, we have seen how to add a trapdoor in a SPHF to ensure the validity of the projection key. In many cases, it is possible to add the trapdoor in a slightly more efficient way, if we accept to use word-dependent CRS. (the trapdoor CRS only works for one word $\mathsf{x}^* = (u^*, e^*)$ chosen before the CRS is generated). Instead of adding three columns and three rows to the matrix Γ (to obtain the matrix Γ_t), it may be possible to only add one row. The second part of the construction ensuring the validity of the projection keys hp_t remains the same.

For example, in Example 2.3.19, the CRS can contain a row $R = (R_1, R_2, R_3, R_4)$ which is $(u^{*s}, e^{*s}, 1, 1)$ in the trapdoor mode for x^*, or $(g^s, h^s, 1, 1)$ in the normal mode (with s a random scalar in $\mathbb{Z}_p^*$). In the trapdoor mode, s is the trapdoor for x^*. The DDH assumption (or the semantic security of ElGamal) ensures that the two setups are indistinguishable. We then have:

$$\vec{\hat{C}}_t = \vec{\hat{C}} = (u, e, 1, 1)$$
$$\Gamma_t = \begin{pmatrix} \Gamma \\ \hline R \end{pmatrix} = \begin{pmatrix} g & h & 1 & 1 \\ 1 & g & u & e/g \\ 1 & 1 & g & h \\ \hline R_1 & R_2 & R_3 & R_4 \end{pmatrix}.$$

In normal mode, the last row R is s times the first row of Γ_t, and so the new element in the projection key, $\mathsf{hp}_{t,4} = R \bullet \mathsf{hk}$ gives no more information than the first element $\mathsf{hp}_{t,1} = \mathsf{hp}_1$

(from an information theoretic point of view). That is why the smoothness does still hold in normal mode.

In trapdoor mode, we remark that $\mathsf{hp}_{t,4} = R \bullet \mathsf{hk} = (u^{*\mathsf{hk}_1} e^{*\mathsf{hk}_2})^s$. This is exactly the hash value of x^* raised to the power of s (if hp is valid). So knowing the trapdoor s and $\mathsf{hp}_{t,4}$ enables to compute the hash value of C^*.

Formal Construction of iZK with Word-Dependent CRS. We suppose to have two setup algorithms:

- $\mathsf{iSetup}(\mathsf{crs})$ generates a row vector $R \in \mathbb{G}^{1\times n}$ which is linearly depend of the rows of any matrix Γ for any C (we recall that Γ may depend on C). Then it returns $\mathsf{icrs} = (\mathsf{crs}, R)$.
- $\mathsf{iTSetup}(\mathsf{crs}, \mathsf{x}^*)$ generates a row vector $R \in \mathbb{G}^{1\times n}$ and a trapdoor a row vector $\overrightarrow{\lambda^*} \in \mathbb{Z}_p^{k+1}$ so that:
$$\overrightarrow{\lambda^*} \bullet \begin{pmatrix} \Gamma \\ \hline R \end{pmatrix} = \vec{\hat{C}}.$$
In other word $\overrightarrow{\lambda^*}$ is a witness for the language defined by $\begin{pmatrix} \Gamma \\ \hline R \end{pmatrix}$ and θ. Then it returns $\mathsf{icrs} = (\mathsf{crs}, R)$ and $\mathsf{i}\mathcal{T} = \overrightarrow{\lambda^*}$.

Then we do the same construction as in Section 4.3.2, except we use the following matrices $\Gamma_t(\mathsf{x})$, $\vec{\hat{C}}_t$, $\overrightarrow{\lambda_t}(\mathsf{iw})$, and $\overrightarrow{\lambda_t}(\mathsf{i}\mathcal{T})$:

$$\vec{\hat{C}}_t = \vec{\hat{C}} \qquad \Gamma_t = \begin{pmatrix} \Gamma \\ \hline R \end{pmatrix}$$
$$\overrightarrow{\lambda_t}(\mathsf{iw}) = (\overrightarrow{\lambda}, 0) \qquad \overrightarrow{\lambda_t}(\mathsf{i}\mathcal{T}) = \overrightarrow{\lambda^*}.$$

If the two setup iSetup and $\mathsf{iTSetup}$ are indeed indistinguishable, the proof of security is almost identical to the one for the generic construction in Section 4.3.4.

When it is usable, this construction is slightly more efficient than the generic one with re-usable CRS, since the resulting matrix Γ_t has 4 less columns and 4 less rows.

4.3.8 iZK for Languages Defined by a Computational Structure

We have shown that a SPHF for some language $\mathscr{L}$ yields an iZK for the same language $\mathsf{i}\mathscr{L} = \mathscr{L}$. However, if the class of NP languages handled by SPHFs is sufficient for many applications, there is still a large variety of useful languages which are not captured by the framework we presented above. We thus now (informally) explain how to construct iZK for any languages just from their representation through a given computational structure.

Of course, every NP language can be represented by the most general computational structure, the *circuit*. However, more efficient, but more restricted computational structures are widely used in cryptography, such as Boolean branching programs, arithmetic formulas, etc. A computational structure of particular interest is the model of *Arithmetic Branching Programs* (ABP). They provide a very compact way to represent multivariate polynomials and capture, among others, the two structure previously given.

A language $\mathsf{i}\mathscr{L}$ represented by a computational structure can be converted into a language $\mathscr{L}$ which can be handled by the generic framework for SPHFs, by essentially extending the words with commitments to particular elements of the computational structure itself. Thus,

on a given language, we can construct an iZK whose size is essentially the size of the most efficient computational structure which can represent the language.

In the following, we present the main ideas of how to construct an iZK for any NP language defined by a circuit, and also for any language defined by an ABP. We stress that they represent the most commonly used, and the most interesting, computational structures, but iZK can be constructed for others computational structures, depending of our need — other constructions exist for other representations of languages and these examples aim at illustrating the way such constructions can be made.

For any NP Language Defined by a Circuit. Let us build an iZK for an NP language $\mathscr{L}$ defined by a (polynomial-size) circuit $\mathscr{C}$ that evaluates a function F: a word x is in $\mathscr{L}$ if and only if there exists a witness iw verifying $F(\mathsf{x}, \mathsf{iw}) = 1$. We remark that any NP language can be defined by such a circuit.

The idea for the iZK construction is the following: the prover sends (as part of the public key ipk) ElGamal ciphertexts encrypting both all the bits of iw and all the values of the wires of the circuit $\mathscr{C}$ when evaluated on x and iw. Then he uses an SPHF to implicitly prove that:

- encryption of input bits of iw indeed contain bits (which is our Example 2.3.17);
- encryption of the output wire of the circuit really contains 1 (which is similar to our Example 2.3.15);
- each gate is evaluated correctly.

All these properties are guaranteed together by the conjunction of all the languages, as in our Example 2.3.18. It is thus indeed sufficient to show how to handle every individual language with the generic framework for SPHFs. The resulting scheme is an iZK for the NP language defined by $\mathscr{C}$, secure under plain DDH. It is straightforward to extend it to be secure under weaker assumptions such as DLIN.

For Languages Defined by an ABP. Arithmetic branching programs (ABP) are efficient computational models that capture, among others, the computation of Boolean formulas, Boolean branching programs and arithmetic formulas. They also give a very compact representation of multivariate polynomials. A branching program is defined by a directed acyclic graph (V, E) with two special vertices $\mu, \nu \in V$ and a labeling function Φ. An ABP computes a function $F : \mathbb{F}_p^\ell \to \mathbb{F}_p$ (p is a prime power) as follows: Φ assigns to each edge of E either a constant value or an affine function in any number of the input variables of F, and $F(z)$ is the sum over all the path from μ to ν of the product of all the values along the path. The evaluation of F can be performed by assigning a value to each node, when nodes are sorted topologically (i.e., in such an ordering, a node appears always after its predecessors). The last node is ν and its value if the value $F(z)$.

In our case, we use ABP to define an NP language in the following way: a word x is in the language $\mathscr{L}$ if there exists a witness iw such that $F(\mathsf{x}, \mathsf{iw}) = 0$. The prover sends (as part of the public key ipk), ElGamal ciphertexts encrypting both all the bits of iw and all the values of the nodes when Φ is instantiated with x and iw. Then, as above, he uses a SPHF to implicitly prove that:

- encryption of input bits of iw indeed contain bits;
- encryption of the last node ν really contains 0;

- each value for the nodes are computed correctly; the plaintext is just the sum of the values of the previous nodes multiplied by affine evaluations on the input $(\mathsf{x}, \mathsf{iw})$.

Every individual language can be efficiently represented by an SPHF, and then conjunctions help to conclude, under the DDH assumption.

4.3.8.1 iZK for any NP Language Defined by a Circuit

In every construction described below, we consider that the additively homomorphic ElGamal encryption scheme is used. We will denote $\mathcal{E}_{\mathsf{pk}}(a; r)$ the encryption of a under the public key pk and with randomness r.

Notations. Let $F : \{0,1\}^\ell \to \{0,1\}$ be a function computed by a circuit $\mathscr{C}$ on a basis B of boolean gates (with two input wires, without loss of generality), given by its directed acyclic graph (V, E). F takes as input $z = (\mathsf{x}, \mathsf{iw})$ where $\mathsf{x} \in \{0,1\}^{\ell_\mathsf{x}}$ and $\mathsf{iw} \in \{0,1\}^{\ell_\mathsf{iw}}$ such that $\ell_\mathsf{x} + \ell_\mathsf{iw} = \ell$. Nodes or gates v in V are either an input gate corresponding to some bit x_i of x, iw_i of iw, or a constant bit, or a boolean gate in the basis B. Let $s = |V|$ be the size of the circuit. We consider the partial order on the set of gates V defined by $u \preccurlyeq v$ if there is a path from the gate u to the gate v (the graph is acyclic). Then, we index the gates $V = (v_i)_i^s$, in an order-preserving way, such that for $i = 1, \ldots, \ell$, v_i corresponds to the input bit z_i of z and, if $v_i \preccurlyeq v_j$, then $i \leq j$. For each internal gate v_i, with $i > \ell$, we denote by $\{i_1, i_2\} = \mathcal{P}(i)$ the indexes of the two preceding gates whose outputs are the inputs of v_i. The output bit of the gate v_i when evaluated on $z = (\mathsf{x}, \mathsf{iw})$ is denoted A_i (for input gates, the output bit is just the value of the input).

Extended Language for F. We want an iZK for the language

$$\mathsf{i}\mathscr{L} = \{\mathsf{x} \in \{0,1\}^{\ell_\mathsf{x}} \mid \exists \mathsf{iw} \in \{0,1\}^{\ell_\mathsf{iw}}, F(\mathsf{x}, \mathsf{iw}) = 1\}.$$

However, this language cannot be directly handled by the SPHF framework, and we have to extend it first: we consider the extended language $\mathscr{L}$ of words of $\mathsf{i}\mathscr{L}$ along with the encryption of the output bits A_i of the gates v_i for $i > \ell_\mathsf{x}$ (hence including the input gates corresponding to the bits of iw but excluding those corresponding to the bits of x, which are anyway already known). Witness for the new language will be the random coins for all the ciphertexts, together with the values A_i for $i > \ell_\mathsf{x}$. We recall that $A_i = \mathsf{x}_i$ for $i = 1, \ldots, \ell_\mathsf{x}$.

Formally, for a gate v_i, let $(\beta_i, \beta_i^+, \beta_i^\times)$ be three integers such that, on input (x, y), the output of the gate is $(\beta_i + \beta_i^+(x + y) + \beta_i^\times xy)$. This models all the (symmetric) binary gates: $\mathsf{XOR} = (0, 1, -2)$, $\mathsf{OR} = (0, 1, -1)$, $\mathsf{AND} = (0, 0, 1)$, $\mathsf{NAND} = (1, 0, -1)$, while the unary gate NOT is just XOR 1. For $i = \ell_\mathsf{x} + 1, \ldots, s$, we consider a ciphertext $c_i = \mathcal{E}(A_i; r_i)$ of A_i with random coins r_i. We now consider the language $\mathscr{L}$ of the words $C = (\mathsf{x}, (c_i)_{i=\ell_\mathsf{iw}+1}^s)$ such that there exist witnesses $(A_i, r_i)_{i=\ell_\mathsf{iw}+1}^s$ satisfying: $A_s = 1$, for all $i = \ell_\mathsf{x} + 1, \ldots, s$, c_i encrypts the bit A_i with random coins r_i, and, for $i = \ell, \ldots, s$, A_i verifies the appropriate relation with A_{i_1} and A_{i_2}, for $\{i_1, i_2\} = \mathcal{P}(i)$. However, there are quadratic relations, we thus need additional variables to linearize the system.

Now, let us show how to construct an SPHF on this language $\mathscr{L}$ which can be automatically used to construct an iZK using the framework defined in Section 4.3 for the above language $\mathsf{i}\mathscr{L}$. Concretely, we use an ElGamal encryption in basis g, with public key h, and we write $c_i = (c_{i1} = g^{r_i}, c_{i2} = h^{r_i} g^{A_i})$. $C = (\mathsf{x}, (c_i)_{i=\ell_\mathsf{iw}+1}^s)$ is in $\mathscr{L}$ if and only if there exist

Algorithm 1 Dynamic ABP Computation

```
1: procedure DAC(F, x) // F is an ABP and x is its input
2:     A_0 ← 1
3:     for i = 1 to |V| do
4:         A_i ← 0
5:         for all v_j ∈ prec(v_i) do
6:             A_i ← A_i + Φ((v_j → v_i), x) · A_j   // A_0 is set as the value of the predecessor of v_1
7:     return (A_i)_{2≤i≤|V|}   // A_{|V|} = F(x)
```

$(A_i)_{i=\ell_{\mathsf{x}}+1}^{s} \in \{0,1\}^{s-\ell_{\mathsf{x}}}$, $(r_i)_{i=\ell_{\mathsf{x}}+1}^{s} \in \mathbb{Z}_p^{s-\ell_{\mathsf{x}}}$, $(\mu_i)_{i=\ell_{\mathsf{x}}+1}^{s} \in \mathbb{Z}_p^{s-\ell_{\mathsf{x}}}$ and $(\mu'_i)_{i=\ell+1}^{s} \in \mathbb{Z}_p^{s-\ell}$, such that:

$$g^{r_i} = c_{i1} \quad \text{and} \quad h^{r_i} \cdot g^{A_i} = c_{i2}$$
$$(c_{i1})^{A_i} \cdot g^{-\mu_i} = 1 \quad \text{and} \quad (c_{i2}/g)^{A_i} \cdot h^{-\mu_i} = 1$$

for $i = \ell_{\mathsf{x}} + 1, \ldots, s$ and:

$$(c_{i_2 1})^{A_{i_1}} \cdot g^{-\mu'_i} = 1 \quad \text{and} \quad g^{\beta_i^+ A_{i_1}} \cdot g^{\beta_i^+ A_{i_2}} \cdot (c_{i_2 2})^{\beta_i^\times A_{i_1}} \cdot h^{-\beta_i^\times \mu'_i} \cdot g^{-A_i} = g^{-\beta_i}$$

for $i = \ell + 1, \ldots, s$, with $\{i_1, i_2\} = \mathcal{P}(i)$, since the second of equations ensures $\mu_i = r_i A_i$ and $A_i(A_i - 1) = 0$ (i.e., A_i is a bit), while the third one ensures $\mu'_i = r_{i_2} A_{i_1}$ and $A_i = \beta_i + \beta_i^+(A_{i_1} + A_{i_2}) + \beta_i^\times A_{i_1} A_{i_2}$. These linear equations (in the exponents) directly provides the matrix $\Gamma(C)$, while $\theta(C)$ is defined by the right-hand sides of the relations. This then leads to an SPHF over $\mathscr{L}$, based on the plain DDH.

4.3.8.2 iZK for any NP Language Defined by an ABP

Notations: Let $F : \mathbb{Z}_p^\ell \to \mathbb{Z}_p$ be a function computed by an ABP given by its directed acyclic graph (V, E), two special vertices $\mu, \nu \in V$ and a labeling function $\Phi : E \times \mathbb{Z}_p^\ell \to \mathbb{Z}_p$. F takes as input $z = (\mathsf{x}, \mathsf{iw})$, where $\mathsf{x} \in \mathbb{Z}_p^{\ell_{\mathsf{x}}}$ and $\mathsf{iw} \in \mathbb{Z}_p^{\ell_{\mathsf{iw}}}$ such that $\ell_{\mathsf{x}} + \ell_{\mathsf{iw}} = \ell$. Let $s = |E|$ be the size of the ABP. We denote by $(u \to v)$ the edge from the vertex u to the vertex v. We consider the partial order on the set of vertices V defined by $u \preccurlyeq v$ if there is a path from the gate u to the gate v (the graph is acyclic). Then, we index the vertices $V = (v_i)_s$ in an order-preserving way: $v_i \preccurlyeq v_j \Rightarrow i \leq j$, $\mu = v_1$ and $\nu = v_{|V|}$. For each node $v \neq \mu$, we denote by $\mathrm{prec}(v)$ the set of direct predecessors of v, i.e., the vertices u such that $(u \to v) \in E$. Algorithm 1 describes the way the ABP is evaluated in an input x. When the input x can be seen as a pair of tuples $x = (\mathsf{x}, \mathsf{iw}) \in \mathbb{Z}_p^{\ell_{\mathsf{x}}} \times \mathbb{Z}_p^{\ell_{\mathsf{iw}}} = \mathbb{Z}_p^{\ell}$, we consider the problem, for a given x, of the existence of a witness iw such that $F(\mathsf{x}, \mathsf{iw}) = 0$. We want to build an iZK on the language of the words x with such witnesses iw.

Extended Language for F. As above, we want an iZK for the language

$$\mathfrak{i}\mathscr{L} = \{x \in \mathbb{Z}_p^{\ell_{\mathsf{x}}} \mid \exists \mathsf{iw} \in \mathbb{Z}_p^{\ell_{\mathsf{iw}}}, F(\mathsf{x}, \mathsf{iw}) = 0\}.$$

We can extend it, as above, with the ciphertexts c_i of all the witnesses iw_i and a_i of all intermediate values A_i of the dynamic ABP computation (except the special vertices $A_1 = 1$

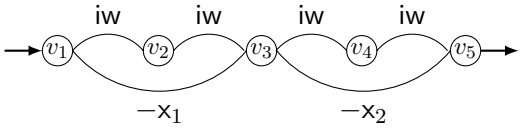
and $A_{|V|} = 0$). Then the witnesses are $(r_i)_{i=1}^{\ell_{\mathsf{iw}}}$ and $(s_i)_{i=2}^{|V|-1}$, the random coins for the encryption. We now consider the language $\mathscr{L}$ of the words $(\mathsf{x}, (c_i)_{i=1}^{\ell_{\mathsf{iw}}}, (a_i)_{i=2}^{|V|-1})$ such that there exist witnesses $((r_i, \mathsf{iw}_i)_{i=1}^{\ell_{\mathsf{iw}}}, (s_i, A_i)_{i=2}^{|V|-1})$ satisfying: for all $i = 1, \dots \ell_{\mathsf{iw}}$, c_i encrypts the scalar iw_i with random coins r_i, for $i = 2, \dots, |V| - 1$, a_i encrypts the scalar A_i with random coins s_i, and A_i verifies the appropriate relation w.r.t. its predecessors, as well $A_1 = 1$ and $A_{|V|} = 0$, which introduces again quadratic relations.

As above, using ElGamal encryption, we can write $c_i = (c_{i1} = g^{r_i}, c_{i2} = h^{r_i} g^{\mathsf{iw}_i})$ and $a_i = (a_{i1} = g^{s_i}, a_{i2} = h^{s_i} g^{A_i})$. The word $C = (\mathsf{x}, (c_i)_{i=1}^{\ell_{\mathsf{iw}}}, (a_i)_{i=2}^{|V|-1})$ is in $\mathscr{L}$ if and only if there exist $(\mathsf{iw}_i)_{i=1,\dots,\ell_{\mathsf{iw}}} \in \mathbb{Z}_p^{\ell_{\mathsf{iw}}}$, $(r_i)_{i=1,\dots,\ell_{\mathsf{iw}}} \in \mathbb{Z}_p^{\ell_{\mathsf{iw}}}$, $(A_i)_{i=2}^{|V|-1} \in \mathbb{Z}_p^{|V|-2}$, $(s_i)_{i=2}^{|V|-1} \in \mathbb{Z}_p^{|V|-2}$, and $\mu_{i,j} \in \mathbb{Z}_p$, for $i = 2, \dots, |V| - 1$ and $j = 1, \dots, \ell_{\mathsf{iw}}$ (but actually for all the values iw_j that appears in labels on edges leaving from v_i), such that:

$$
\begin{aligned}
g^{r_i} = c_{i1} \ \text{ and } \ h^{r_i} \cdot g^{\mathsf{iw}_i} = c_{i2} \qquad & \text{for } i = 1, \dots, \ell_{\mathsf{x}} \\
g^{s_i} = a_{i1} \ \text{ and } \ h^{s_i} \cdot g^{A_i} = a_{i2} \qquad & \text{for } i = 2, \dots, |V| - 1 \\
(a_{i1})^{\mathsf{iw}_j} \cdot g^{-\mu_{i,j}} = 1 \qquad & \text{for } i = 2, \dots, |V| - 1, \text{ for } j = 1, \dots, \ell_{\mathsf{iw}} \\
g^{\sum_{v_j \in \mathrm{prec}(v_i)} A_i \cdot \Phi((v_j \to v_i), x) - A_i} = 1 \qquad & \text{for } i = 2, \dots, |V|, \text{ with } A_{|V|} = 0
\end{aligned}
$$

where $x = (\mathsf{x}, \mathsf{iw})$. We recall that $\Phi(v_j \to v_i)$ is an affine function (or a constant) in x (known by both players) and iw (encrypted in the c_i's). So quadratic terms $A_i \mathsf{iw}_j$ can be computed using the intermediate value $\mu_{i,j}$, as above, that implicitly corresponds to $r_i \mathsf{iw}_j$ to remove extra terms in h introduced by $(a_{i2})^{\mathsf{iw}_j}$: $(a_{i2})^{\mathsf{iw}_j} \cdot h^{-\mu_{i,j}}$ is indeed $g^{A_i \mathsf{iw}_j}$ when the first row is enforced.

A Concrete Example. Let us consider the following language $\mathsf{i}\mathscr{L} = \{(\mathsf{x}_1, \mathsf{x}_2) \in \mathbb{Z}_p^2 \mid \exists \mathsf{iw} \in \mathbb{Z}_p, (\mathsf{iw}^2 - \mathsf{x}_1)(\mathsf{iw}^2 - \mathsf{x}_2) = 0\}$ of pairs of integers modulo p such that at least one of the elements of the pair is a square. This language can be efficiently represented by the following ABP:



Applying the dynamic ABP computation algorithm, we get $A_1 = 1$, $A_2 = \mathsf{iw}$, $A_3 = \mathsf{iw}A_2 - \mathsf{x}_1 A_1 = \mathsf{iw}^2 - \mathsf{x}_1$, $A_4 = \mathsf{iw}A_3$, and $A_5 = (\mathsf{iw}^2 - \mathsf{x}_2)A_3 = (\mathsf{iw}^2 - \mathsf{x}_1)(\mathsf{iw}^2 - \mathsf{x}_2)$. Thus, we construct the extended language of words $\mathscr{L}' = \{(\mathsf{x}_1, \mathsf{x}_2), (c_1, c_2), (a_{i1}, a_{i2})_{i=2}^4\}$ such that there exists $(\mu_i)_{i=2}^4 \in \mathbb{Z}_p^3$ so that the plaintexts iw, and (A_2, A_3, A_4) satisfy:

$$
\begin{aligned}
g^r = c_1 \ \text{ and } \ h^r \cdot g^{\mathsf{iw}} = c_2 & \\
g^{s_i} = a_{i1}, \ h^{s_i} \cdot g^{A_i} = a_{i2} \ \text{ and } \ (a_{i1})^{\mathsf{iw}} \cdot g^{-\mu_i} = 1 \qquad & \text{for } i = 2, 3, 4 \\
g^{\mathsf{iw}} \cdot g^{-A_2} = 1 \ \text{ and } \ (a_{22})^{\mathsf{iw}} \cdot h^{-\mu_2} \cdot g^{-A_3} = g^{\mathsf{x}_1} & \\
(a_{32})^{\mathsf{iw}} \cdot h^{-\mu_3} \cdot g^{-A_4} = 1 \ \text{ and } \ (a_{42})^{\mathsf{iw}} \cdot h^{-\mu_4} = g^{\mathsf{x}_2} &
\end{aligned}
$$

We have 15 equations and 11 witnesses $(\mathsf{iw}, r, (A_i)_2^4, (s_i)_2^4, (\mu_i = s_i \mathsf{iw})_2^4)$. However, in this particular example, we can drop three equations and two witnesses: as the value A_2 is exactly the witness iw, we can drop (c_1, c_2) and use (a_{21}, a_{22}) instead. In addition, we can remove

the two first equations and the equation $g^{\mathsf{iw}} \cdot g^{-A_2} = 1$: we now have 12 equations and 10 witnesses $((A_i)_2^4, (s_i)_2^4, (\mu_i = s_i\mathsf{iw})_2^4)$. We stress the fact that in particular applications with a "good" structure, it is often possible to get optimizations on the theoretical size of the corresponding iZK. This leads to a iZK with public key of size $|\mathsf{ipk}| = 30$ and ciphertexts of size $|c| = 24$ (using the optimization of 4.3.7).

4.4 Applications

4.4.1 Semi-Honest to Malicious Transformation

In the seminal work [GMW87b], Goldreich, Micali and Wigderson have proven that there exists a compiler which, given any two-party semi-honest interactive protocol, outputs an "equivalent protocol" for the malicious model. This compiler (which we call GMW compiler) is formally described in [Gol04]. It is divided in three phases: the Input-Commitment Phase, where the players commit to their own inputs; the Coin-Generation Phase, where the players run an augmented coin-tossing protocol to generate unbiased random tapes while providing commitments on them for later validity proofs; and the Protocol Emulation Phase, where zero-knowledge proofs are used to ensure semi-honest behavior of all the players, from the committed inputs, the committed random tapes and the flows. This last phase is the one on which we focus in this section.

Indeed, while NIZK could be used to prove correct generation of the flows, they would either be quite inefficient (with general NIZK constructions) or require strong settings and assumptions (assumptions in bilinear groups for Groth-Sahai NIZK). On the other hand, interactive zero-knowledge proofs imply a blow-up in the interactivity of the protocol.

We present another compiler (see Figure 4.6) which is divided in four phases: there are still the Input-Commitment Phase and the Coin-Generation Phase, which end up with commitments of the inputs and of the unbiased random tapes of the two players, as in the GMW compiler. Note that if inputs should belong in a non-trivial language, validity of the commitments has to be proven as in the next phase. These are constant-round phases, which are then followed by the Protocol Emulation Phase: each flow x from the initial protocol is combined with an iZK, and so with a public key ipk, so that the other player can mask all the subsequent flows with K (or derivative masks) encapsulated in c. More precisely, from the ephemeral key K, we write $k^{(i)}$ for $\mathrm{PRG}^{(i)}(K)$, and each flow is masked by all the previous keys, and so we use the next block from the PRG for any new mask. Hence, as soon as one player tries to cheat, all the subsequent flows sent by the other player will be masked by a random value. Eventually, a Verification Phase provides an explicit validity check: the two players have to prove they were able to extract all the ephemeral keys, which guarantees their semi-honest behavior during the whole protocol.

Proof Sketch. For the security proof, we first assume we are dealing with a deterministic function: on private inputs x and y, the first player receives $f(x, y)$ and the second receives $g(x, y)$. For the sake of simplicity, we also make the assumption that the semi-honest protocol provides execution traces with formats (size and number of flows) that are independent of the inputs. Eventually, we make use of extractable commitments.

We are thus given a simulator $\mathcal{S}im$ for the semi-honest protocol $\mathcal{P}$. And we describe a simulator $\mathcal{S}im'$ for the compiled protocol $\mathcal{P}'$: If both players are honest, $\mathcal{S}im'$ simply runs the simulator $\mathcal{S}im$ to generate all the basic flows, and generates all the iZK proofs as well

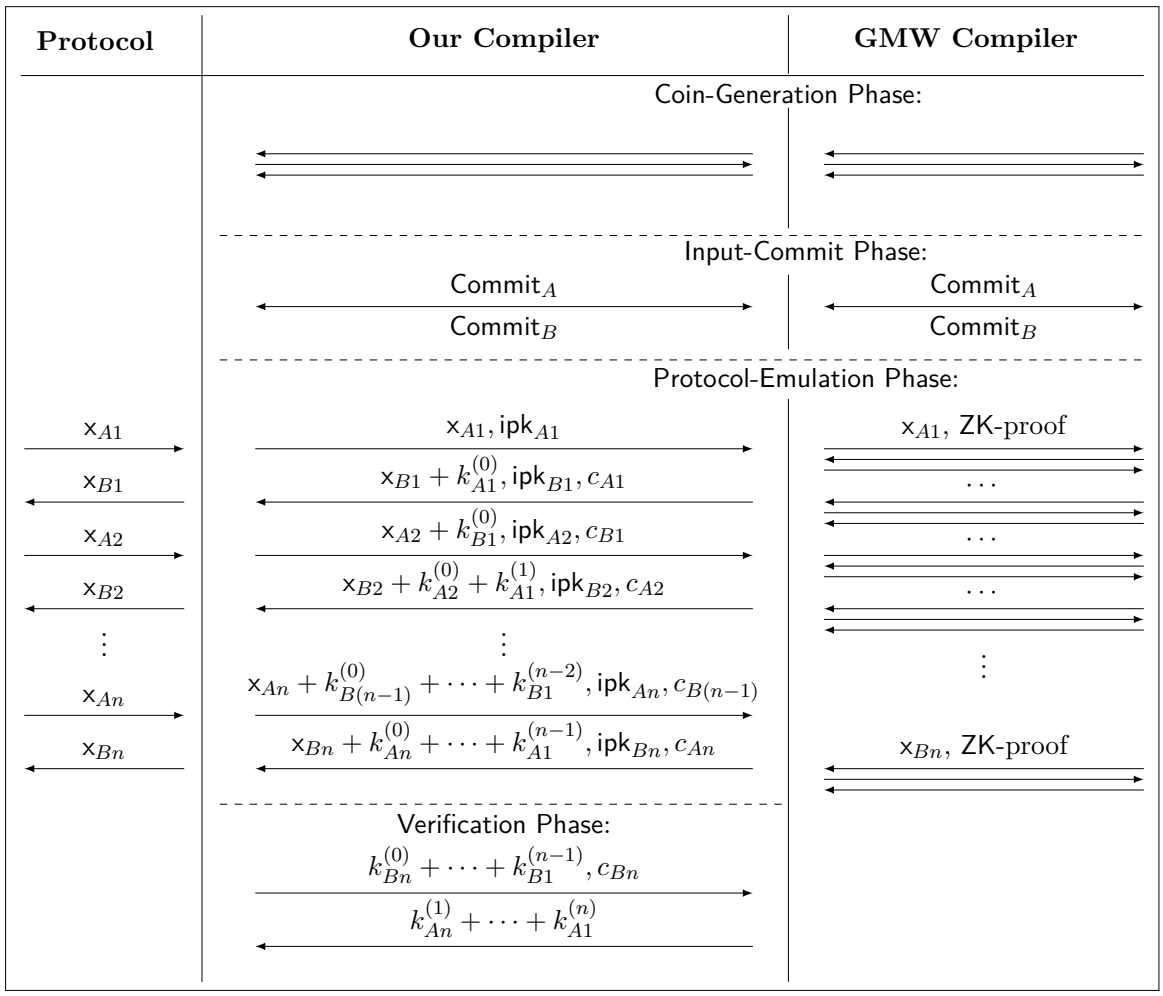


Figure 4.6: Semi-honest to malicious compilers

as the verification flows, but using random keys K for deriving the masks. If one player is malicious, $\mathcal{S}im'$ first extracts its inputs and random coins from the extractable commitment, sends the inputs to the ideal functionality to learn the outcome and provides it to $\mathcal{S}im$ to generate the basic flows of the honest player. This time, valid iZK proofs for the flows of the honest player have to be generated since the malicious player will be able to check them, and $\mathcal{S}im'$ has to be able to immediately detect dishonest behavior of the malicious player in order to replace all the subsequent flows by random flows: the trapdoor for the iZK, in the CRS, allows $\mathcal{S}im'$ to extract the ephemeral key even without a witness, and then to get back the plaintext sent by the malicious player; from the inputs and the random tape of the malicious player, as well as the previous flows already exchanged, $\mathcal{S}im'$ can anticipate and check the flow that should have been generated with a semi-honest behavior. As soon as a cheating attempt is detected, in the real world, the subsequent masks would become random looking to the malicious player, $\mathcal{S}im'$ can thus safely send random flows (the masked parts).

4.4.2 Secure Computation of Inner Products

In case of biometric authentication, a server $\mathcal{S}$ wants to compute the Hamming distance between a fresh user's feature and the stored template, but without asking the two players to reveal their own input: the template y from the server side and the fresh feature x from the client side. One can see that the Hamming distance between the ℓ-bit vectors x and y is the sum of the Hamming weights of x and y, minus twice the inner product of x and y. Let us thus focus on this private evaluation of the inner product: a client $\mathcal{C}$ has an input $x = (x_i)_{i=1}^{\ell} \in \{0,1\}^{\ell}$ and a server $\mathcal{S}$ has an input $y = (y_i)_{i=1}^{\ell} \in \{0,1\}^{\ell}$. The server $\mathcal{S}$ wants to learn the inner product $\mathsf{IP} = \sum_{i=1}^{\ell} x_i y_i \in \{0, \ldots, \ell\}$, but nothing else, while the client $\mathcal{C}$ just learns whether the protocol succeeded or was aborted.

Semi-Honest Protocol. $\mathcal{C}$ can send an ElGamal encryption of each bit under a public key of her choice and then $\mathcal{S}$ can compute an encryption of $\mathsf{IP} + R$, with $R \in \mathbb{Z}_p$ a random mask, using the homomorphic properties of ElGamal, and sends this ciphertext. $\mathcal{C}$ finally decrypts and sends back $g^{\mathsf{IP}+R}$ to $\mathcal{S}$ who divides it by g^R to get g^{IP}. Since IP is small, an easy discrete logarithm computation leads to IP.

Malicious Setting. To transform this semi-honest protocol into one secure against malicious adversaries, we could apply our generic conversion presented in Section 4.4.1. Here, we propose an optimized version of this transformation for this protocol. We use the ElGamal scheme for the encryption $\mathcal{E}_{\mathsf{pk}}$, where pk is a public key chosen by $\mathcal{C}$ and the secret key is $\mathsf{sk} = (\mathsf{sk}_j)_{j=1}^{\log p}$, and the Cramer-Shoup scheme [CS98] for commitments C, of group elements or multiple group elements with randomness reuse, where the public key is in the CRS. The CRS additionally contains the description of a cyclic group and a generator g of this group. The construction is presented on Figure 4.7. First, the client commits to her secret key (this is the most efficient alternative as soon as $n \gg \ell$) and sends encryptions $(c_i)_{i \leq n}$ of her bits. Then, the server commits to his inputs $(y_i)_i$ and to two random integers (R, R'), computes the encryption $(\hat{u}, \hat{e})$ of $g^{R \cdot \mathsf{IP} + R'}$), re-randomized with a randomness ρ, masked by an iZK to ensure that the c_i's encrypt bits under the key pk whose corresponding secret key sk is committed (masking one of the two components of an ElGamal ciphertext suffices). The client replies with $g^{R \cdot \mathsf{IP} + R'}$, masked by a SSiZK (this is required for UC security) to ensure that the $\mathsf{C}(g^{y_i})$ contains bits, and that the masked ciphertext has been properly built. The server then recovers $g^{R \cdot \mathsf{IP} + R'}$, removes R and R', and tries to extract the discrete logarithm IP. If no solution exists in $\{0, \ldots, \ell\}$, the server aborts. This last verification avoids the 2-round verification phase from our generic compiler: if the client tries to cheat on $R \cdot \mathsf{IP} + R'$, after removing R and R', the result would be random, and thus in the appropriate range with negligible probability ℓ/p, since ℓ is polynomial and p is exponential. We prove in Section 4.4.4 that *the above protocol is secure against malicious adversaries in the UC framework with static corruptions, under the plain DDH assumption, and in the common reference string setting.*

Efficiency and Comparison with Other Methodologies. In Section 4.4.3, we provide a detailed analysis of our inner product protocol in terms of complexity. Then, we estimate the complexity of this protocol when, instead of using iZK, the security against malicious adversaries in the UC model is ensured by using the Groth-Sahai methodology [GS08] or Σ-protocols. In this section, we sum up our comparisons in a table. The notation $>$ indicates that the given complexity is a lower bound on the real complexity of the protocol (we have not taken into account the linear blow-up incurred by the conversion of NIZK into $\mathsf{SS\text{-}NIZK}$), and $\gg$ indicates a very loose lower bound. Details are given in Section 4.4.3. We stress

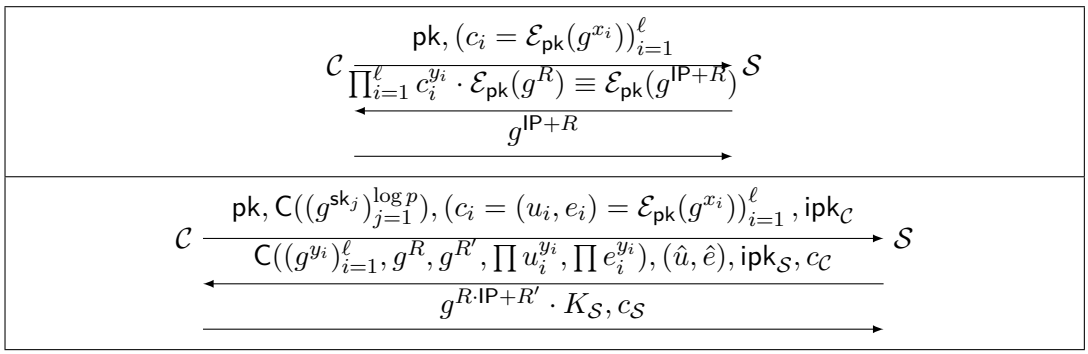


Figure 4.7: Semi-Honest and Malicious Protocols for Secure Inner Product Computation

Proofs	Pairings	Exponentiations	Communication	Rounds
Σ-proofs	0	38ℓ	20ℓ	5
GS proofs	$> 14\ell$	$\gg 28\ell(\mathbb{G}_1) + 6\ell(\mathbb{G}_2)$	$> 11\ell(\mathbb{G}_1) + 10\ell(\mathbb{G}_2)$	3
iZK (this work)	0	45ℓ	14ℓ	3

Table 4.1: Comparison of the costs of various approaches for UC-secure two-party computation of the inner product

that with usual parameter, an element of $\mathbb{G}_2$ is twice as big as an element of $\mathbb{G}_1$ (or $\mathbb{G}$) and the number of rounds in the major efficiency drawback (see Section 4.1). The efficiency improvement of iZK compared to NIZK essentially comes from their "batch-friendly" nature (see Section 4.4.3). Estimations of the costs of the three approaches are given on Table 4.1.

Moreover, our iZKs do not require pairings, which allows us to use more efficient elliptic curves than the best existing curves for the Groth-Sahai methodology. With a reasonable choice of two curves, one without pairing and one with pairing, for 128 bits of security, we get the results represented on Table 4.2 (counting efficiency as a multiple of the running time of an exponentiation in $\mathbb{G}_1$)

4.4.3 Details on the Inner Product Protocols

We will now provide a detailed analysis of the performances of our UC-secure protocol to compute the inner product. Next, we compare the performances to the performances of a similar protocol whose security is based on the Groth-Sahai methodology [GS08] to illustrate the fact that, in applications where pairings are not fundamentally required for the protocol (meaning, the semi-honest version of the protocol can be done without pairings), being

Curve \ Efficiency	Pairings	Exponentiations in $\mathbb{G}_1$	Exponentiations in $\mathbb{G}_2$
Curve25519 [Ber06]	no pairings	1	✗
[BGM+10]	≈ 8	≈ 3	≈ 6

Table 4.2: Costs for computing exponentiations and pairings in different curves

able to avoid them allows us to provide way more efficient solutions. We also provide the performances of a protocol based on the Schnorr proofs (Σ-proofs), which implies more interactivity.

4.4.3.1 Intuition on the Efficiency Improvements

First, let us provide an intuition of the reasons why we can expect some efficiency improvement over round-efficient protocols in the malicious setting based on NIZK.

Avoiding Pairings Saves Computations. Pairing are an expensive operation; on the best known curves such as [BGM+10], computing a pairing is roughly three time slower than computing an exponentiation. Moreover, not every elliptic curve has a pairing, and it turns out that the most efficient curves, such as [Ber06], have indeed no pairings. In the best curves without pairings, exponentiations in $\mathbb{G}$ are roughly three times faster than exponentiations in $\mathbb{G}_1$ in the best curves with pairings, and even six times faster than exponentiations in $\mathbb{G}_2$.

iZK Can be Efficiently Batched. iZK are somewhat "batch-friendly": batch techniques, which reduce computation and communication, can always be used with an iZK without requiring more interactions. To batch a proof in NIZK-based protocols a seed is needed, so the prover has to first commit to his values, then he receives the seed and computes a short NIZK from it, that he sends back. This adds two rounds compared to the classical one-flow protocol in which the prover directly sends commitments plus a NIZK. But with iZKs, things are different: the prover sends commitments and an ipk, and the verifier replies with the next flow encrypted with ipk. It turns out that the prover and the verifier can agree on a batched version of the proof before even knowing the seed, so the prover can compute ipk without knowing the seed, and the verifier can just send the seed together with the masked second flow. Consequently, we can apply batch techniques to iZK-based protocols to reduce the communication without adding interactivity.

The Conversion of iZK into SSiZK is Efficient. We presented in Section 4.3 a generic construction of SSiZK from iZK. It is worth mentioning that this construct is *efficient* as it only adds a small constant number of group elements to the original iZK. Conversely, turning NIZK into simulation-sound NIZK comes at huge cost, a linear blow-up of the size of the proof. As soon as strong security requirements are considered, such as security in the UC framework, simulation-sound zero-knowledge proofs become, in the general case, unavoidable.

4.4.3.2 Setup

Let us provide some details about the iZK proofs which ensure the security of the inner product protocol described in Section 4.4. We work in a cyclic group $\mathbb{G}$ of prime order p, where the DDH assumption holds. We denote by λ the bit length of p, and by g a generator of the group. We also set the Cramer-Shoup public key to $\left(g_1, g_2, a, b, (h_i)_{i=1}^{\lambda+\ell+2}\right)$, together with a universal hash function $\mathsf{H}(\cdot)$. Since we apply the randomness-reuse technique for the Cramer-Shoup encryption, we need as many group elements h_i as the maximal size of the vector we will encrypt. The value $\lambda + \ell + 2$ is a clear upper-bound. The group description and this key (to be used for the commitment) are in the CRS.

Committing to the Secret Key. As described in Section 4.4, the client has to commit to her secret key; such a commitment adds $O(\lambda)$ to the communication complexity of the

protocol. However, the same requirement holds for any secure variant of the inner product protocol (based on the Groth-Sahai methodology or based on Σ-proofs), so, for the sake of simplicity, we omit this commitment (and the proof that it is indeed the secret key) in the protocols we are going to compare. The reason is that we focus on the setting $\ell \gg \lambda$ (for example, in the biometric setting, we can have $\lambda = 128$ while $\ell \approx 2000$) so this $O(\lambda)$ will not affect the overall comparison, even though the constants can differ from one protocol to the other.

4.4.3.3 Inner Product Protocol with iZK

Equations for the Language of the First Flow ($\mathsf{i}\mathscr{L}_\mathcal{C}$). These equations ensure that all the encrypted values are bits. the ϵ_i denote random values (used to batch the equations) which do not appear in the matrix of the SPHF associated to the iZK, so they will be picked by the server once he received the ciphertexts. $\mathsf{i}\mathscr{L}_\mathcal{C}$ is the language of words $(u_i^{\epsilon_i}, e_i^{\epsilon_i})_{i\leq\ell}$ such that there exists $((r_i, x_i)_{i\leq\ell}, \mu)$ satisfying:

1. for $i = 1$ to ℓ, $u_i^{\epsilon_i} = g^{\epsilon_i r_i}$ and $e_i^{\epsilon_i} = h^{\epsilon_i r_i} g^{\epsilon_i x_i}$
2. $1 = (\prod u_i^{\epsilon_i x_i}) \cdot g^{-\mu}$ and $1 = (\prod (e_i/g)^{\epsilon_i x_i}) \cdot h^{-\mu}$

The $2\ell + 2$ equations involve $2\ell + 1$ witnesses, $((\epsilon_i r_i)_{i\leq\ell}, (\epsilon_i x_i)_{i\leq\ell}, \mu)$. The witness μ corresponds to $\sum \epsilon_i r_i x_i$. Omitting the constants, this lead to an iZK with public key $\mathsf{ipk}_\mathcal{C}$ of size 2ℓ and ciphertext $c_\mathcal{C}$ of size 2ℓ.

Equations for the Language of the Second Flow ($\mathsf{i}\mathscr{L}_\mathcal{S}$). Let $(d_1, d_2, (e_i)_{i\leq\ell+4}, f)$ denote the Cramer-Shoup commitments of the values $((g^{y_i})_{i\leq\ell}, g^R, g^{R'}, \prod u_i^{y_i}, \prod e_i^{y_i})$, and let $(\hat{u}, \hat{e})$ denote the encryption of $R \cdot \mathsf{IP} + R'$. These equations ensure that the first ℓ committed values are bits, that the two last committed values are $\prod u_i^{y_i}$ and $\prod e_i^{y_i}$ and that $(\hat{u}, \hat{e})$ is a randomized encryption of the inner product additively and multiplicatively randomized by two committed values. The values are committed using randomness reuse techniques, which makes the commitment four times smaller but prevents us from batching our equations as we did in the first flow. $\mathsf{i}\mathscr{L}_\mathcal{S}$ is the language of words $(d_1, d_2, (e_i)_{i\leq\ell+4}, f, \hat{u}, \hat{e})$ such that there exists $((y_i)_{i\leq\ell}, (\mu_i)_{i\leq\ell+1}, r', R, R', \rho)$ satisfying:

1. $d_1 = g_1^{r'}, d_2 = g_2^{r'}, f = (ab^\xi)^{r'}$
2. for $i = 1$ to ℓ, $e_i = h_i^{r'} g^{y_i}, 1 = d_1^{y_i} g^{-\mu_i}$ and $1 = (e_i/g)^{y_i} h_i^{-\mu_i}$
3. $e_{\ell+1} = h_{\ell+1}^{r'} g^R, e_{\ell+2} = h_{\ell+2}^{r'} g^{R'}$ ans $1 = d_1^R g^{-\mu_{\ell+1}}$
4. $e_{\ell+3} = h_{\ell+3}^{r'} \prod u_i^{y_i}, e_{\ell+4} = h_{\ell+4}^{r'} \prod e_i^{y_i}$
5. $\hat{u} = g^\rho e_{\ell+3}^R h_{\ell+3}^{-\mu_{\ell+1}}, \hat{e} = h^\rho e_{\ell+4}^R h_{\ell+4}^{-\mu_{\ell+1}} g^{R'}$

The $3\ell+10$ equations involve $2\ell+5$ witnesses, $((y_i)_{i\leq\ell}, (\mu_i)_{i\leq\ell+1}, r', R, R', \rho)$. The witnesses $(\mu_i)_{i\leq\ell}$ correspond to the $r'y_i$'s and $\mu_{\ell+1}$ corresponds to $r'R$. ρ is the randomness used to randomize the ciphertext $(\hat{u}, \hat{e})$. Omitting the constants, the corresponding SSiZK has a public key $\mathsf{ipk}_\mathcal{S}$ of size 2ℓ and ciphertext $c_\mathcal{S}$ of size 2ℓ.

Communication Complexity. omitting the constants, the total communication complexity of the protocol, counting the ciphertexts, the commitments, the iZK and the SSiZK, is $2\ell + 2\ell + 4\ell + \ell + 2\ell + 3\ell = 14\ell$.

Computational Complexity. Exponentiations are required to compute the ciphertexts, the commitments, and elements of the iZK involved in the two iZKs: $(\mathsf{hp}, \mathsf{tp}, H, \mathsf{t}H, \mathsf{proj}H, \mathsf{tproj}H)$. Recall that as $(x_i, y_i)_{i \leq \ell}$ are bits, exponentiations with these values are free.

- First iZK: $2 \times 5\ell$ (for hp and tp), plus $2 \times 2\ell$ (for H and $\mathsf{tproj}H$), plus $2 \times 2\ell$ (for $\mathsf{t}H$ and $\mathsf{proj}H$), plus $2 \times 2\ell$ (for the ElGamal ciphertexts). Hence 20ℓ exponentiations in total.
- Second iZK: $2 \times 6\ell$ (for hp and tp), plus $2 \times 2\ell$ (for H and $\mathsf{tproj}H$), plus $2 \times 4\ell$ (for $\mathsf{t}H$ and $\mathsf{proj}H$), plus $2 \times \ell$ (for the commitments). Hence 25ℓ exponentiations in total.

Omitting the constants, the execution of the whole protocol requires 45ℓ exponentiations.

4.4.3.4 Inner Product Protocol with Groth-Sahai NIZKs

Unlike our iZK-based protocol, we do not intend to fully construct a UC-secure protocol for the inner product with the Groth-Sahai methodology, but rather to provide a lower bound on the complexity of such a protocol, which is enough to assess our claim that iZKs provide consistent efficiency improvement over the Groth-Sahai methodology to design UC-secure protocols whose semi-honest version does not originally involve pairings. Notice that we can apply batch techniques to reduce drastically the number of equations needed for the NIZK of the first flow, as the client cannot gain knowledge from the second flow by cheating (IP is randomized by R and R'), but the same cannot be done for the second flow because the client cannot send the decrypted value without being sure that the server was honest.

Simulation-Soundness for Groth-Sahai NIZKs. The inner product protocol involves quadratic equations and pairing product equations. While very efficient (quasi-adaptive) simulation-sound NIZKs have been designed for linear equations, to our knowledge, the best simulation-sound NIZKs for quadratic equations and pairing product equations are those of [HJ12]. However, the conversion of a NIZK into a simulation-sound NIZK with this method incurs a huge additive overhead (because of the signature) and a linear blow-up of the size of the NIZK. As the conversion of NIZK into simulation-sound NIZK involves precise computations and optimizations, we have *not* attempted to evaluate it in this section; as a consequence, all the estimations are (loose) *lower bounds* on the real complexity of a Groth-Sahai-based UC-secure inner product protocol.

Communication Complexity. To prove that all the committed values are bits, which is a quadratic equation, the x_i's have to be committed over $\mathbb{G}_1$ and $\mathbb{G}_2$, and the randomness of the ElGamal ciphertexts $(r_i)_{i \leq \ell}$ has to be committed over $\mathbb{G}_2$. These commitments and the ciphertexts represent in total 4ℓ group elements over $\mathbb{G}_1$ and 4ℓ group elements over $\mathbb{G}_2$. However, all the equations (checking that the ElGamal ciphertexts are well-formed, checking that values committed over $\mathbb{G}_1$ and $\mathbb{G}_2$ are indeed the same, checking that all the x_i are bits) can be batched. For the second flow, the server has to send commitments of the y_i's over $\mathbb{G}_1$ and $\mathbb{G}_2$, together with encryptions of the y_i's (required for the simulatability, but randomness reuse can be applied here to reduce linearly the number of group elements) and commitments over $\mathbb{G}_2$ of the randomness of the encryptions of the y_i's. Moreover, proving that the y_i are

bits involves ℓ quadratic equations, which represents 2ℓ elements over $\mathbb{G}_1$ and 2ℓ elements over $\mathbb{G}_2$. As we explained, we cannot batch those equations without adding two rounds to the protocol. The proof that the ciphertexts do indeed encrypt the committed values costs ℓ group elements over $\mathbb{G}_1$ and proving that values committed over $\mathbb{G}_1$ and $\mathbb{G}_2$ are indeed the same costs at least ℓ elements over $\mathbb{G}_1$. Thus, the second flow contains at least 7ℓ group elements over $\mathbb{G}_1$ and 6ℓ group elements over $\mathbb{G}_2$.

Total. the communication complexity of the whole execution of a UC-secure inner product protocol using the Groth-Sahai methodology is lower bounded by 11ℓ group elements over $\mathbb{G}_1$ and 10ℓ group elements over $\mathbb{G}_2$. $\mathbb{G}_2$ being approximately twice as big as $\mathbb{G}_1$ with usual settings, this represents roughly 31ℓ elements over $\mathbb{G}_1$, which is 50% more than the iZK-based protocol.

Pairings and Exponentiations. Counting the number of exponentiations of Groth-Sahai proofs is quite involved, as this number is quadratic $O(\ell^2)$ in the general case, but linear in nearly every specific application, if the correct optimizations are used. Instead of counting the exponentiations, we focus on a loose lower bound by counting only the exponentiations required to compute ciphertexts and commitments, without even considering the computations required for the construction and the verification of the proofs. this leads to a lower bound of 28ℓ exponentiations over $\mathbb{G}_2$ ans 6ℓ exponentiations over $\mathbb{G}_1$. Moreover, several paper have lowered the number of pairing needed to verify the proofs; even if we consider that the verification of all the proofs can be batched into a single verification of a pairing-product equation, using the optimizations of [BFI+10], at least 4ℓ pairings are required for the first flow. For the second flow, which cannot be batched, verification (using [BFI+10]) of one pairing-product equation, two multi-scalar multiplication equations and one quadratic equation is lower bounded by $(4+2+2+2)\ell = 10\ell$ pairings. The overall number of pairings is thus lower-bounded by 14ℓ. As we can choose more efficient curves, with fast exponentiations, by avoiding the need of pairings, even these very loose values represent considerably more computations that the exponentiations required by the iZK-based protocol.

4.4.3.5 Inner Product Protocol with Schnorr Σ-Protocols

Let us now provide an estimation of the cost of an UC-secure protocol for the inner product relying on Σ-Protocols (i.e., protocols with a three-move structure, namely (*commitments*, *challenge*, *response*)). There are two ways of designing such a protocol:

1. One can rely on the OR trick to prove, for each ciphertext (u, e), that either (u, e) or (u, ge^{-1}) is an encryption of 0 (a DDH tuple).

2. Alternatively, one can commit to $(x_i^2)_{i\leq\ell}$, prove that the commitments contains the square of the encrypted values (using a Chaum-Pedersen proof of same discrete logarithm with different bases), and then batch all the proofs by proving a statement of the form $\sum_{i=1}^{\ell} \lambda_i(x_i - x_i^2) = 0$, for a random tuple of values $(\lambda_i)_{i\leq\ell}$ chosen by the verifier after the prover has committed.

We will focus on the second technique for our estimation; both techniques seem roughly equivalent in terms of communication and computation. The commitment scheme used in this protocol is the Pedersen commitment scheme, which can be seen as the second part of an ElGamal ciphertext: $c(m; r) = h^r g^m$. The reader might refer to [Mau09] to get an

intuition of the cost of the different proofs we are going to construct, as all our proofs can be seen as proving the knowledge of a preimage of a group homomorphism, which fits into the framework of [Mau09]. Moreover, all those proofs can be turned into simulation-sound ZK proofs at a small, constant additive cost, using the generic transformation of [GMY03]. The protocol goes as follow: (we omit the constants when we provide the number of elements exchanged)

Protocol.

1. The client sends ℓ ElGamal ciphertexts $(u_i, e_i)_{i\leq\ell}$ and ℓ commitments $(w_i)_{i\leq\ell}$ of the squares of the encrypted values. He also generates 3ℓ randomness for the proof, hash them using a collision-resistant hash function, and commits to this value.

2. The server replies with a challenge c, ℓ ElGamal ciphertexts of his own values (required for the simulatability) plus the randomness (R, R') (with his key), ℓ commitments of the squares of his values and an encryption (with the client key) of $(R \cdot \mathsf{IP} + R')$.

3. The client sends a proof, which contains 3ℓ scalars and 3ℓ openings of the randomness whose hash value he committed to in the first flow. he also sends a challenge c'.

4. The server checks that the openings are correct, and if they are, that the proofs hold, i.e.that the values where indeed bits and that $(\lambda_i)_{i\leq\ell}$, the values λ_i being computed from the challenge c with a pseudo-random generator. Then, he sends himself a similar proof, ensuring his values are bits ($3\ell + 3\ell$ elements), plus a proof that the randomized scalar product was correctly computed (2ℓ elements).

5. If the openings and the proofs are correct, the client sends the decrypted randomized inner product to the server.

For details on how Σ-protocols can be built for statements such as "I know openings of commitments such that one of them opens to the product of the two other committed values", the reader might refer to [Mau09]. We enhance the security of the original Σ-protocols by adding commitments to the randomness and revealing the openings after receiving the challenge; such enhanced protocols can be proven secure against malicious verifiers, and so are truly zero-knowledge.

Efficiency. The communication complexity can be easily counted from our description of the protocol: $(2 + 1 + 2 + 1 + 3 + 3 + 3 + 3 + 2)\ell = 20\ell$. The computational complexity, counted as a number of exponentiations and omitting constant values and other operations, is 38ℓ:

- $2\ell + \ell$ for the ciphertexts and Pedersen commitments of the first flow.
- $3\ell + 3\ell$ for the random ciphertexts and Pedersen commitments hashed and committed in the first flow.
- $2\ell + \ell$ for the ciphertexts and Pedersen commitments of the second flow.
- $3\ell + 3\ell$ for the random ciphertexts and Pedersen commitments hashed and committed in the second flow.

- $3\ell + 2\ell$ to check the opening of the random ciphertexts and Pedersen commitments hashed and committed in the first flow.
- $3\ell + 2\ell$ to check the proofs (3ℓ for the commitments of squares of encrypted values, 2ℓ for the batched proof of bit values)
- $3\ell + 2\ell$ to check the opening of the random ciphertexts and Pedersen commitments hashed and committed in the second flow.
- $3\ell + 2\ell$ to check the proofs (3ℓ for the commitments of squares of encrypted values, 2ℓ for the batched proof of bit values).
- 2ℓ to check the proof that the inner product was correctly computed.

4.4.4 Proof of Security

In this section, we prove that (the malicious version of) our scheme in Section 4.4 is secure in the UC model [Can00], with authenticated channels and static corruptions.

Details on the Scheme. Here are some implicit details related to UC for the scheme in Section 4.4: all flows contains an identifier (1 for the first flow, 2 for the second flow and 3 for the third flow). Every flow not formatted correctly is ignored. Every commitment is supposed to be labeled with an identifier of the commitment (1 for the one of the first flow and 2 for the one of the second flow), the identifier of $\mathcal{C}$ and $\mathcal{S}$, the session and sub-session identifiers *sid* and ssid. We use the labeled version of the Cramer-Shoup encryption scheme [CHK+05] for that purpose. We recall that this scheme is IND-CCA secure.

Ideal Functionality. The ideal functionality is depicted in Figure 4.8. Basically, the client $\mathcal{C}$ sends its input $(x_i)_{i=1}^{\ell} \in \{0,1\}^{\ell}$, then the server sends its input $(y_1)_{i=1}^{\ell}$, and finally, when the adversary or simulator $\mathcal{S}im$ specifies it, the server gets back the inner product IP of (x_i) and (y_i). Corruptions of the client or the server are supposed to be static, i.e., before the first message `Client-Send` is sent (for a given session $(sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$). The authentication and the flow identifiers above ensure that if one of the player is honest at the beginning, he remains honest during all the session and the adversary in the real world cannot modify his flow (though he may drop them as usual and attempt a denial-of-service attack).

Proof. We exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}im$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

Game $\mathbf{G}_0$: This is the real game, where the simulator knows the inputs of all the honest players and honestly play their role (on their behalf).

Game $\mathbf{G}_1$: We first deal with the case when $\mathcal{C}$ *and* $\mathcal{S}$ *are both honest.* In that case, the simulator $\mathcal{S}im$ replaces all commitments and ciphertexts of $\mathcal{C}$ and $\mathcal{S}$ by commitments and ciphertexts of random values. In addition, except if the adversary $\mathcal{A}$ drops some flows, the simulator $\mathcal{S}im$ never abort on behalf of $\mathcal{S}$ and outputs the correct inner product $\mathsf{IP} = \sum_{i=1}^{\ell} x_i \cdot y_i$ he can compute since he still knows the inputs (x_i) of $\mathcal{C}$ and (y_i) of $\mathcal{S}$. $\mathcal{S}im$ also sends to the message `Result-Send` when required. This game is

The functionality $\mathcal{F}_{\mathsf{IP}}$ is parametrized by a security parameter k. It interacts with an adversary $\mathcal{S}im$ and a set of parties via the following queries:

Upon receiving a query $(\texttt{Client-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i)_{i=1}^{\ell})$ from party $\mathcal{C}$ (client): Ignore the message if $(x_i)_{i=1}^{\ell} \notin \{0,1\}^{\ell}$. Record the tuple $(sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i))$ and send $(\texttt{Client-Sent}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$ to $\mathcal{S}im$. Ignore further `Client-Send`-message with the same $(\mathsf{ssid}, \mathcal{C}, \mathcal{S})$ from $\mathcal{C}$.

Upon receiving a query $(\texttt{Server-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (y_i)_{i=1}^{\ell})$ from party $\mathcal{S}$ (client): Ignore the message if $(y_i)_{i=1}^{\ell} \notin \{0,1\}^{\ell}$. Ignore the message if $(sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i))$ is not recorded (for some (x_i)) and replace this record by $(sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i), (y_i))$; otherwise mark the record as used and send $(\texttt{Server-Sent}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$ to $\mathcal{S}im$. Ignore further `Server-Send`-message with the same $(\mathsf{ssid}, \mathcal{C}, \mathcal{S})$ from $\mathcal{S}$.

Upon receiving a query $(\texttt{Result-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$ from the adversary $\mathcal{S}im$: ignore the message if $(sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i), (y_i))$ is not recorded (for some (x_i) and (y_i)); otherwise remove the record and send $(\texttt{Result-Sent}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, \mathsf{IP})$ to $\mathcal{S}$ (and to $\mathcal{S}im$ if $\mathcal{S}$ is corrupted), with $\mathsf{IP} = \sum_{i=1}^{\ell} x_i \cdot y_i$. Ignore further `Result-Send`-message with the same $(\mathsf{ssid}, \mathcal{C}, \mathcal{S})$ from $\mathcal{S}im$.

Figure 4.8: Ideal Functionality for Inner Product $\mathcal{F}_{\mathsf{IP}}$

indistinguishable from the previous one under the IND-CPA property of the encryption scheme and the commitment scheme.

We remark that now, we do not need to know the exact inputs of honest players.

Game $\mathbf{G}_2$: We now deal with sessions between *a malicious client $\mathcal{C}$ and a honest server $\mathcal{S}$*. $\mathcal{S}im$ first extracts the commitment of the bits of the secret keys sk_j, and recovers sk. If these commitments do not contains bit or if these bits do not correspond to a valid secret key sk (i.e., such that the sent public key $\mathsf{pk} = g^{\mathsf{sk}}$), then $\mathcal{S}im$ chooses $K_{\mathcal{C}}$ uniformly at random. Otherwise, $\mathcal{S}im$ uses this secret key to decrypt the ciphertexts c_i for $i = 1, \ldots, \ell$, and get bits x_i. If the corresponding plaintexts are not bits, then $\mathcal{S}im$ chooses $K_{\mathcal{C}}$ uniformly at random. This game is statistically indistinguishable from the previous one, thanks to the soundness of the iZK.

Game $\mathbf{G}_3$: We now replace the CRS of the two iZK (which were generated by iSetup) by a CRS generated by TSetup and we remember the corresponding trapdoors $\mathsf{i}\mathcal{T}$. This game is computationally indistinguishable from the previous one, thanks to the setup indistinguishability of the iZK.

Game $\mathbf{G}_4$: We now simulate all the iZK using iTKG and iTDec made by the simulator. This game is statistically indistinguishable from the previous one, thanks to the zero-knowledge property of the iZK.

Game $\mathbf{G}_5$: We now deal again with sessions between *a malicious client $\mathcal{C}$ and a honest server $\mathcal{S}$* in this game and the following ones. We replace the commitment of g^{y_i}, g^R and $g^{R'}$ by commitments of random values. This game is computationally indistinguishable from the previous one, thanks to the IND-CCA property of the Cramer-Shoup encryption scheme used for the commitment (and the fact that extractions are always done with

different labels), and the fact that the random coins used by these commitments are no more necessary to decrypt the iZK ciphertexts $c_{\mathcal{C}}$ (thanks to the previous game).

Game $\mathbf{G}_6$: *Sim* directly generates c as an encryption of $g^{R\mathsf{IP}+R'}$, by computing IP as $\sum_{i=1}^{\ell} x_i \cdot y_i$ (x_i being extracted by the encryption c_i and y_i being given as inputs to $\mathcal{S}$). This is perfectly indistinguishable to the previous game.

Game $\mathbf{G}_7$: *Sim* now aborts on behalf of $\mathcal{S}$ if the last flow if not $g^{R\mathsf{IP}+R'}$ instead of just aborting when it is not such that $g^{R\mathsf{IP}'+R'}$ with $\mathsf{IP}' \in \{0, \ldots, \ell\}$. In addition, if $\mathcal{S}$ does not abort, instead of computing IP from the last flow (and so potentially getting IP'), *Sim* directly outputs IP. We remark that for any IP and any fixed value for u, for any value of the last flow different than $g^{R\mathsf{IP}+R'}$, the probability this flow is $g^{R\mathsf{IP}'+R'}$ with $\mathsf{IP}' \in \{0, \ldots, \ell\}$ (so with $\mathsf{IP} \neq \mathsf{IP}'$), when R and R' are chosen uniformly at random conditioned by $u = R\mathsf{IP} + R'$, is at most ℓ/p, which is negligible. Since the adversary $\mathscr{A}$ does not know R and R' but only $R\mathsf{IP} + R'$, this game is statistically indistinguishable from the previous one.

Game $\mathbf{G}_8$: *Sim* now generates c as an encryption of g^S for a random S and aborts when the last flow is not g^S. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_9$: *Sim* now sends $(\texttt{Client-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (x_i))$ in behalf of $\mathcal{C}$ to the ideal functionality with (x_i) the extracted values of the malicious client $\mathcal{C}$. If $\mathcal{S}$ does not abort, *Sim* also sends $(\texttt{Result-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$ to the ideal functionality. In addition *Sim* let the ideal functionality generate the output for $\mathcal{S}$. This game is perfectly indistinguishable from the previous one, since both will output the same value IP.

Game $\mathbf{G}_{10}$: We now deal again with sessions between *a honest client $\mathcal{C}$ and a malicious server $\mathcal{S}$* in this game and the following ones. *Sim* now returns $g^{R\mathsf{IP}+R'}$ in the last flow (if $\mathcal{C}$ did not abort) instead of decrypting c. This game is perfectly indistinguishable from the previous one.

Game $\mathbf{G}_{11}$: In this game, we now replace the commitments of sk_j by commitments of random values. This game is computationally indistinguishable from the previous one under the IND-CCA property of the commitment scheme. We remark that we do not use anymore sk.

Game $\mathbf{G}_{12}$: In this game, *Sim* now encrypts random values in c_i instead of the x_i's. This game is computationally indistinguishable from the previous one under the IND-CPA property of the commitment scheme. We remark that we do not use anymore the x_i's.

Game $\mathbf{G}_{13}$: *Sim* now extracts the commitment of the bits y_i, together with g^R and $g^{R'}$. If y_i are not bits or if the ciphertext c received by $\mathcal{S}$ (under pk for which *Sim* knows the secret key sk) does not contain $g^{R\mathsf{IP}+R'}$ (with $\mathsf{IP} = \sum_{i=1}^{\ell} x_i \cdot y_i$, with the extracted y_i's and the x_i given as inputs to $\mathcal{C}$), then *Sim* chooses $K_{\mathcal{S}}$ uniformly at random. This game is statistically indistinguishable from the previous one, thanks to the *simulation-soundness* of the second iZK.

Game $\mathbf{G}_{14}$: *Sim* now sends $(\texttt{Server-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S}, (y_i))$ in behalf of $\mathcal{S}$ to the ideal functionality with (y_i) the extracted values of the malicious server $\mathcal{S}$. If $\mathcal{C}$ does not abort, *Sim* also sends $(\texttt{Result-Send}, sid, \mathsf{ssid}, \mathcal{C}, \mathcal{S})$ to the ideal functionality, and get

the value of IP. This game is perfectly indistinguishable from the previous one, since the computed value of IP (as $\sum_{i=1}^{\ell} x_i \cdot y_i$ with (x_i) the input of $\mathcal{C}$ and (y_i) extracted from $\mathcal{S}$) is always equal to the value IP returned by the functionality.

Game $\mathbf{G}_{15}$: In last game, $\mathcal{S}im$ does not use anymore the inputs given to the honest parties. So this game is exactly the game in the ideal world.

5

Zero-Knowledge Arguments over the Integers under the RSA Assumption

5.1 Introduction

In this chapter, we turn our attention to the security of interactive zero-knowledge argument systems. Specifically, we consider *zero-knowledge arguments over the integers*, a useful type of interactive zero-knowledge arguments that allows to directly and efficiently manipulate statements over the integers. For preliminaries on zero-knowledge arguments over the integers, we refer the reader to Section 3.7, but in short, these arguments consist in proving relations between values committed with *integer commitment schemes*. The first integer commitment scheme was introduced by Okamoto and Fujisaki [FO97], and was later generalized in [DF02]. Unlike classical commitment schemes, an integer commitment scheme allows to commit to any $m \in \mathbb{Z}$. Intuitively, this is done by committing to m in a group $\mathbb{Z}_\tau$ of unknown order, where division by units cannot be performed in general. Integer zero-knowledge arguments have turned out to be particularly suited to efficiently handle statements that can be expressed as Diophantine relations. A typical case of such statements are range arguments, where the goal is to show that some committed value belongs to some specific interval. Such range arguments have found many applications in various cryptographic primitives.

Assumptions for Proofs on Integer Commitments. Unfortunately, we do not have a wide variety of integer commitment schemes at our disposal. In fact, the Damgård-Fujisaki commitment scheme is essentially the only known integer commitment scheme to be both *compact* (a commitment to a message $m \in \mathbb{Z}$ of arbitrary size has a size independent of $|m|$) and additively homomorphic over $\mathbb{Z}$. Nonetheless, the security properties of this scheme are well understood: the binding property of the Damgård-Fujisaki commitment scheme relies on the hardness of factoring composite integers, and its hiding property holds statistically. However, even though the intractability of factoring is widely considered as a mild computational assumption, when the Damgård-Fujisaki commitment scheme is used as a component for zero-knowledge arguments over the integers, the state of affairs is less satisfying. Indeed, the knowledge-extractability of zero-knowledge arguments using the Damgård-Fujisaki commitment scheme relies on the Strong-RSA assumption [BP97; FO97], which is a much stronger assumption. We refer the reader to Section 2.2.2.3 for definitions and discussions related to the Strong-RSA assumption, but in short, this assumption states that, given a composite integer n and a random element $u \in \mathbb{Z}_n^*$, it is hard to find a pair (v, e) such that $u = v^e \bmod n$. Unlike the RSA assumption [RSA78b], where the exponent $e > 1$ is imposed, there are exponentially many solutions to a given instance of the Strong-RSA problem, which makes it less desirable. This implies, in turn, that all cryptographic applications relying on zero-knowledge arguments over the integers of any kind need to assume the Strong-RSA assumption.

Range Proof. As discussed in Section 3.7, the most widespread reason to work over the integers is to prove that a committed value x lies in a public integer range $[\![a\,;b]\!]$. Indeed, working over the integers allows to show that $x - a$ and $b - x$ are positive by demonstrating that they can be decomposed as a sum of four squares, following the well-known Lagrange's result. Lipmaa [Lip03] was the first to propose such a method by relying on a commitment over the integers. As a consequence, the knowledge extractability of his range proof requires the Strong-RSA assumption.

Applications of Range Proofs. Range proofs have found numerous applications in cryptography. They have been used in the context of privacy-preserving data mining

and statistical analysis [GLLM05; GMS10], voting [Gro05], anonymous credentials [CG08], secure generation of RSA keys [JG02; DM10; HMRT12], zero-knowledge watermarking detection [ARS04], sanitizable signatures [YSLM08], private information retrieval [KLL+15], and multiparty computation [CPP15a], among others. As a consequence, the security analysis of efficient instantiations of these applications, relying on Lipmaa's range proof over the integers, must assume the Strong-RSA assumption.

5.1.1 Our Contribution

In this chapter, we revisit the Damgård-Fujisaki integer commitment scheme and show that the security of arguments of knowledge of openings can be based on the standard RSA assumption, instead of the Strong-RSA assumption. In the reduction, we use the rewinding technique in another way than in [DF02] as well as the splitting lemma [PS96; PS00]. Our result extends to any protocols involving argument or relation between committed integers which first prove the knowledge of the inputs before proving that the relation is satisfied. This implies that the security of numerous protocols, such as two-party computation [JS07; CPP15a], e-cash [CHL05], e-voting [LAN01; Gro05], secure generation of RSA keys [JG02; DM10; HMRT12], zero-knowledge primality tests [CM99a], password-protected secret sharing [JKK14], and range proofs [Lip03], among many others, can be proven under the RSA assumption instead of the Strong-RSA assumption for free. In addition, we believe that the ideas on which our proof relies could be used in several other constructions whose security was proven under the Strong-RSA assumption, and might allow to replace the Strong-RSA assumption by the standard RSA assumption in such constructions.

5.1.2 Related Works

The Damgård-Fujisaki commitment scheme [FO97; DF02] is the only known homomorphic statistically-hiding commitment scheme over the integers. Arguments of knowledge over the integers were studied in [Lip03; KTY04; CCT07].

Range proofs were introduced in [BCDv88]. They are a core component in numerous cryptographic protocols, including e-cash [CHL05], e-voting [Gro05], private auctions [LAN01], group signatures [CM99b], and anonymous credentials [CL01], among many others. There are two classical methods for performing a range proof:

- Writing the number in binary notation [BCDv88; Gro11] or u-ary notation [CCs08], committing to its decomposition and performing a specific proof for each of these commitments For example, membership to $[\![0\,;2^\ell]\!]$ is proven in communication $O(\ell/(\log \ell - \log\log \ell))$ in the protocol of[CCs08], and in communication $O(\ell^{1/3})$ in the protocol of [Gro11] (counting the number of group elements). Both rely on pairing-based assumptions, namely, the computational double pairing assumption (which is implied by the decisional Diffie-Hellman assumption) and the q-Strong Diffie-Hellman assumption [BB04].

- Using an integer commitment scheme [Bou00; Lip03; Gro05].

Note that protocols such as [CFT98] do also allow to prove that a committed integer x lies in a given interval $[\![0\,;a]\!]$, but not *exactly*: the proof shows only membership to $[\![0\,;(1+\delta)a]\!]$ for some accuracy parameter $\delta > 0$.

Eventually, several papers have proposed signatures based on the standard RSA assumption [HW09; HJK11; BHJ+13] as alternatives to classical signature schemes based on the Strong-RSA assumption. Our work is in the same vein than these papers, replacing the Strong-RSA assumption by the RSA assumption in arguments over the integers. However, note that we do not actually propose a new argument system to get rid of the Strong-RSA assumption, but rather show that the security of the classical argument system is implied by the RSA assumption. As a consequence, the schemes using arguments over the integers do not need to be modified to benefit from our security analysis.

5.1.3 Organization

Section 5.2 recalls the Damgård-Fujisaki commitment scheme, its properties, and the argument of knowledge of [DF02]. A new security proof of the latter, under the standard RSA assumption, is given in details Section 5.3. Section 5.4 illustrates some extensions of our result. First, we show how one can commit to vectors at once with generalized commitments. And then, we show how one can make range proofs under the standard RSA assumption.

For the sake of completeness, in Section 5.4.4 we exhibit a flaw in the optimized version of Lipmaa's range proof [Lip03, Annex B]. We then propose a fix and prove it.

5.2 Commitment of Integers Revisited

In [FO97], Okamoto and Fujisaki proposed a statistically-hiding commitment scheme allowing commitment to arbitrary-size integers. Their commitment was later generalized in [DF02]. It relies on the fact that when the factorization is unknown, it is infeasible to know the order of the sub-group QR_n of the squares in $\mathbb{Z}_n^*$, where n is a strong RSA modulus. Hence, the only way for a computationally-bounded committer to open a commitment is to do it over the integers.

In addition, [FO97] gave an argument of knowledge of an opening of a commitment and proved that the knowledge extractability of the argument is implied by the Strong-RSA assumption. A flaw in the original proof was later identified and corrected in [DF02]. We will revisit the argument of knowledge of an opening due to Damgård-Fujisaki [DF02] and provide a new proof for its knowledge extractability, in order to remove the requirement of the Strong-RSA assumption. Our proof requires the standard RSA assumption only, with an exponent randomly chosen in a polynomially-bounded set.

5.2.1 Commitments over the Integers

Description. Let us recall the commitment of one integer m (for the definition of the algorithm GenMod, refer to Section 2.2.2):

- $\mathsf{Setup}(1^\kappa)$ runs $(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$, and picks two random generators g, h of QR_n. It returns $\mathsf{pp} = (n, g, h)$;
- $\mathsf{Commit}(\mathsf{pp}, m; r)$, for $\mathsf{pp} = (n, g, h)$, a message $m \in \mathbb{Z}$, and some random coins $r \xleftarrow{\$} [\![0\,; n/4]\!]$, computes $c = g^m h^r \bmod n$, and returns (c, d) with $d = r$;
- $\mathsf{Verify}(\mathsf{pp}, c, d, m)$ parses pp as $\mathsf{pp} = (n, g, h)$ and outputs 1 if $c = \pm g^m h^d \bmod n$ and 0 otherwise.

One should note that an honest user will always open such that $c = g^m h^d \bmod n$. But the binding property cannot exclude the change of sign. In this description, we provide a trusted setup algorithm. But as we see below, the guarantees for the prover (the hiding property of the commitment) just rely on the existence of α such that $g = h^\alpha \bmod n$. For the verifier to be convinced, one can just let him generate the parameters (n, g, h), and prove the existence of such an α to the prover.

Security Analysis. The above commitment scheme is obviously *correct*. The *hiding* property relies on the existence of α such that $g = h^\alpha \bmod n$ (they are both generators of the same subgroup QR_n), and so, for any $m' \in \mathbb{Z}$,

$$\begin{aligned} c = \mathsf{Commit}(\mathsf{pp}, m; r) &= g^m h^r = h^{r+\alpha m} = h^{(r+\alpha(m-m'))+\alpha m'} \\ &= g^{m'} h^{r+\alpha(m-m')} = \mathsf{Commit}(\mathsf{pp}, m'; r'), \end{aligned}$$

with $r' \leftarrow [r + \alpha(m - m') \bmod p'q']$, that is smaller than n and follows the same distribution as r (statistically), as $p'q' \approx n/4$. The binding property relies on the Integer Factorization assumption: indeed, from two different openings m_0, d_0, m_1, d_1 for a commitment c, with $d_1 > d_0$, the validity checks show that $g^{m_0} h^{d_0} = \pm g^{m_1} h^{d_1} \bmod n$, and so $g^{m_0 - m_1} = \pm h^{d_1 - d_0} \bmod n$. Since g and h are squares, and -1 is not a square, necessarily $g^{m_0 - m_1} = h^{d_1 - d_0} \bmod n$. The Fact 2 from Proposition 2.2.7 leads to a non-trivial factor of n.

5.2.2 Zero-Knowledge Argument of Opening

Let us now study the argument of knowledge of a valid opening for such a commitment. The common inputs are the public parameters pp and the commitment $c = g^x h^r \bmod n$, together with the bit-length k_x of the message x, that is then assumed to be in $[\![-2^{k_x}\,; 2^{k_x}]\!]$, while $r \in [\![0\,; n]\!]$ and x are the private inputs, i.e.the witness of the prover. We stress that k_x is chosen by the prover, since this reveals some information about the integer x, while r is always in the same set, whatever the committed element x is.

Description of the Protocol. The protocol works as follows:

Initialize: $\mathcal{P}$ and $\mathcal{V}$ decide to run the protocol on input $(\mathsf{pp}, \kappa, c, k_x)$;

Commit: $\mathcal{P}$ computes $d = g^y h^s \bmod n$, for randomly chosen $y \xleftarrow{\$} [\![0\,; 2^{k_x+2\kappa}]\!]$ and $s \xleftarrow{\$} [\![0\,; 2^{|n|+2\kappa}]\!]$, and sends d to the $\mathcal{V}$;

Challenge: $\mathcal{V}$ outputs $e \xleftarrow{\$} [\![0\,; 2^{\kappa}]\!]$;

Response: $\mathcal{P}$ computes and outputs the integers $z = ex + y$ and $t = er + s$;

Verify: $\mathcal{V}$ accepts the proof and outputs 1 if $c^e d = g^z h^t \bmod n$. Otherwise, $\mathcal{V}$ rejects the proof and outputs 0.

In the rest of this section, we prove this protocol is indeed a zero-knowledge argument of knowledge of an opening. Which means it is correct, zero-knowledge, and knowledge-extractable.

Correctness. First, the correctness is quite obvious: if $c = g^x h^r \bmod n$, with $z = ex + y$ and $t = er + s$, we have $g^z h^t = (g^x h^r)^e \cdot g^y h^s = c^e d \bmod n$.

Zero-Knowledge. For the zero-knowledge property, in the honest-verifier setting, the simulator $\mathcal{S}im$ (that is $\mathcal{S}im_{\mathsf{ZK}}$ in this case) can simply do as follows:

$\mathcal{D}_0$	$y \xleftarrow{\$} [\![0\,;2^{k_x+2\kappa}]\!], s \xleftarrow{\$} [\![0\,;2^{\lvert n\rvert+2\kappa}]\!],$ $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!], z = xe + y, t = re + y, d = g^y h^s \bmod n$
$\mathcal{D}_1$	$z \xleftarrow{\$} [\![xe\,;2^{k_x+2\kappa} + xe]\!], t \xleftarrow{\$} [\![re\,;2^{\lvert n\rvert+2\kappa} + re]\!],$ $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!], d = g^{z-xe} h^{t-re} \bmod n$
$\mathcal{D}_2$	$z \xleftarrow{\$} [\![xe\,;2^{k_x+2\kappa} + xe]\!], t \xleftarrow{\$} [\![re\,;2^{\lvert n\rvert+2\kappa} + re]\!],$ $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!], d = g^{z} h^{t} c^{-e} \bmod n$
$\mathcal{D}_3$	$z \xleftarrow{\$} [\![0\,;2^{k_x+2\kappa}]\!], t \xleftarrow{\$} [\![0\,;2^{\lvert n\rvert+2\kappa}]\!],$ $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!], d = g^{z} h^{t} c^{-e} \bmod n$

Figure 5.1: Distributions for the Zero-Knowledge Property

1. $\mathcal{S}im$ chooses a random challenge $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!]$;
2. $\mathcal{S}im$ chooses random responses $z \xleftarrow{\$} [\![0\,;2^{k_x+2\kappa}]\!]$ and $t \xleftarrow{\$} [\![0\,;2^{\lvert n\rvert+2\kappa}]\!]$;
3. $\mathcal{S}im$ sets $d = g^z h^t c^{-e} \bmod n$.

The simulated transcript is the tuple $(d, e, (z, t))$, where the elements follow the distribution $\mathcal{D}_3$ from Figure 5.1, while the real transcript follows the distribution $\mathcal{D}_0$.

However, it is clear that $\mathcal{D}_0 = \mathcal{D}_1 = \mathcal{D}_2$, while the distance between $\mathcal{D}_2$ and $\mathcal{D}_3$ is the sum of the distances between the distributions of z and t, respectively in $\mathcal{Z}_2 = [\![xe\,;2^{k_x+2\kappa} + xe]\!]$ and $\mathcal{Z}_3 = [\![0\,;2^{k_x+2\kappa}]\!]$, and $\mathcal{T}_2 = [\![re\,;2^{\lvert n\rvert+2\kappa} + re]\!]$ and $\mathcal{T}_3 = [\![0\,;2^{\lvert n\rvert+2\kappa}]\!]$:

$$\begin{aligned}\Delta_z &= \sum_{Z=0}^{2^{k_x+2\kappa}+xe} \left|\Pr[z \xleftarrow{\$} \mathcal{Z}_2 : z = Z] - \Pr[z \xleftarrow{\$} \mathcal{Z}_3 : z = Z]\right| \\ &= \sum_{Z=0}^{xe-1} 2^{-k_x-2\kappa} + \sum_{Z=2^{k_x+2\kappa}+1}^{2^{k_x+2\kappa}+xe} 2^{-k_x-2\kappa} = 2 \cdot xe \cdot 2^{-k_x-2\kappa} \leq 2 \cdot 2^{k_x+\kappa} \cdot 2^{-k_x-2\kappa}\end{aligned}$$

that is bounded by $2 \cdot 2^{-\kappa}$. Similarly, $\Delta_t \leq 2 \cdot 2^{-\kappa}$. Hence the statistical zero-knowledge property, since the real distribution $\mathcal{D}_0$ and the simulated distribution $\mathcal{D}_3$ have a negligible distance bounded by $2^{-\kappa+2}$.

Knowledge-Extractability. The last property is the most intricate, and this is the one that required the Strong-RSA assumption in the original proof of Damgård and Fujisaki [DF02]. In the next section, we will prove the following theorem:

Theorem 5.2.1. *Given a prover $\mathcal{P}$' able to convince a verifier $\mathcal{V}$ of its knowledge of an opening of c for random system parameters $\mathsf{pp} = (n, g, h)$ with probability greater than ε within time t, one either breaks the RSA assumption with expected time upper-bounded by $256t/\varepsilon^3$, or extracts a valid opening with expected time upper-bounded by $16t/\varepsilon^2$.*

Let us first provide an intuition of our proof, starting from the original proof of Damgård and Fujisaki, and explaining our refinements. The starting point of Damgård and Fujisaki is the natural approach to prove the knowledge of extractability of Schnorr-like protocols: the simulator will run the malicious prover twice, and get (with non-negligible probabilities) two accepting transcripts corresponding to the same first flow (commit phase). Writing (z_0, t_0)

and (z_1, t_1) those two accepting transcripts, corresponding to two challenges (e_0, e_1) sent by the simulator, as well as the *same* first flow d, the simulator obtains:

$$c^{e_0} d = g^{z_0} h^{t_0} \bmod n \qquad\qquad c^{e_1} d = g^{z_1} h^{t_1} \bmod n$$

Dividing the two equations to cancel the d terms gives

$$c^{e_0 - e_1} = g^{z_0 - z_1} h^{t_0 - t_1} \bmod n$$

If the protocol had been executed over a group of known order, this would be essentially over, as $((z_0 - z_1)(e_0 - e_1)^{-1} \bmod \varphi(n), (t_0 - t_1)(e_0 - e_1)^{-1} \bmod \varphi(n))$ would form a valid opening to c. However, the issue here is that we cannot compute $(e_0 - e_1)^{-1} \bmod \varphi(n)$ without knowing the factorization of n. In their proof, Damgård and Fujisaki are therefore left with distinguishing two cases.

1. If $(e_0 - e_1)$ divides both $(z_0 - z_1)$ and $(t_0 - t_1)$, we are essentially fine: we can compute an opening to c as in the Schnorr protocol. In practice, things are slightly more complicated, but it can be shown that either a valid opening will be computed, or a non-trivial square root of 1 will be obtained, which would break the factorization assumption.

2. If $(e_0 - e_1)$ does not divide either $(z_0 - z_1)$ or $(t_0 - t_1)$, then a more careful analysis is required.

For the second case above, Damgård and Fujisaki start by observing that when this happens, then with probability at least $1/2$, it must hold that $(e_0 - e_1)$ does not divide $\alpha(z_0 - z_1) + (t_0 - t_1)$ (recall that $g = h^\alpha \bmod n$). In essence, this stems from the fact that h^α only leaks $\alpha \bmod \varphi(n)$ to an unbounded adversary; by picking α to be large enough in the setup, the verifier can therefore ensure that a large part of α remains information-theoretically hidden. Building on this observation, Damgård and Fujisaki develop an information-theoretic argument to conclude that $(e_0 - e_1)$ does not divide $\alpha(z_0 - z_1) + (t_0 - t_1)$, with non-negligible probability.

From this point, they invoke a method commonly known as 'Shamir's gcd-in-the-exponent trick' which exploits the Bézout relation to find a non-trivial root of h, for some exponent x equal to $\gcd(e_0 - e_1, \alpha(z_0 - z_1) + (t_0 - t_1))$. This gives a solution to a Strong-RSA challenge with exponent x, and the knowledge-extractability of the scheme therefore follows, under this assumption.

At first sight, it is not obvious that this result can be improved: the exponent x clearly depends on the answers (z_0, z_1, t_0, t_1) of the prover, which has therefore some control over it, hence it is unlikely that the simulator could force him to solve an RSA challenge, whose exponent is sampled before the protocol and not chosen by the prover. Our solution proceeds as follow: we will show that, even though the prover has some freedom upon choosing the exponent x, he must choose x to be *small*. More precisely, we show that x must be inversely related to the success probability of the adversary. Therefore, if the adversary has non-negligible success probability, then x is of polynomial size.

Then, we essentially *guess* the small exponent that the adversary will choose: we rely on a variant of the RSA assumption where the exponent is drawn uniformly at random from a set of polynomial size. The simulator ensures that no information leaks about the exponent that was drawn when interacting with the prover; hence, when extracting an exponent x from this

prover, it must be that this exponent has non-negligible probability of being the exponent initially drawn, which allows to break the RSA assumption with non-negligible probability.

It remains to explain why the exponent x is inversely related to the success probability of the prover. To show this, we consider a simulator that executes a *third rewind*, getting a third successful answer from the prover with non-negligible probability. With the two first transcripts, as we saw previously, the simulator had obtained a relation of the form $c^e = g^z h^t \bmod n$ (with $e = e_0 - e_1$, $z = z_0 - z_1$, and $t = t_0 - t_1$). Similarly, with the first and the last transcript, the simulator gets a relation of the form $c^{e'} = g^{z'} h^{t'} \bmod n$. Raising the first relation to the power of e, the second to e', and dividing the two relations gives

$$g^{e'z - ez'} = h^{et' - e't}$$

Now, we know that if we can find two integers (a, b) such that $g^a = h^b$, we can solve the factorization (see Proposition 2.2.7). Therefore, under the factorization assumption, it must be that these two exponents are trivial:

$$e'z - ez' = et' - e't = 0$$

Note that these equations depend on values (e, e'), which are differences between challenges picked by the simulator. By the pigeonhole principle, and using the fact that an integer k has about $O(1/k)$ divisors, we can show that if $x = \gcd(e, \alpha z + t)$ is too large, then there is a non-negligible probability that the above equation will not hold (upon random choices of the challenge difference e'), which would contradict the factorization assumption. Therefore, it must be that x remains small, which concludes the proof. Below, we give a formal and detailed proof that follows this intuition.

5.3 Proof of Theorem 5.2.1

We start with some preliminaries, and then discuss various cases.

5.3.1 Preliminaries

The proof will make use of the splitting lemma [PS96; PS00], that we recall below:

Lemma 5.3.1. *Let $A \subset X \times Y$ such that $\Pr[(x, y) \in A] \geq \varepsilon$. For any $\varepsilon' < \varepsilon$, if one defines $B = \{(x, y) \in X \times Y \mid \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \varepsilon'\}$, then it holds that:*

$$(i)\ \Pr[B] \geq \varepsilon' \qquad (ii)\ \forall (x, y) \in B, \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \varepsilon' \qquad (iii)\ \Pr[B \mid A] \geq \varepsilon'/\varepsilon.$$

In the proof, we will consider an adversary with a random tape R who succeeds with some probability ε in any run of the full argument. Our proof will make use of rewinding: we will rewind the adversary several time to get several transcripts of the protocol for the *same* random tape R, and various challenges. The purpose of the splitting lemma is therefore to get a bound on the probability of getting valid transcripts when we fix R and run the adversary on various challenges.

5.3.2 Detailed Proof

Let us suppose the extractor $\mathcal{S}im$ (that is $\mathcal{S}im_{\mathsf{KE}}$ in this case) is given a $4/\varepsilon$-RSA challenge (n, e, u), which means that the exponents e is randomly chosen prime to $\varphi(n)$ but also in the set $[1, 4/\varepsilon]$. It sets $h \leftarrow u^2 \bmod n$ and $g \leftarrow h^\alpha \bmod n$ for a random exponent $\alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_{n^2}$. It sets $\mathsf{pp} = (n, g, h)$. Note that as u is random in $\mathbb{Z}_n^*$, (g, h) are indeed distributed as in the real protocol. We consider an adversary $\mathcal{A}$ that provides a convincing proof of knowledge of an opening of c with probability ε, with the parameters $(\mathsf{pp} = (n, g, h), \kappa, c, k_x)$.

Note that the probability distribution of a protocol execution is $D = (R, e_0)$, where R is the adversary's random tape that determines d and e_0 is the random challenge from the honest verifier. Since this is a "good" adversary, we assume that on a random pair (R, e_0), its probability to output a valid transcript (d, e_0, z_0, t_0) is greater than ε. We apply the splitting lemma with $\varepsilon' = \varepsilon/2$ for the distribution $D = \{R\} \times \{e\}$: after one execution, with probability greater than ε, we obtain a successful transcript (d, e_0, z_0, t_0). In such a case, with probability greater than $1/2$, R is a good random tape, which means that another execution with the same R but a random challenge e_i will lead to another successful transcript (d, e_i, z_i, t_i) with probability $\varepsilon' = \varepsilon/2$. Note that since R is kept unchanged, d is the same. Globally, with probability greater than $\varepsilon^2/4$, after 2 executions of the protocol, one gets two related successful transcripts: (d, e_0, z_0, t_0) and (d, e_1, z_1, t_1).

Without loss of generality, we may assume $e_0 \geq e_1$. Writing $e'_1 \leftarrow e_0 - e_1$, $z'_1 \leftarrow z_0 - z_1$, and $t'_1 \leftarrow t_0 - t_1$, the two valid tuples lead to the relation $c^{e'_1} = g^{z'_1} h^{t'_1} \bmod n$. We now consider three cases.

Case 1: e'_1 divides both z'_1 and t'_1 with probability greater than $\varepsilon^2/8$. $\mathcal{S}im$ simply outputs the pair of integers $(x_1, r_1) \leftarrow (z'_1/e'_1, t'_1/e'_1)$. If e'_1 is odd, and thus prime to $\varphi(n)$, we have $c = g^{x_1} h^{r_1} \bmod n$. However, if $e'_1 = 2^v \rho$ for an odd ρ and $v \geq 1$, $(c^{-1} g^{x_1} h^{r_1})^{2^v} = 1 \bmod n$: from the Fact 1 from Proposition 2.2.7, $(c^{-1} g^{x_1} h^{r_1})^2 = 1 \bmod n$:

- either $c^{-1} g^{x_1} h^{r_1} = \pm 1 \bmod n$, and so $c = \pm g^{x_1} h^{r_1} \bmod n$ (valid opening);
- or we have a non-trivial square root of 1, which leads to the factorization of n (see Proposition 2.2.7). As the RSA assumption is stronger than the factorization, when we solve the factorization, we can compute the solution to the RSA challenge.

Case 2: e'_1 divides $\alpha z'_1 + t'_1$ (but does not divide both z'_1 and t'_1). Let us argue e'_1 cannot divide $\alpha z'_1 + t'_1$ with probability greater than $1/2$ when e'_1 does not divide both z'_1 and t'_1 (independently of the actions of $\mathcal{A}$). Note that this is exactly the case 2 from [DF02]. The intuition behind the proof is that the only information that $\mathcal{A}$ can get about α is from $g = h^\alpha \bmod n$. However, this leaks only $\alpha \bmod p'q'$, while α was taken at random in $\mathbb{Z}_{n^2}$: all the information on its most significant bits is *statistically* hidden. We recall below the proof given by Damgård and Fujisaki, for completeness.

Let Q be a prime factor of e'_1 and j be the integer such that Q^j divides e'_1 but Q^{j+1} does not divide e'_1, and at least one of z'_1 or t'_1 is non-zero modulo Q^j. Since e'_1 does not divide both z'_1 and t'_1, so such a pair (Q, j) does necessarily exist. Actually, if Q^j divides z'_1, as it divides e'_1, it must also divide $\alpha z'_1 + t'_1$ and therefore t'_1, which was excluded (at least one of z'_1 or t'_1 is non-zero modulo Q^j). Therefore, $z'_1 \neq 0 \bmod Q^j$.

We can write $\alpha = [\alpha \bmod p'q'] + \lambda p'q'$ for some λ. Let us denote $\mu = [\alpha \bmod p'q']$. The tuple (n, g, h) uniquely determines μ, whereas λ is statistically unknown to the prover. As

Q^j divides e'_1, it also divides $\alpha z'_1 + t'_1$:

$$\alpha z'_1 + t'_1 = \lambda z'_1 p'q' + \mu z'_1 + t'_1 = 0 \bmod Q^j.$$

Note that $p'q' \neq 0 \bmod Q$, since p' and q' are κ-bit primes and the challenges are less than 2^κ. And from the view of the adversary, λ is uniformly distributed in $\mathbb{Z}_n$, while it must satisfy the above equation for Case 2 to occur. But since this equation has at most $\gcd(z'_1 p'q', Q^j)$ solutions, which is a power of Q (and at most Q^{j-1}), and since n is larger than Q^j by a factor (far) bigger than 2^κ, the distribution of $\lambda \bmod Q^j$ is statistically close to uniform in $\mathbb{Z}_{Q^j}$, and the probability that λ satisfies the above equation is bounded by $1/Q - 2^{-\kappa} \leq 1/2$, independently of the actions of $\mathcal{A}$. Hence,when Case 1 does not happen, Case 2 cannot occur either with probability greater than $1/2$ when one gets two related successful transcripts. Overall, when Case 1 does not happen, we are necessarily in the following situation:

Case 3: e'_1 does not divide $\alpha z'_1 + t'_1$ with probability greater than $\varepsilon^2/16$. We will now prove that when Case 3 occurs, $\mathcal{S}im$ can solve an RSA instance, which is the difference with the original proof. Let $\beta_1 = \gcd(e'_1, \alpha z'_1 + t'_1)$. Since e'_1 does not divide $\alpha z'_1 + t'_1$, we necessarily have $1 \leq \beta_1 < e'_1$. Let $\Gamma_1 \leftarrow e'_1/\beta_1$ and $F_1 \leftarrow (\alpha z'_1 + t'_1)/\beta_1$: F_1/Γ_1 is the irreducible fraction form of $(\alpha z'_1 + t'_1)/e'_1$ and $e'_1 \geq \Gamma_1 > 1$.

We now consider the following complementary situations:

- **Subcase 3.a.** $\Gamma_1 \leq 4/\epsilon$ with probability at least $\varepsilon^2/32$
- **Subcase 3.b.** $\Gamma_1 > 4/\epsilon$ with probability at least $\varepsilon^2/32$

Subcase 3.a. If $\Gamma_1 \leq 4/\varepsilon$, since $\beta_1 < e'_1$, we must have $\Gamma_1 \in [\![2\,;4/\varepsilon]\!]$. In order to simplify the notations, after one rewind, we get (e', z', t') so that $c^{e'} = g^{z'} h^{t'} \bmod n$ and $\beta = \gcd(e', \alpha z' + t')$ with $1 < \Gamma_1 = e'/\beta \leq 4/\varepsilon$.

We note $e' = \beta\Gamma_1$ and $\alpha z' + t' = \beta k$ for relatively prime integers Γ_1 and k. Since $h = u^2 \bmod n$ and $c^{e'} = h^{\alpha z' + t'} \bmod n$, we have $c^{e'} = u^{2(\alpha z' + t')} \bmod n$, which reduces to $c^{\Gamma_1} = c^{e'/\beta} = \pm u^{2(\alpha z' + t')/\beta} = \pm u^{2k} \bmod n$, where Γ_1 and k are relatively prime, and $\Gamma_1 > 1$. We now consider two additional complementary situations:

- if $\Gamma_1 = 2^a$ with $a \geq 1$ with probability at least $\varepsilon^2/64$, we thus have with probability $\varepsilon^2/64$ an odd k such that $c^{2^a} = u^{2k} \bmod n$: $c^{2^{a-1}}$ and u^k are two square roots of the same value. Since no information leaks about the actual square roots $\{u, -u\}$ known for h, nor for $h^k \bmod n$, so $c^{2^{a-1}} \neq \pm u^k \bmod n$ with probability $1/2$, which leads to the factorization of n with probability $1/2$ (see Proposition 2.2.7). Hence, when Case 3 happens and in the case of this situation, we solve the RSA challenge with probability at least $\varepsilon^2/128$.
- If $\Gamma_1 = 2^a v$ with an odd $v > 1$ with probability at least $\varepsilon^2/64$, it thus holds with probability $\varepsilon^2/64$ that $C^v = u^{2k} \bmod n$, for $C = \pm c^{2^a}$ and $\gcd(v, 2k) = 1$, since $v|\Gamma_1$ and v is odd. Using Proposition 2.2.9, one gets the v-th root of u modulo n, for $v \in [\![3\,;4/\varepsilon]\!] \cap \mathsf{P}_n$. Since our simulation that uses the RSA challenge (n, u, e) does not leak *any* information about e, $v = e$ with probability greater than $\varepsilon/2$, if the exponent e is randomly chosen in $[\![2\,;4/\varepsilon]\!] \cap \mathsf{P}_n$ (this set being exactly the set of odd integers smaller than $4/\varepsilon$, it contains approximately $2/\varepsilon$ elements). Hence, when Case 3 happens and $\Gamma_1 \leq 4/\varepsilon$, if we are in this situation, we solve an RSA challenge with probability at least $\varepsilon^2/64 \times \varepsilon/2 = \varepsilon^3/128$.

Overall, each time Case 3 happens and we are in the situation 3.*a.*, we get a solution to the RSA challenge with probability at least $\varepsilon^3/128$.

Subcase 3.b. We now assume that when Case 3 occurs, $\Gamma_1 > \varepsilon/4$ with probability at least $\varepsilon^2/32$. When $\Gamma_1 > \varepsilon/4$, the simulator rewinds the protocol once more. Now, consider all the possible challenges e_2 for this rewinding. For random challenges e_2, the differences $|e_0 - e_2|$ are uniformly distributed over $[\![0\,;2^\kappa]\!]$, and the number of challenge-differences that Γ_1 divides is at most $(2^\kappa+1)/\Gamma_1 < 4(2^\kappa+1)/\varepsilon$. Therefore, the probability that Γ_1 divides $|e_0 - e_2|$ for a random e_2 is at most $\varepsilon/4$. Recall that if the first rewinding succeeds, by the splitting lemma, rewinding the protocol once more produces a successful transcript with probability at least $\varepsilon/2$; therefore, when one rewinds the protocol with challenge e_2 and gets a successful transcript, Γ_1 does *not* divide $|e_0 - e_2|$ with probability at least $1/2$.

Therefore, when rewinding the protocol with challenge e_2, one gets a successful transcript such that Γ_1 does not divide $|e_0 - e_2|$ with probability at least $\varepsilon/4$. When this happens, the value Γ_2 (defined in the same way than Γ_1), which does divide $|e_0 - e_2|$ by definition, cannot be equal to Γ_1. Keeping this in mind, suppose we got the following two relations from two successful rewindings: $c^{e'_1} = g^{z'_1}h^{t'_1} \bmod n$ and $c^{e'_2} = g^{z'_2}h^{t'_2} \bmod n$, and so $c^{e'_1e'_2} = g^{e'_2z'_1}h^{e'_2t'_1} = g^{e'_1z'_2}h^{e'_1t'_2} \bmod n$. This leads, for $\Delta_z = e'_2z'_1 - e'_1z'_2$ and $\Delta_t = e'_2t'_1 - e'_1t'_2$, to

$$g^{\Delta_z} = g^{e'_2z'_1 - e'_1z'_2} = h^{e'_1t'_2 - e'_2t'_1} = h^{-\Delta_t} \bmod n.$$

If $\Delta_z = \Delta_t = 0$, then it holds that $z'_2/e'_2 = z'_1/e'_1$ and $t'_2/e'_2 = t'_1/e'_1$:

$$\frac{F_2}{\Gamma_2} = \frac{\alpha z'_2 + t'_2}{e'_2} = \alpha \cdot \frac{z'_2}{e'_2} + \frac{t'_2}{e'_2} = \alpha \cdot \frac{z'_1}{e'_1} + \frac{t'_1}{e'_1} = \frac{\alpha z'_1 + t'_1}{e'_1} = \frac{F_1}{\Gamma_1}.$$

Since they are both the irreducible notations of the same fraction, we necessarily have $\Gamma_1 = \Gamma_2$ and $F_1 = F_2$. But recall that we have shown that with probability ay least $1/2$, this does not happen. Hence, when subcase 3.b happens and $\Gamma_1 > \varepsilon/4$, the next rewinding produces a successful transcript such that the pair (Δ_z, Δ_t) is non-trivial with probability at least $\varepsilon/4$, which leads to the factorization of n with probability $1/2$, from the Fact 2 from Proposition 2.2.7. Overall, when Case 3 occurs and we are in the situation 3.*b.*, we get a solution to the RSA challenge with probability at least $\varepsilon^2/32 \times \varepsilon/4 \times 1/2 = \varepsilon^3/256$.

Overall Success Probability. All in all, each time Case 3 occurs, in any of the two complementary situations 3.*a.* and 3.*b.*, we get a solution to the RSA challenge with probability at least $\varepsilon^3/256$. But we have already seen that when Case 1 does not occur, Case 3 occurs necessarily; hence, each time Case 1 does not occur, we get a solution to the RSA challenge with probability at least $\varepsilon^3/256$. Suppose now that Case 1 occurs. There are two complementary situations: either we get a valid opening with probability at least $\varepsilon^2/16$, or we get a non-trivial square root of 1 with probability at least $\varepsilon^2/16$. Overall, whichever case occurs, we either get a valid opening with probability at least $\varepsilon^2/16$, or we solve an RSA challenge with probability at least $\varepsilon^3/256$. □

5.4 Classical Extensions and Applications

We revisit the natural implications of the commitment scheme of Section 5.2 and its argument of knowledge. More precisely, we generalize the results of previous sections while we commit to vectors of integers. Then, we also show the security of Lipmaa's range proofs [Lip03]

under the RSA assumption to illustrate how the result of Section 5.3 extends to more general arguments over the integers.

5.4.1 Generalized Commitment of Integers

The following commitment scheme allows committing to a vector of integers $(m_1, \ldots, m_\ell)$ with a single element of the form $c = g_1^{m_1} \cdots g_\ell^{m_\ell} h^r \bmod n$:

- $\mathsf{Setup}(1^\kappa, \ell)$ runs $(n, (p, q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$, and picks $\ell+1$ random generators $(g_1, \ldots, g_\ell, h)$ of QR_n. It returns $\mathsf{pp} = (n, g_1, \ldots, g_\ell, h)$;
- $\mathsf{Commit}(\mathsf{pp}, \vec{m}; r)$, for $\mathsf{pp} = (n, g_1, \ldots, g_\ell, h)$, a vector $\vec{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}^\ell$, and some random coins $r \xleftarrow{\$} [\![0\,; n/4]\!]$, computes $c = g_1^{m_1} \cdots g_\ell^{m_\ell} h^r \bmod n$, and returns (c, d) with $d = r$;
- $\mathsf{Verify}(\mathsf{pp}, c, d, \vec{m})$ parses pp as $\mathsf{pp} = (n, g_1, \ldots, g_\ell, h)$ and outputs 1 if $c = g_1^{m_1} \cdots g_\ell^{m_\ell} h^d \bmod n$ and 0 otherwise.

Again, the above commitment scheme is obviously *correct*. The *hiding* property relies on the existence of α_i such that $g_i = h^{\alpha_i} \bmod n$ for $i = 1, \ldots, \ell$, and so

$$\begin{aligned} c = \mathsf{Commit}(\mathsf{pp}, \vec{m}; r) &= g_1^{m_1} \cdots g_\ell^{m_\ell} h^r = h^{r + \sum \alpha_i m_i} \\ &= h^{(r + \sum \alpha_i (m_i - m_i')) + \sum \alpha_i m_i'} = g_1^{m_1'} \cdots g_\ell^{m_\ell'} h^{r + \sum \alpha_i (m_i - m_i')} \\ &= \mathsf{Commit}(\mathsf{pp}, \vec{m}'; r'), \end{aligned}$$

for any $\vec{m}' = (m_1', \ldots, m_\ell') \in \mathbb{Z}$, with $r' \leftarrow [r + \sum \alpha_i (m_i - m_i') \bmod p'q']$, that is smaller than n.

The binding property relies on the Integer Factorization assumption: indeed, from two different openings $(\vec{m}, d)$ and $(\vec{m}', d')$ for a commitment c, with $d' > d$, the validity checks show that $g_1^{m_1} \cdots g_\ell^{m_\ell} h^d = g_1^{m_1'} \cdots g_\ell^{m_\ell'} h^{d'} \bmod n$, and so, if one has chosen β_i such that $g_i = g^{\beta_i} \bmod n$, for a random square g, then one knows $g^{\sum \beta_i (m_i - m_i')} = h^{d' - d} \bmod n$. The Fact 2 from Proposition 2.2.7 leads to the conclusion.

To avoid a trusted setup, one can note that the guarantees for the prover (the hiding property) just rely on the existence of α_i such that $g_i = h^{\alpha_i} \bmod n$ for $i = 1, \ldots, \ell$. The well-formedness of the RSA modulus is for the security guarantees against the verifier. It is important for him that the prover cannot break the RSA assumption. So the setup can be run by the verifier, with an additional proof of existence of α_i such that $g_i = h^{\alpha_i} \bmod n$ for $i = 1, \ldots, \ell$ to the prover.

5.4.2 Zero-Knowledge Argument of Opening

An argument of knowledge of an opening of a commitment $c = g_1^{x_1} \cdots g_\ell^{x_\ell} h^r \bmod n$ in the general case can be easily adapted from the normal case leading to a transcript of the form $(d, e, (z_1, \ldots, z_\ell, t))$ with $d = g_1^{y_1} \cdots g_\ell^{y_\ell} h^s$, and $c^e d = g_1^{z_1} \cdots g_\ell^{z_\ell} h^t \bmod n$.

As above, the knowledge-extractor rewinds the execution for the same d, but two different challenges $e_0 \neq e_1$. Doing the quotient of the two relations, d cancels out: $c^{e'} = g_1^{z_1'} \cdots g_\ell^{z_\ell'} h^{t'} \bmod n$.

Let us assume that one would have set $g_i = g^{a_i} h^{b_i} \bmod n$, we would have

$$c^{e'} = g^{\sum a_i z'_i} h^{\sum b_i z'_i + t'} \bmod n.$$

Under the RSA assumption, we are in the above case 1: e' divides both $\sum a_i z'_1$ and $\sum b_i z'_i + t'$ with non-negligible probability. Since the coefficients a_i's and b_i's are random, this means that e' divides all the z'_i's and t'. Hence, one can set $\mu_i = z'_i/e'$, for $i = 1, \ldots, \ell$ and $\tau = t'/e'$, and $c = g_1^{\mu_1} \cdots g_\ell^{\mu_\ell} h^\tau \bmod n$ is a valid opening of c.

5.4.3 Equally Efficient Range Proofs from RSA

We show that Lipmaa's range proof [Lip03] also benefits from our technique as the Strong-RSA assumption can also be avoided in the security analysis.

Range Proof from Integer Commitment Scheme. Let $c = g^x h^r \bmod n$ be a commitment of a value x and $[\![a\,;b]\!]$ be a public interval. As the commitment is homomorphic, one can efficiently compute a commitment c_a of $x - a$ and a commitment c_b of $b - x$ from c. To prove that $x \in [\![a\,;b]\!]$, this is enough to show that c_a and c_b commit to positive values. Let us focus on the proof that $c_a = g^{x-a} h^r \bmod n$ commits to a positive value, since the same method applies for c_b. To do so, the prover computes (x_1, x_2, x_3, x_4) such that $x - a = \sum_{i=1}^4 x_i^2$. By a famous result from Lagrange, such a decomposition exists if and only if $x - a \geq 0$. Moreover, this decomposition can be efficiently computed by the Rabin-Shallit algorithm [RS86], for which Lipmaa [Lip03] also suggested some optimizations. The prover commits to (x_1, x_2, x_3, x_4) in (c_1, c_2, c_3, c_4), where $c_i = g^{x_i} h^{r_i} \bmod n$ for each $i = 1$ to 4. Now, the prover proves his knowledge of openings $x - a$, x_1, x_2, x_3, x_4 (along with random coins r, r_1, r_2, r_3, r_4) of c_a, c_1, c_2, c_3, c_4 satisfying $\sum_{i=1}^4 x_i^2 = x - a$ over the integers.

The reason allowing to solely rely on the RSA assumption in the range proof comes from the fact that the first part of the argument reduces to an argument of knowledge of openings x_1, x_2, x_3, x_4 of c_1, c_2, c_3, c_4 while the remaining part simply ensures the relation $\sum_{i=1}^4 x_i^2 = x - a$ to hold. Indeed, once the witnesses are extracted, this is implied by the representation $c_a = \prod_{i=1}^4 c_i^{x_i} h^{r - \sum x_i r_i} \bmod n$ which can be seen as generalized commitment scheme with basis (c_1, c_2, c_3, c_4, h) from which the opening cannot change. Therefore, the argument can be seen as five parallel arguments of knowledge, the fifth one being an argument of knowledge for a generalized commitment, where the opening for the last argument is the vector of the openings for the other arguments. A formal proof of an optimized version of this protocol under the intractability of the RSA assumption is presented in the next chapter, in Section 5.4.4.

Extension. Since most of the arguments of knowledge of a solution to a system of equations over the integers [CCT07] can be split into parallel arguments of knowledge of values assigned to the variables and a proof of membership (in the language composed of all the solutions of the system), which is expressed as representations corresponding to generalized commitments, our analysis extends to all "discrete-logarithm relation set" (see [KTY04]): the description of the protocol is unchanged but the security only relies on the standard RSA assumption.

5.4.4 A Correction on Lipmaa's Argument for Positivity

For the sake of completeness, in this section, we outline a flaw in the protocol of [Lip03, Annex B]. We then construct a corrected protocol and prove its security. We notified the author of our finding, who acknowledged the issue.

5.4.4.1 Initial Protocol

Lipmaa [Lip03, Annex B] proposed an efficient zero-knowledge argument of positivity. Since the exact protocol was not fully detailled, we found that the protocol could be understood in two different way. If one closely follow the description of the protocol, then the resulting scheme does not satisfy correctness; the proof of correctness seems to assume a different scheme.

We describe on Figure 5.2 our understanding from reading its proof of correctness. Unfortunately, it is not sound, and the flaw comes from the fact that the original protocol is described as using a generalized Damgård-Fujisaki commitment scheme. However, the same basis is used to commit to masks m_1, m_2, m_3, m_4, which implies that the prover will only be (computationally) bound to $\sum_i x_i$ in the argument.

$\mathcal{P}$ knows (x, r) such that $c = g^x h^r \bmod n$ and $x \geq 0$. $\mathcal{V}$ knows c.

1. $\mathcal{P}$ computes $(x_i)_{i\leq 4}$ such that $x = \sum_{i=1}^{4} x_i^2$. $\mathcal{P}$ commits the x_i's with fresh random coins $r_i \xleftarrow{\$} [\![0\,;n]\!]$ as $c_i = g^{x_i} h^{r_i} \bmod n$. $\mathcal{P}$ sends c_1, c_2, c_3, c_4 to $\mathcal{V}$.
2. $\mathcal{P}$ picks $(m_i)_{i=1}^4 \xleftarrow{\$} [\![0\,;2^{B/2+2\kappa}]\!]^4$, $s_1 \xleftarrow{\$} [\![0\,;2^{2\kappa+|n|}]\!]$, and $s_2 \xleftarrow{\$} [\![0\,;2^{B/2+|n|+2\kappa}]\!]$. Then, $\mathcal{P}$ sends $d_1 = g^{m_1+m_2+m_3+m_4} \cdot h^{s_1} \bmod n$ and $d_2 = c_1^{m_1} c_2^{m_2} c_3^{m_3} c_4^{m_4} \cdot h^{s_2} \bmod n$.
3. $\mathcal{V}$ picks a challenge $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!]$ and sends it to $\mathcal{P}$.
4. $\mathcal{P}$ computes and sends $z_i = ex_i + y_i$, for $i = 1$ to 4, $t_1 = e\sum_i r_i + s_1$ and $t_2 = e(r - \sum_i x_i r_i) + s_2$.
5. $\mathcal{V}$ accepts the argument if both

$$(c_1c_2c_3c_4)^e \cdot d_1 = g^{z_1+z_2+z_3+z_4} h^{t_1} \quad \text{and} \quad c^e \cdot d_2 = c_1^{z_1} c_2^{z_2} c_3^{z_3} c_4^{z_4} \cdot h^{t_2}.$$

Figure 5.2: Lipmaa's Compact Argument for Positivity

Actually, it does not seem possible to rely on generalized commitments to get a more efficient protocol. Concretely, let us consider a prover $\mathcal{P}^*$ holding (x, r) such that $c = g^x h^r$ and $x = -1$. $\mathcal{P}^*$ commits $x_1 = 0, x_2 = 1, x_3 = 0$ and $x_4 = 0$, and computes d_1, d_2 honestly. After receiving a challenge, however, $\mathcal{P}^*$ sets $\bar{x}_1 = 2, \bar{x}_2 = -1, \bar{x}_1 = 0, \bar{x}_1 = 0$, and sends $\bar{z}_i = e\bar{x}_i + m_i$ for $i = 1$ to 4 instead of the correct z_i, and $\bar{t}_2 = e(r - \sum_i \bar{x}_i r_i) + s_2$ instead of the correct t_2. The values $\bar{x}_i$ were chosen so that $\sum_i \bar{x}_i = \sum_i x_i$, hence $\sum_i \bar{z}_i = e(\sum_i \bar{x}_i) + \sum_i m_i = e(\sum_i x_i) + \sum_i m_i = \sum_i z_i$, and so the check that $(c_1c_2c_3c_4)^e \cdot d_1 = g^{\bar{z}_1+\bar{z}_2+\bar{z}_3+\bar{z}_4} h^{t_1}$ succeeds. The second verification is equivalent to checking that $\sum_i x_i \cdot \bar{x}_i = x$, which is the case here $(-1 = 0 \times 2 + 1 \times (-1) + 0 \times 0 + 0 \times 0)$: $\mathcal{V}$ accepts the argument even though the value x known by $\mathcal{P}^*$ is strictly negative.

A natural way to fix this flaw without increasing the communication would be to require the verifier to send a seed λ between step 1 and step 2, from which pseudo-random values $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are stretched, to send $d_1 = \sum_i \lambda_i m_i$, $t_1 = e\sum_i \lambda_i r_i + s_1$ and to adapt the verification equation accordingly. However, an attack quite similar to the one we've just described succeeds with good probability in this case (it is sufficient that the gcd of λ_i and λ_j is small, for some $i \neq j$, for the attack to succeed). The interesting point is that we cannot

batch the arguments of knowledge and the proof of membership at the same time.

5.4.4.2 Corrected Protocol

In this section, we propose a variant of Lipmaa's protocol [Lip03] proving that a committed x is a sum of four squares. There are two correct ways to construct an optimized argument of positivity. A first possibility is to rely on a collision-resistant hash function to strongly reduce the length of the flow sent by $\mathcal{P}$ in step 2 (note that we only require the hash function to be collision-resistant, hence the protocol is in the standard model). An alternative would be to let $\mathcal{P}$ send all individual values $(d_i)_i$ and d in step 2 instead of a single hash, and to stretch pseudo-random values from e in step 4 to batch all the t_i into a single value. We describe the former solution, on Figure 5.3, as it is slightly more efficient than the latter in terms of communication and enjoys a better security reduction.

$\mathcal{P}$ knows (x, r) such that $c = g^x h^r$ and $x \geq 0$. $\mathcal{V}$ knows c. Let $H : \mathbb{Z}_n^5 \mapsto \{0,1\}^{2\kappa}$ be a collision-resistant hash function.

1. $\mathcal{P}$ computes $(x_i)_{i\leq 4}$ such that $x = \sum_{i=1}^4 x_i^2$. $\mathcal{P}$ commits the x_i's with fresh random coins $r_i \xleftarrow{\$} [\![0\,;n]\!]$ as $c_i = g^{x_i} h^{r_i} \bmod n$. $\mathcal{P}$ sends c_1, c_2, c_3, c_4 to $\mathcal{V}$.
2. $\mathcal{P}$ picks $(m_i)_{i=1}^4 \xleftarrow{\$} [\![0\,;2^{B/2+2\kappa}]\!]^4$, $(s_i)_{i=1}^4 \xleftarrow{\$} [\![0\,;2^{2\kappa}n]\!]^4$, $s \xleftarrow{\$} [\![0\,;2^{B/2+2\kappa}n]\!]$, computes $(d_i = g^{m_i} h^{s_i} \bmod n)_{i=1}^4$, $d = \prod_{i=1}^4 c_i^{m_i} h^s$, and sends the commitment $\Delta = H(d_1, d_2, d_3, d_4, d)$ to $\mathcal{V}$.
3. $\mathcal{V}$ picks a challenge $e \xleftarrow{\$} [\![0\,;2^{\kappa}]\!]$ and sends it to $\mathcal{P}$.
4. $\mathcal{P}$ computes and sends $z_i = ex_i + m_i$ and $t_i = er_i + s_i$ for $i = 1$ to 4, and $t = e(r - \sum x_i r_i) + s$.
5. $\mathcal{V}$ accepts the argument if $\Delta = H\left((g^{z_i} h^{t_i} c_i^{-e} \bmod n)_{i=1}^4, \prod_{i=1}^4 c_i^{z_i} h^t c^{-e} \bmod n\right)$.

Figure 5.3: Variant of Lipmaa's Compact Argument for Positivity

5.4.4.3 Proof of Security

Correctness immediately follows from a careful inspection of the protocol.

Zero-Knowledge Property. We now argue that the protocol is honest-verifier zero-knowledge: given c and a challenge e, the simulator $\mathcal{S}im_{\mathsf{ZK}}$ sends random group elements c_1, c_2, c_3, c_4, and picks random $(z_i, t_i) \xleftarrow{\$} [\![0\,;2^{B/2+2\kappa}]\!] \times [\![0\,;2^{2\kappa}n]\!]$ for $i = 1$ to 4, and a random $t \xleftarrow{\$} [\![0\,;2^{B/2+2\kappa}n]\!]$. In step 2, $\mathcal{S}im_{\mathsf{ZK}}$ sends $\Delta = H\left((g^{z_i} h^{t_i} c_i^{-e} \bmod n)_{i=1}^4, \prod_{i=1}^4 c_i^{z_i} h^t c^{-e} \bmod n\right)$. The commitments $(c_i)_i$ are perfectly indistinguishable from valid commitments, and $((z_i)_i, (t_i)_i, t)$ are statistically indistinguishable from honestly computed integers, with a similar analysis as in Section 5.2.

Knowledge Extractability. Let us now prove the knowledge extractability of the protocol under the RSA assumption. A prover $\mathcal{P}'$ which succeeds in providing a convincing proof with probability ε is rewinded, to provide two valid proofs for the same initial commitments

$c_1, c_2, c_3, c_4, \Delta$. Under the collision-resistance of the hash function: $g^{z_i} h^{t_i} c_i^{-e} = g^{z'_i} h^{t'_i} c_i^{-e'} \bmod n$, for $i = 1$ to 4, and $\prod c_i^{z_i} h^t c^{-e} = \prod c_i^{z'_i} h^{t'} c^{-e'} \bmod n$.

Hence, we have, for $i = 1$ to 4, $c_i^{e'-e} = g^{z_i - z'_i} h^{t_i - t'_i} \bmod n$, and $c^{e'-e} = \prod c_i^{z_i - z'_i} h^{t-t'} \bmod n$. Using a similar argument as in the proof of Theorem 5.2.1, unless one can break the RSA assumption, $e' - e$ likely divides all the other differences and so, with $\rho_i = (z_i - z'_i)/(e' - e)$ and $w_i = (t_i - t'_i)/(e' - e)$ for $i = 1$ to 4, and $w = (t - t')/(e' - e)$, we have $c_i = g^{\rho_i} h^{w_i}$, and $c = \prod_{i=1}^{4} c_i^{\rho_i} h^w$.

Altogether, this implies that $c = \prod_{i=1}^{4} g^{\rho_i^2} h^{w_i \rho_i} h^w = g^{\sum \rho_i^2} h^{w + \sum w_i \rho_i} \bmod n$. The commitment c thus contains $x = \sum \rho_i^2$, that is necessarily positive.

6

More Efficient Zero-Knowledge Arguments over the Integers

6.1 Introduction

In this chapter, we continue the study of zero-knowledge arguments over the integers, by providing new constructions of such arguments. Compared with existing techniques, our new constructions save communication and greatly reduce the work of the verifier, in exchange for an increased work for the prover. Therefore, they are well-suited for use in secure computation protocols based on a client-server model, where a computationally weak client asks a powerful server to perform some computation, and must verify efficiently that the server behaved honestly. We note that the constructions presented in this chapter do also benefit from the new analysis developped in Chapter 5. Therefore, their security is based on the RSA assumption.

6.1.1 Our Method in a Nutshell

As a starting point, we revisit a commitment scheme which was formally introduced in [Gen04], where a commitment for a message $m \in \mathbb{Z}_\pi$ with randomness $R \in \mathbb{Z}_n^*$ is computed as $c \leftarrow g^m R^\pi \bmod n$. This commitment scheme is perfectly hiding, and its binding property relies on the RSA assumption with prime exponent π in $\mathbb{Z}_n^*$. As for the Damgård-Fujisaki commitment scheme, the security of an argument of knowledge of an opening can be based on the classical RSA assumption. In addition, we identify an interesting property that is satisfied by this commitment, which corresponds informally to the possibility to see this commitment scheme either as an integer commitment scheme (i.e., $c = g^m h^r \bmod n$), or, after some secret exponent has been revealed, as a commitment scheme over $\mathbb{Z}_\pi$ for some prime π (i.e., $c = g^m R^\pi \bmod n$). Note that in both situations, the security of the commitment scheme and the argument of knowledge relies on the RSA assumption only. More specifically, it relies on two different variants of the RSA assumption, with respect to the distribution of random small exponents, or with respect to the distribution of random prime exponents.

We show how one can take advantage of this feature to improve the efficiency of zero-knowledge arguments over the integers as the knowledge of the order π is *delayed* in the protocol. Note, however, that this comes at the cost of making the argument private coin (hence, unlike classical Σ-protocols, it cannot be made non-interactive in the random oracle model anymore). Our method allows to save communication and greatly reduces the work of

the verifier, compared with a classical zero-knowledge argument for the same statement. We illustrate our method on range proofs [Lip03], a zero-knowledge argument of knowledge of an input to a commitment such that the input belongs to some public interval.

6.1.2 Organization

Section 6.2 revisits the commitment scheme of [Gen04] and shows how, by switching from the previous commitment to this one, we can get a new method for performing zero-knowledge arguments over the integers, that is more efficient. Eventually, Section 6.3 illustrates our method on range proofs, with concrete efficiency comparisons.

6.2 Commitment with Knowledge-Delayed Order

In this section, we show that the Damgård-Fujisaki commitment scheme can be efficiently combined with another RSA-based commitment scheme which, as far as we know, was proposed by Gennaro [Gen04]: we show how Damgård-Fujisaki commitments (which are homomorphic over the integers) can be converted into Gennaro commitments (which are homomorphic over $\mathbb{Z}_\pi$ for some prime π). We rely on this feature to design a method to improve the efficiency of zero-knowledge arguments over the integers on several aspects, by allowing the players to perform some of the computations over $\mathbb{Z}_\pi$ rather than over the integers. We then illustrate our technique on the famous example of range proofs.

6.2.1 RSA-based Commitments with Known Order

We recall the homomorphic commitment scheme over $\mathbb{Z}_\pi$ of [Gen04]. The order of the commitment is a known prime $\pi > 2^\kappa$.

Description of the Generalized Commitment Scheme. Let us describe the commitment of vectors of integers $(m_1, \ldots, m_\ell)$:

- $\mathsf{Setup}(1^\kappa)$: runs $(n, (p, q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$, and picks ℓ random generators $g_1, \ldots, g_\ell$ of QR_n. Then, it picks a random prime $\pi \in [\![2^{\kappa+1}; 2^{\kappa+2}]\!]$, and returns $\mathsf{pp} = (n, g_1, \ldots, g_\ell, \pi)$;
- $\mathsf{Commit}(\mathsf{pp}, \vec{m}; r)$: for $\mathsf{pp} = (n, g_1, \ldots, g_\ell, \pi)$, a vector $\vec{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}_\pi^\ell$, and some random coins $r \xleftarrow{\$} \mathbb{Z}_n$, computes $c = g_1^{m_1} \cdots g_\ell^{m_\ell} r^\pi \bmod n$, and returns (c, d) with $d = r$;
- $\mathsf{Verify}(\mathsf{pp}, c, d, \vec{m})$: parses pp as $\mathsf{pp} = (n, g_1, \ldots, g_\ell, \pi)$ and outputs 1 if $c = g_1^{m_1} \cdots g_\ell^{m_\ell} r^\pi \bmod n$, and 0 otherwise.

The above commitment scheme is obviously *correct*. The *hiding* property relies on the bijectivity of the π-th power modulo n (as π is prime): for any message $\vec{m}' = (m'_1, \ldots, m'_\ell) \in \mathbb{Z}_\pi^\ell$, we have $c = g_1^{m'_1} \cdots g_\ell^{m'_\ell} \times g_1^{m_1 - m'_i} \cdots g_\ell^{m_\ell - m'_i} \times r^\pi \bmod n$. By noting s the π-th root of $g_1^{m_1 - m'_i} \cdots g_\ell^{m_\ell - m'_i}$, $c = \mathsf{Commit}(\mathsf{pp}, \vec{m}'; rs)$. The binding property uses an extension of Proposition 2.2.9 from Section 2.2.2.2: if one has chosen β_i such that $g_i = u^{2\beta_i}$, for a challenge RSA $u \in \mathbb{Z}_n^*$, two distinct openings $(\vec{m}, r) \neq (\vec{m}', s)$ satisfy $g_1^{m_1} \cdots g_\ell^{m_\ell} r^\pi = g_1^{m'_1} \cdots g_\ell^{m'_\ell} s^\pi \bmod n$, and so $(s/r)^\pi = u^{2a} \bmod n$, where $a = \sum \beta_i (m_i - m'_i) = a_1 \pi + a_0$, with

$0 \leq a_0 < \pi$. Let us note α and β the integers such that $\alpha\pi + \beta 2a_0 = \gcd(\pi, 2a_0) = 1$, and output $u_0 := u^{\alpha - 2a_1\beta} \cdot (s/r)^{\beta} \bmod n$, then

$$u_0^{\pi} = u^{\alpha\pi - 2a_1\beta\pi} \cdot (s/r)^{\beta\pi} = u^{1-2(a_0+a_1\pi)\beta} \cdot u^{2a\beta} = u \bmod n.$$

This breaks the RSA assumption with exponent π.

Homomorphic-Opening. In addition, this commitment scheme is homomorphic in $\mathbb{Z}_\pi$: given $c = g_1^{m_1} \cdots g_\ell^{m_\ell} r^\pi \bmod n$ and $d = g_1^{m'_1} \cdots g_\ell^{m'_\ell} s^\pi \bmod n$ with known openings, we can efficiently open the commitment $c \cdot d \bmod n$ to $\vec{\bar{m}} = (\bar{m}_1, \ldots, \bar{m}_\ell)$, with $\bar{m}_i = m_i + m'_i \bmod \pi$ for $1 \leq i \leq \ell$, and a random coin $rs \prod g_i^{(m_i + m'_i) \div \pi} \bmod n$, where $a \div b$ is the quotient of the Euclidean division. We emphasize this property to be essential to avoid working with integers in the arguments of knowledge of an opening: the prover can "reduce" its openings since π is known.

Argument of Opening. Given $\mathsf{pp} = (n, g_1, \ldots, g_\ell, \pi)$ and $c = g_1^{x_1} \cdots g_\ell^{x_\ell} r^\pi \bmod n$, with witness $(x_1, \ldots, x_\ell, r)$, we can describe a standard argument of knowledge of an opening:

Initialize: $\mathcal{P}$ and $\mathcal{V}$ decide to run the protocol on input (pp, κ, c);

Commit: $\mathcal{P}$ computes $d = g_1^{y_1} \cdots g_\ell^{y'_\ell} s^\pi$, for $y_i \xleftarrow{\$} \mathbb{Z}_\pi$, and $s \xleftarrow{\$} \mathbb{Z}_n^*$, and sends d to $\mathcal{V}$;

Challenge: $\mathcal{V}$ outputs $e \xleftarrow{\$} [\![0\,;2^\kappa]\!]$;

Response: $\mathcal{P}$ computes k_i, z_i, t such that $ex_i + y_i = k_i\pi + z_i$, with $0 \leq z_i < \pi$, and $t = g_1^{k_1} \cdots g_\ell^{k_\ell} \cdot r^e s \bmod n$. $\mathcal{P}$ outputs $(z = (z_i)_i, t)$;

Verify: $\mathcal{V}$ accepts the proof and outputs 1 if, for each i, $0 \leq z_i < \pi$, and $c^e d = g_1^{z_1} \cdots g_\ell^{z_\ell} t^\pi \bmod n$. Otherwise, $\mathcal{V}$ rejects the proof and outputs 0.

Completeness and *zero-knowledge* are straightforward. Then, let us focus on the *knowledge-extractability*: From two related valid transcripts, for the same d, we get as usual $c^{e-e'} = g_1^{z_1 - z'_1} \cdots g_\ell^{z_\ell - z'_\ell} \cdot (t/t')^\pi \bmod n$. Since the prime $\pi > 2^\kappa \geq ||e - e'||$, the simulator can compute $\alpha(e - e') + \beta\pi = 1$ and we have

$$c^{1-\beta\pi} = c^{\alpha(e-e')} = g_1^{\alpha(z_1 - z'_1)} \cdots g_\ell^{\alpha(z_\ell - z'_\ell)} \cdot (t/t')^{\alpha\pi} \bmod n.$$

Then, for $\alpha(z_i - z'_i) = l_i\pi + x'_i$ with $0 \leq x'_i < \pi$, and $T = c^\beta \cdot g_1^{l_1} \cdots g_\ell^{l_\ell} \cdot (t/t')^\alpha \bmod n$, we have a valid opening $(x'_1, \ldots, x'_\ell, T)$ of c.

6.2.2 Commitment with Knowledge-Delayed Order

Now, we show how we can hide the above commitment scheme with known prime order π into a commitment scheme of Section 5.2 with hidden order.

Description of the Commitment Scheme. As explained earlier, the setup could have been run by the verifier, with an additional proof of existence of α, such that $g = h^\alpha \bmod n$, to guarantee the hiding property. In this protocol, the verifier runs the setup:

- $\mathsf{Setup}(1^\kappa)$ runs $(n,(p,q)) \xleftarrow{\$} \mathsf{GenMod}(1^\kappa)$, and picks $h_0 \xleftarrow{\$} \mathsf{QR}_n$ and a random prime $\pi \in [\![2^{\kappa+1}\,;2^{\kappa+2}]\!]$. Then, it picks $\rho \xleftarrow{\$} [\![0\,;n^2]\!]_\pi$ and sets $g \leftarrow h_0^\rho \bmod n$ and $h \leftarrow h_0^\pi \bmod n$. Finally, it returns $\mathsf{pp} = (n,g,h)$ and keeps $\mathsf{sk} = (\pi, h_0)$. Actually, we have $h^\rho = g^\pi \bmod n$. So, if one sets $\alpha = \rho \cdot \pi^{-1} \bmod \varphi(n)$, one has $g = h^\alpha \bmod n$, and proves it;
- $\mathsf{Commit}(\mathsf{pp}, m; r)$ parses pp as above and commits to $m \in \mathbb{Z}$ by picking $r \xleftarrow{\$} \mathbb{Z}_{n/4}$ and computing $c = g^m h^r \bmod n$. It returns (c, r);
- $\mathsf{Verify}(\mathsf{pp}, c, m, r)$ parses $\mathsf{pp} = (n,g,h)$ and outputs 1 if $c = \pm g^m h^r \bmod n$ and 0 otherwise;
- $\mathsf{Reveal}(\mathsf{pp}, \mathsf{sk})$ returns $\mathsf{sk} = (\pi, h_0)$;
- $\mathsf{Adapt}(\mathsf{pp}, \mathsf{sk}, c, m, r)$ first parses $\mathsf{sk} = (\pi, h_0)$ and checks whether $h = h_0^\pi \bmod n$. Then, it adapts the opening by computing $m = k\pi + \bar{m}$ for $0 \leq \bar{m} < \pi$ and $t = g^k h_0^r \bmod n$. It outputs $(\bar{m}, t)$;
- $\mathsf{Verify}'(\mathsf{pp}, \pi, c, \bar{m}, t)$ outputs 1 if $c = g^{\bar{m}} t^\pi \bmod n$, and 0 otherwise.

This construction easily extends to commitments of vectors. Note that from $g^{\bar{m}} t^\pi = c = g^{\bar{m}'} t'^\pi \bmod n$, with $\bar{m} \neq \bar{m}' \bmod \pi$, setting $h_0 = y^2$ from an RSA challenge (n, y) of exponent $\pi > 2^\kappa$, we obtain $y^{2\rho(\bar{m}-\bar{m}')} = (t'/t)^\pi \bmod n$, with $2\rho(\bar{m} - \bar{m}') \neq 0 \bmod \pi$, which leads to the π-th root of y modulo n (using Proposition 2.2.9 from Section 2.2.2.2).

Switching between Commitments. Our goal is to use the more efficient commitment scheme of Gennaro, that we denote com_π, and also the associated proofs of relations in $\mathbb{Z}_\pi$: in the case of a single integer $m \in \mathbb{Z}_\pi$, $\mathsf{com}_\pi(m; r) = g^m r^\pi \bmod n$, for $r \xleftarrow{\$} \mathbb{Z}_n^*$. But let $\mathcal{V}$ run the setup from Section 6.2.2, which outputs $\mathsf{pp} = (n,g,h)$ (while keeping $\mathsf{sk} = (\pi, h_0)$), as in Section 5.2: this reveals no information about π (in an information-theoretic way). Now, $\mathcal{P}$ can use (n,g,h) for the Damgård-Fujisaki integer commitment scheme that we denote com: for an integer $m \in \mathbb{Z}$ and $r \xleftarrow{\$} \mathbb{Z}_n$, $c = \mathsf{com}(m; r) = g^m h^r \bmod n$. After some time, $\mathcal{V}$ reveals (π, h_0), which allows $\mathcal{P}$ to open c as a *commitment over* $\mathbb{Z}_\pi$ of $\mathfrak{r}_\pi(m) = m \bmod \pi$:

$$\mathsf{com}(m; r) = \mathsf{com}_\pi(\mathfrak{r}_\pi(m); g^{\mathfrak{q}_\pi(m)} h_0^r), \tag{6.1}$$

where $\mathfrak{q}_\pi(m)$ and $\mathfrak{r}_\pi(m)$ indeed denote the quotient and remainder of the euclidean division of m by π. This then allows to use efficient proofs on com_π, but still with good properties on the integers, since the prover did not know π at the commit time.

6.2.3 Improving Zero-Knowledge Arguments over the Integers

In this section, we introduce our new technique to build zero-knowledge arguments for statements over the integers, while using com_π. We restrict our attention to statements that can be expressed as membership to a set $S \in \mathbf{D}$. For preliminaries on the class $\mathbf{D}$ and on Diophantine equations, we refer the reader to Section 3.7. Our technique allows us to provide more efficient membership arguments, with a lower communication and a smaller verifier computation (applying the technique *delegates* some of the work of the verifier to the prover). As all the protocols considered in this paper, the protocol we describe is *honest-verifier zero-knowledge*, but can be improved to full-fledged zero-knowledge using standard methods (see the end of this section).

Membership Argument for D. Let us consider a set $S \in \mathbf{D}$ with representing polynomial P_S with k-vector input and ℓ-vector witness. We assume that $\mathcal{P}$ and $\mathcal{V}$ have agreed on a bound t such that each $\vec{x} \in S$ has a witness $\vec{w}$ of size $||\vec{w}||_1 \leq (||\vec{x}||_1)^t$ ($S \in \mathbf{D}$, so there is always such a t. As shown in [Lip03], $t < 2$ is sufficient for most cryptographic applications).

Let $\vec{x}$ be a secret vector held by $\mathcal{P}$, and $\vec{w}$ be a *witness* for $\vec{x} \in S$ (i.e., a vector satisfying $P_S(\vec{x}, \vec{w}) = 0$). It is known that zero-knowledge arguments can be constructed for polynomial relations over committed inputs (see e.g. [BS02]). Intuitively, this is done by committing to intermediate values, and proving additive and multiplicative relationships between those values and the inputs. To prove a multiplicative relationship $z = xy$ between values (x, y, z) committed in (c_x, c_y, c_z), $\mathcal{P}$ proves knowledge of inputs (x, y, z) and random coins (r_x, r_y, r_z) such that $c_x = g^x r_x^\pi \bmod n$, $c_y = g^y r_y^\pi \bmod n$, and $c_z = c_x^y r_z^\pi$. Let us now consider the following situation, where commitments are applied component-wise:

1. $\mathcal{P}$ picks random coins $(\vec{r_x}, \vec{r_w})$ and commits to $(\vec{x}, \vec{w})$ with $(\vec{r_x}, \vec{r_w})$ as $(\vec{c_x}, \vec{c_w}) \leftarrow (\mathsf{com}_\pi(\vec{x}; \vec{r_x}), \mathsf{com}_\pi(\vec{w}; \vec{r_w}))$;
2. $\mathcal{P}$ performs a zero-knowledge argument with $\mathcal{V}$ to prove his knowledge of four vectors $(\vec{x}, \vec{w}, \vec{r_x}, \vec{r_w})$ such that $(\vec{c_x}, \vec{c_w}) = (\mathsf{com}_\pi(\vec{x}; \vec{r_x}), \mathsf{com}_\pi(\vec{w}; \vec{r_w}))$ and $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$.

As com_π is a commitment scheme over $\mathbb{Z}_\pi$, this protocol is an argument of knowledge of $(\vec{x}, \vec{w})$ such that $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$. But this by no means proves the knowledge of *integers* belonging to the Diophantine set S. However, our main observation is that com_π can also be seen as an *integer commitment scheme* (the commitment scheme we denoted com).

Argument of knowledge of the inputs and witnesses.

1. $\mathcal{V}$ runs the setup from the Section 6.2.2, which generates $\mathsf{pp} = (n, g, h)$ and $\mathsf{sk} = (\pi, h_0)$: this defines $\mathsf{com} : (x; r) \mapsto g^x h^r \bmod n$. It additionally proves the existence of α such that $g = h^\alpha \bmod n$;
2. $\mathcal{P}$ picks random coins $(\vec{r_x}, \vec{r_w})$ and commits to $(\vec{x}, \vec{w})$ with $(\vec{r_x}, \vec{r_w})$ as $(\vec{c_x}, \vec{c_w}) \leftarrow (\mathsf{com}(\vec{x}; \vec{r_x}), \mathsf{com}(\vec{w}; \vec{r_w}))$;
3. $\mathcal{P}$ performs a $\mathsf{ZKAoK}\{(\vec{x}, \vec{w}, \vec{r_x}, \vec{r_w}) \mid \vec{c_x} = g^{\vec{x}} h^{\vec{r_x}} \wedge \vec{c_w} = g^{\vec{w}} h^{\vec{r_w}}\}$, we thereafter refer to ZK_1, with $\mathcal{V}$. If the argument fails, $\mathcal{V}$ aborts the protocol.

Argument of knowledge of $(\vec{x'}, \vec{w'})$ such that $P_S(\vec{x'}, \vec{w'}) = 0 \bmod \pi$.

1. $\mathcal{V}$ reveals (π, h_0) to $\mathcal{P}$ who checks whether $h = h_0^\pi \bmod n$ or not, to switch to com_π. Let $(\vec{x}', \vec{w}') = (\mathfrak{r}_\pi(\vec{x}), \mathfrak{r}_\pi(\vec{w})) = (\vec{x}, \vec{w}) \bmod \pi$.
2. $\mathcal{P}$ performs a $\mathsf{ZKAoK}\{(\vec{x}', \vec{w}', \vec{R_x}, \vec{R_w})\}$, we thereafter refer to ZK_2, such that $(\vec{c_x}, \vec{c_w}) = (\mathsf{com}_\pi(\vec{x}; \vec{R_x}), \mathsf{com}_\pi(\vec{w}; \vec{R_w}))$ and $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$. Note that $(\vec{c_x}, \vec{c_w})$ are now seen as commitments over $\mathbb{Z}_\pi$, using the fact that $\mathsf{com}(\vec{x}; \vec{r_x}) = \mathsf{com}_\pi(\mathfrak{r}_\pi(\vec{x}); \vec{R_x})$ and $\mathsf{com}(\vec{w}; \vec{r_w}) = \mathsf{com}_\pi(\mathfrak{r}_\pi(\vec{w}); \vec{R_w})$, with appropriate $(\vec{R_x}, \vec{R_w})$. If the argument succeeds, $\mathcal{V}$ returns accept.

Theorem 6.2.1. *Under the RSA assumption, the above protocol is a statistical zero-knowledge argument of knowledge of openings of $(\vec{c_x}, \vec{c_w})$ to vectors of integers $(\vec{x}, \vec{w})$ such that $P_S(\vec{x}, \vec{w}) = 0$: which proves that $\vec{x} \in S$.*

Proof. The intuition behind Theorem 6.2.1 is that ZK_1 proves that $\mathcal{P}$ knows $(\vec{x}, \vec{w})$ in $(\overrightarrow{c_x}, \overrightarrow{c_w})$, and ZK_2 proves that $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$ for a κ-bit prime π which was revealed *after* $(\vec{x}, \vec{w})$ *were committed*. Hence, $\mathcal{P}$ knew vectors of integer $(\vec{x}, \vec{w})$ such that $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$ for a random κ-bit prime π. This has a negligible probability to happen unless $P_S(\vec{x}, \vec{w}) = 0$ holds over the integers, since P_S is a polynomial. The full proof consists of the three properties: correctness, zero-knowledge, and knowledge-extractability.

Correctness. It easily follows from the correctness of ZK_1 and ZK_2: if $\mathcal{P}$ knows $(\vec{x}, \vec{w}, \overrightarrow{r_x}, \overrightarrow{r_w})$ such that $(\overrightarrow{c_x}, \overrightarrow{c_w}) = (\mathsf{com}(\vec{x}; \overrightarrow{r_x}), \mathsf{com}(\vec{w}; \overrightarrow{r_w}))$ and $P_S(\vec{x}, \vec{w}) = 0$, then the argument of knowledge of $(\vec{x}, \overrightarrow{r_x})$ such that $\overrightarrow{c_x} = \mathsf{com}(\vec{x}; \overrightarrow{r_x})$ will succeed, and it holds that $(\overrightarrow{c_x}, \overrightarrow{c_w}) = (\mathsf{com}_\pi(\vec{x} \bmod \pi; v^{\mathfrak{q}_\pi(\vec{x})}\tilde{h}^{\overrightarrow{r_x}}), \mathsf{com}_\pi(\vec{w} \bmod \pi; v^{\mathfrak{q}_\pi(\vec{x})}\tilde{h}^{\overrightarrow{r_x}}))$. Moreover, as P_S is a polynomial, the modular reduction applies, and leads to $P_S(\vec{x} \bmod \pi, \vec{w} \bmod \pi) = P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$.

Zero-Knowledge. It also follows from the zero-knowledge of ZK_1 and ZK_2, and the hiding property of the commitments. Let $\mathcal{S}im_{\mathsf{ZK}}$ be the following simulator: one first generates dummy commitments $(\overrightarrow{c_x}, \overrightarrow{c_w})$, which does not make any difference under the hiding property, and runs the simulator of ZK_1. Once (π, h_0) is revealed, $\mathcal{S}im_{\mathsf{ZK}}$ runs the simulator of ZK_2.

Since the commitment is statistically hiding, ZK_1 is our statistically zero-knowledge argument of knowledge of opening from Section 5.2 and ZK_2 is an argument of relations on commitments with known order π (since $h = h_0^\pi \bmod n$) that is possible in statistical zero-knowledge, the full protocol is statistically zero-knowledge.

Knowledge Extractability. Consider a $\mathcal{P}$' which succeeds in providing a convincing argument with probability ε, which means that the two protocols ZK_1 and ZK_2 succeed with probability greater than ε.

We first use the extractor of ZK_1 to extract the inputs-witnesses and random coins $(\vec{x}, \vec{w}, \overrightarrow{r_x}, \overrightarrow{r_w})$ such that $\overrightarrow{c_x} = g^{\vec{x}} h^{\overrightarrow{r_x}}$ and $\overrightarrow{c_w} = g^{\vec{w}} h^{\overrightarrow{r_w}}$. This extraction is successful under the RSA assumption.

Then, (π, h_0) is revealed and we use the extractor of ZK_2 to extract the inputs-witnesses and random coins $(\vec{x}', \vec{w}', \overrightarrow{R_x}, \overrightarrow{R_w})$ such that both relations $(\overrightarrow{c_x}, \overrightarrow{c_w}) = (\mathsf{com}_\pi(\vec{x}'; \overrightarrow{R_x}), \mathsf{com}_\pi(\vec{w}'; \overrightarrow{R_w}))$ and $P_S(\vec{x}', \vec{w}') = 0 \bmod \pi$ are satisfied. Again, this extraction is successful under the RSA assumption.

Now, let us consider two situations:

- If $\vec{x}' = \vec{x} \bmod \pi$ and $\vec{w}' = \vec{w} \bmod \pi$, then the value committed over the integers, *before* π *was revealed*, satisfy $P_S(\vec{x}, \vec{w}) = 0 \bmod \pi$, for a random $\pi \in [\![2^{\kappa+1}; 2^{\kappa+2}]\!]$. We stress that the view of (n, g, h) does not reveal any information on the prime π.

  Since there are approximately $2^{\kappa+1}/\kappa$ primes in this set, and this extraction works with probability greater than ε^2, $P_S(\vec{x}, \vec{w}) = 0 \bmod Q$, for $Q \geq 2^{2^\kappa/\varepsilon^2}$, which is much larger than the values that can be taken in the integers, since the inputs and the witnesses have a size polynomial in κ, and the polynomial P_S has a bounded degree.

- If $\vec{x}' \neq \vec{x} \bmod \pi$ or $\vec{w}' \neq \vec{w} \bmod \pi$, wlog, we can assume that $\vec{x}' \neq \vec{x} \bmod \pi$: one knows
  - $(\vec{x}, \overrightarrow{r_x})$ such that (1) $\overrightarrow{c_x} = \pm g^{\vec{x}} h^{\overrightarrow{r_x}} = g^{\mathfrak{r}_\pi(\vec{x})}(\pm g^{\mathfrak{q}_\pi(\vec{x})} h_0^{\overrightarrow{r_x}})^\pi \bmod n$;
  - and $(\vec{x}', \overrightarrow{R_x})$ such that (2) $\overrightarrow{c_x} = g^{\vec{x}'} \overrightarrow{R_x}^\pi \bmod n$.

  Hence, $g^{\mathfrak{r}_\pi(\vec{x})}(\pm g^{\mathfrak{q}_\pi(\vec{x})} h_0^{\overrightarrow{r_x}})^\pi = g^{\vec{x}'} \overrightarrow{R_x}^\pi \bmod n$, and so $g^{\mathfrak{r}_\pi(\vec{x}) - \vec{x}'} = S^\pi \bmod n$, for $S = \overrightarrow{R_x}/(\pm g^{\mathfrak{q}_\pi(\vec{x})} h_0^{\overrightarrow{r_x}}) \bmod n$. If one would have set $h_0 = y^2$ from an RSA challenge (n, y, π)

of exponent $\pi > 2^\kappa$, and thus $g = y^{2\rho}$, using Fact 2.2.9 from Proposition 2.2.7, one gets the π-th root of y modulo n.

This concludes the proof of the knowledge-extractability of the protocol, under the RSA assumption over $\mathbb{Z}_n$. □

On the Efficiency of the Method. The advantages of this method compared to the classical method are twofold. First, most of the work in the protocol comes from the computation of exponentiations; with our technique, most of the work is transfered from $\mathcal{V}$ to $\mathcal{P}$. This comes from the fact that verifying an equation such as $\overrightarrow{c} = \mathsf{com}(x; r)$ involves exponentiations by integers of size $O(\log n + \kappa)$ while verifying the equation $\overrightarrow{c} = \mathsf{com}_\pi(x \bmod \pi; R)$ involves only two exponentiations by κ-bit values, so the work of $\mathcal{V}$ is reduced. However, $\mathcal{P}$ will have to compute exponentiations by integers of size $O(\log n + \kappa)$ to construct the random coin R associated to the commitment mod π (using the identity 6.1 in Section 6.2.2). $\mathcal{V}$ will still need to perform exponentiations by integers during ZK_1, but his work during this step can be made essentially independent of the number N of inputs and witnesses (up to a small $\log N$ additive term) and completely independent of the degree of the representing polynomial.

Second, our method separates the argument of *knowledge* of inputs to a Diophantine equation from the argument that they do indeed satisfy the equation. The arguments of knowledge of an opening of a commitment can be very efficiently batched: if $\mathcal{P}$ commits to $(x_1, \cdots, x_N)$ with random coins $(r_1, \cdots, r_N)$ as $(c_1, \cdots, c_N)$, the verifier can simply send a random seed $\lambda \xleftarrow{\$} \{0,1\}^\kappa$ from which both players compute $(\lambda_1, \cdots, \lambda_N)$ using a pseudo-random generator[1]. Then, $\mathcal{P}$ performs a *single* argument of knowledge of an opening $(\sum_i \lambda_i x_i; \sum_i \lambda_i r_i)$ of the commitment $\prod_i c_i^{\lambda_i}$ (see [BGR98a; BGR98b] for more details). Therefore, when performing multiple membership arguments, $\mathcal{P}$ and $\mathcal{V}$ will have to perform a single argument for ZK_1 (of size essentially independent of the number of committed values).

In general, the higher the degree of the representing polynomial, the better our method will perform (in terms of communication). Still, we show in the following section that even for the case of range proofs, which can be seen as membership proofs to a Diophantine set whose representing polynomial is of degree 2, our method provides efficiency improvements.

Further Improvements. $\mathcal{V}$ can set h to $h_0^{\prod_i \pi_i}$ for several primes π_i instead of h^π. For some integer i, let $p_i \leftarrow \prod_{j \neq i} \pi_j$. Doing so allows $\mathcal{V}$ to reveal $(h_0^{p_i}, \pi_i)$ instead of (h_0, π) in our method. Hence, in addition to allowing arbitrary parrallel arguments with a single prime π, a single setting is sufficient to perform a polynomial number of sequential arguments (fixed in advance) with different primes π_i. In addition, we explained that commitments with knowledge-delayed order allow splitting the arguments of knowledge of the witnesses, denoted ZK_1, and the argument that they indeed belong to a Diophantine set, denoted ZK_2. The arguments ZK_1 can be batched as described above but, for efficiency reason, we should not generate $(\lambda_1, \lambda_2 \ldots, \lambda_N)$ as $(\lambda, \lambda^2, \ldots, \lambda^N)$. Indeed, $|\lambda^j|$ grows linearly with j over the integers. However, for the argument ZK_2, the order of the commitment has been revealed. Hence, we can now use batching technique with such $\lambda_j = \lambda^j$ since the prover is able to reduce the exponents modulo π at this stage. That means that our technique consisting of efficiently

[1] The classical trick that consists of using $\lambda_i = \lambda^i$ is not efficient here since we are in the integers, and so no reduction can be applied.

revealing the order of the commitment between ZK_1 and ZK_2 allows to use any tricks that were only available for discrete-log based proofs of statement over (pairing-free) known-order groups. For instance, we can get a sub-linear size argument to show that a committed matrix is the Hadamard products *over the integers* of two other committed matrices. Indeed, we can commit the rows of the matrices using a generalized commitment and make a batch proof for ZK_1, which remain sub-linear in the number of entries, and then we can import the results of [Gro09; BG12] to ZK_2, preserving its sub-linearity.

Full-Fledged Zero-Knowledge. With an honest verifier, there is no need to prove the existence of α such that $g = h^{\alpha}$. In the malicious setting, this proof guarantees the hiding property of the commitments to the prover, who additionally checks $h = h_0^{\pi} \bmod n$ when they are revealed. Then we can use classical techniques to convert the HVZK protocol into a ZK protocol, such as an equivocable commitment of the challenge by the verifier, before the commitments from the prover.

6.3 Application to Range Proofs

6.3.1 Lipmaa's Compact Argument for Positivity

As explained before, Lipmaa [Lip03] proposed an efficient argument for positivity, using generalized Damgård-Fujisaki commitments, and the proof that an integer is positive if and only if it can be written as the sum of four squares. However, it appears that the explicit construction given in [Lip03, annex B] is flawed — although the high-level description is correct: any prover can provide a convincing argument for positivity, regardless of the sign of the committed integer, and so without holding valid witnesses. This might raise some concerns as the protocol of Lipmaa is the 'textbook' range proof based on hidden order groups (the protocol is suggested in several papers, and was implemented in e.g. [AMA05]). For this reason, in Section 5.4.4, we recall the argument of [Lip03], identify its flaw, and provide a correct optimized version together with a full proof of security (the author of [Lip03] has been notified of this flaw).

In the following, we describe a range proof in the same vein as the positivity argument of Lipmaa: an integer x belongs to an interval $[\![a\,;b]\!]$ if and only if $(x-a)(b-x) \geq 0$. In addition, we take into account the following improvement suggested by Groth [Gro05]: x is positive if and only if $4x+1$ can be written as the sum of three squares, and such a decomposition can be computed in polynomial time by the prover. We view this range proof as an optimized version of the textbook range proof with integer commitments, to which we will compare our new method with knowledge-delayed order commitments.

6.3.2 Three-Square Range Proof

To prove that $x \in [\![a\,;b]\!]$, for x committed with an integer commitment scheme, we prove that $4(x-a)(b-x)+1$ can be written as the sum of three squares. Let (n, g, h) be the public parameters of the Damgård-Fujisaki commitment scheme, generated by the verifier. The three-square range proof ($\mathsf{3SRP}$) is described in full details on Figure 6.1. Basically, both $\mathcal{P}$ and $\mathcal{V}$ know that c_a contains $4(x-a)$ and c_0 contains $(b-x)$. The latter, with c_1, c_2, c_3 containing respectively x_1, x_2, x_3, is proven in a classical way, and the last part of the proof shows that $c_a^{x_0} g$, which implicitly contains $4(x-a)(b-x)+1$ also contains $x_1^2 + x_2^2 + x_3^2$.

For $\mathsf{pp} = (n, g, h)$ generated by $\mathcal{V}$, $\mathcal{P}$ has sent c, for which he knows (x, r) such that $c = g^x h^r \bmod n$ and $x \in [\![a\,;b]\!]$. Let $H : \mathbb{Z}_n^5 \mapsto \{0,1\}^{2\kappa}$ be a collision-resistant hash function. $\mathcal{V}$ compute $c_a = (cg^{-a})^4 \bmod n$ and $c_0 = c^{-1}g^b \bmod n$; $\mathcal{P}$ computes c_a.

1. $\mathcal{P}$ computes $(x_i)_{1\leq i\leq 3}$ such that $4(b-x)(x-a)+1 = \sum_{i=1}^{3} x_i^2$. $\mathcal{P}$ commits to $(x_i)_{1\leq i\leq 3}$ with random coins $(r_i)_{1\leq i\leq 3} \xleftarrow{\$} [\![0\,;n]\!]^3$ as $(c_i = g^{x_i}h^{r_i} \bmod n)_{1\leq i\leq 3}$. Let $x_0 \leftarrow (b-x)$ and $r_0 \leftarrow r$.
2. $\mathcal{P}$ picks $(m_0, \cdots, m_3) \xleftarrow{\$} [\![0\,;2^{B+2\kappa}]\!]^4$, $(s_0, \cdots, s_3) \xleftarrow{\$} [\![0\,;2^{2\kappa}n]\!]^4$, $\sigma \xleftarrow{\$} [\![0\,;2^{B+2\kappa}n]\!]$, and sends $\Delta = H((g^{m_i}h^{s_i} \bmod n)_{0\leq i\leq 3}, h^\sigma c_a^{m_0} \prod_{i=1}^{3} c_i^{-m_i} \bmod n)$.
3. $\mathcal{V}$ picks a challenge $e \xleftarrow{\$} [\![0\,;2^\kappa]\!]$ and sends it to $\mathcal{V}$.
4. $\mathcal{P}$ computes and sends $z_i = ex_i + m_i$ and $t_i = er_i + s_i$ for $i \in \{0,1,2,3\}$, and $\tau = \sigma + e(x_0 r_0 - \sum_{i=1}^{3} x_i r_i)$.
5. $\mathcal{V}$ accepts the argument if

$$\Delta = H\left((g^{z_i}h^{t_i}c_i^{-e} \bmod n)_{0\leq i\leq 3}, h^\tau g^e c_a^{z_0}\left(\prod_{i=1}^{3} c_i^{-z_i}\right) \bmod n\right).$$

Figure 6.1: Three-Square Range Proof (3SRP)

We then illustrate the technique introduced in Section 6.2.3 on this 3SRP protocol. The full converted protocol, denoted 3SRP-KDO, is described on Figure 6.2.

6.3.3 Results

Let $B = \log(b-a)$. Note that for all $i \in \{0,1,2,3\}$, $x_i^2 \leq (b-a)^2$ hence $\log x_i \leq B$. An exponentiation by a t-bit value takes $1.5t$ multiplications using a square-and-multiply algorithm; we do not take into account possible optimizations from multi-exponentiation algorithms. Table 6.1 sums up the communication complexity and the computational complexity of both the 3SRP and the 3SRP-KDO arguments for the execution of N parallel range proofs on the same interval $[\![a\,;b]\!]$, as classical batch techniques [BGR98a; BGR98b] allow to batch arguments of knowledge.

Note that we omit constant terms. The communication is given in bits, while the work is given as a number of multiplications of elements of QR_n. When comparing the work of the prover, we also omit the cost of the decomposition in sum of squares, as it is the same in both protocols. Similarly, we omit the cost of the initial proof of $g = h^\alpha \bmod n$ by the verifier to the prover.

Efficiency Analysis. We now provide a detailed comparison between the 3SRP and the 3SRP-KDO protocols. We set the order of the modulus n to 2048 bits and the security parameter κ to 128. As the communication of the protocols does also depend on the bound 2^B on the size of the interval, we consider various bounds in our estimation. For the sake of simplicity, we assume $B = \log b$.

Small Intervals and Large Intervals. As pointed out in [CCs08], most practical 'direct

For $\mathsf{pp} = (n, g, h)$ and $\mathsf{sk} = (\pi, h_0)$ generated by $\mathcal{V}$, $\mathcal{P}$ has sent c, for which he knows (x, r) such that $c = g^x h^r \bmod n$ and $x \in [\![a\,;b]\!]$. Let $H : \mathbb{Z}_n^6 \mapsto \{0,1\}^{2\kappa}$ be a collision-resistant hash function. $\mathcal{V}$ compute $c_a = (cg^{-a})^4 \bmod n$ and $c_0 = c^{-1}g^b \bmod n$; $\mathcal{P}$ computes c_a.

1. $\mathcal{P}$ computes $(x_i)_{1\leq i\leq 3}$ such that $4(b-x)(x-a)+1 = \sum_{i=1}^{3} x_i^2$. $\mathcal{P}$ commits to $(x_i)_{1\leq i\leq 3}$ with random coins $(r_i)_{1\leq i\leq 3} \xleftarrow{\$} [\![0\,;n]\!]^3$ as $(c_i = g^{x_i}h^{r_i} \bmod n)_{1\leq i\leq 3}$. Let $x_0 \leftarrow (b-x)$ and $r_0 \leftarrow r$.
2. $\mathcal{P}$ picks $m \xleftarrow{\$} [\![0\,;2^{B+3\kappa}]\!]$, $(m_0, \cdots, m_3) \xleftarrow{\$} [\![0\,;2^{\kappa}]\!]^4$, $s \xleftarrow{\$} [\![0\,;2^{3\kappa}n]\!]$, $(s_0, \cdots, s_3) \xleftarrow{\$} [\![0\,;n]\!]^4$, $\sigma \xleftarrow{\$} [\![0\,;2^{B+2\kappa}n]\!]$, and sends $\Delta = H(g^m h^s \bmod n, (g^{m_i}h^{s_i} \bmod n)_{0\leq i\leq 3}, h^{\sigma}c_a^{m_0}\prod_{i=1}^{3} c_i^{-m_i} \bmod n)$.
3. $\mathcal{V}$ picks a challenge $e' \xleftarrow{\$} [\![0\,;2^{\kappa}]\!]$ and sends (e', π, h_0) to $\mathcal{P}$.
4. $\mathcal{P}$ extends the challenge e' into $(e, (\lambda_i)_{0\leq i\leq 3}) \in [\![0\,;2^{\kappa}]\!]^5$, computes and sends $z = e\sum \lambda_i x_i + m$ and $t = e\sum \lambda_i r_i + s$, as well as $z_i = \mathfrak{r}_{\pi}(ex_i + m_i)$ and $T_i = h_0^{er_i+s_i} g^{\mathfrak{q}_{\pi}(ex_i+m_i)} \bmod n$ for $i \in \{0,1,2,3\}$, and $T = h_0^{\sigma+e(x_0r_0-\sum_{i=1}^{3} x_ir_i)} c_a^{\mathfrak{q}_{\pi}(ex_0+m_0)} \prod_{i=1}^{3} c_i^{-\mathfrak{q}_{\pi}(ex_i+m_i)} \bmod n$.
5. $\mathcal{V}$ accepts the argument if

$$\Delta = H\left(g^z h^t \left(\prod_{i=0}^{3} c_i^{\lambda_i}\right)^{-e} \bmod n, (g^{z_i}T_i^{\pi}c_i^{-e} \bmod n)_{i=0}^{3}, T^{\pi}g^e c_a^{z_0}\left(\prod_{i=1}^{3} c_i^{-z_i}\right) \bmod n\right)$$

Figure 6.2: Three-Square Range Proof with Knowledge-Delayed Order (3SRP-KDO)

applications' of range proofs, such as e-voting [Gro05] and e-cash [CHL05], involve quite small intervals (say, of size at most 2^{30}, and so $B \leq 30$). However, when range proofs are used instead as a basis to construct cryptographic schemes, very large intervals are commonly involved. Examples include anonymous credentials [CL01], mutual private set intersection protocols [KLC12], secure generation of RSA keys [JG02; DM10], zero-knowledge primality tests [CM99a], and some protocols for performing non-arithmetic operations on Paillier ciphertexts [GMS10; CPP15a]. In such protocols, B typically range from 1024 to 4096 (and is even larger is some cases). We note that such intervals are exactly the ones for which range proofs based on groups of hidden order are likely to be used, since for for small intervals, protocols based on some u-ary decomposition of the input [CCs08; Gro11] will in general have better performances (essentially because they avoid the need of the Rabin-Shallit algorithm, which is computationally involved).

Comparisons. Table 6.2 gives a summary of our results. As already noted, the overhead of the work of the prover in 3SRP-KDO is measured by comparing the works *without considering the cost of the Rabin-Shallit algorithm*; the latter one, however, is by far the dominant cost when B is large (as it runs in expected $O(B^2 \log B \cdot M(\log B))$ time, where $M(\log B)$ is the time taken to perform a multiplication of $(\log B)$-bit integers). Therefore, for a large B, the overhead of the work of the prover in 3SRP-KDO is very small, whereas there is a huge gain for the verifier. As expected, the 3SRP-KDO protocol provides interesting performances in

	3SRP	3SRP-KDO
Communication (in bits)	$N(8\log n + 18\kappa + 5B) + 3\kappa$	$N(8\log n + 4\kappa) + 10\kappa + 2\log n + B + \log N$
Prover's work (exponentiations)	$1.5N(8\log n + 12B + 26\kappa + \log a)$	$1.5(N(13\log n + 13B + 18\kappa + \log a) + \log n + B + 6\kappa + \log N)$
Verifier's work (exponentiations)	$1.5(N(5\log n + 9B + 30\kappa + \log a + \log b) + \kappa)$	$1.5(N(12\kappa + \log a + \log b) + \log n + B + 10\kappa + \log N)$

Table 6.1: Complexities of 3SRP and 3SRP-KDO

	communication overhead	prover's work overhead	verifier's work overhead
$B = 30, N = 1$	+16%	+60.2%	−66%
$B = 1024, N = 1$	−3.7%	+44%	−71.7%
$B = 2048, N = 1$	−17%	+36.4%	−74.1%
$B = 30, N = 10$	−7.6%	+47.5%	−86.8%
$B = 1024, N = 10$	−26.5%	+33.2%	−87.7%
$B = 2048, N = 10$	−39.1%	+26.5%	−88%

This is for various interval sizes (2^B) and numbers N of parallel executions
Percentages indicate $100 \times (\mathsf{cost}(\text{3SRP-KDO}) - \mathsf{cost}(\text{3SRP}))/\mathsf{cost}(\text{3SRP})$, where prover's cost does not consider the 3-square decomposition.

Table 6.2: Comparison between the 3SRP and the 3SRP-KDO

settings where:

- The verifier is computationally weak (e.g. in secure Cloud computing), and/or
- Multiples range proofs are likely to be used in parallel, and/or
- The intervals are large.

7

Conclusion and Open Questions

There are no problems, just pauses between ideas.
– The Brotherhood of the Rose

I have not failed. I've just found 10,000 ways that won't work.
– Thomas Edison

7.1 Conclusion

The contributions presented in this thesis focus on the design and the analysis of zero-knowledge systems, and target their applications to secure computation. To this aim, we have developed a new type of zero-knowledge proof system, called implicit zero-knowledge arguments, which can be seen as a weak form of (designated-verifier) non-interactive zero-knowledge. Unlike classical non-interactive zero-knowledge, implicit zero-knowledge arguments can be based on a large variety of standard assumption, and in particular, do not require pairings. They also have better computational and communication efficiency, and can therefore advantageously replace them in round-efficient two-party computation.

We have also studied zero-knowledge argument systems over the integers, which have found applications in a variety of secure computation protocols such as electronic voting, e-cash, and anonymous credentials. Our main contribution is a new security analysis of the existing argument systems over the integers which shows that, unlike what had been assumed for the last two decades, their security can be based directly on the well-studied RSA assumption. Our proof involves novel ideas, and essentially all applications of zero-knowledge arguments over the integers benefit from our improved analysis.

In addition, we have constructed new arguments over the integers, which provide communication improvements in some scenarios, and which strongly reduce the work of the verifier in general, making them well-suited for use in client-server settings for secure computation.

7.2 Open Questions

Our work on implicit zero-knowledge arguments aims at providing an efficient alternative to NIZKs for round-efficient secure computation, under well-studied assumptions. Still, publicly-verifiable NIZK proof systems have a wider range of applications, and are a core primitive in

cryptography. Therefore, we believe that it is a problem of great interest to find out new efficient constructions of NIZKs, under standard assumptions. This leads us to a first open question:

Question 7.1. *Is it possible to build efficient publicly-verifiable non-interactive zero-knowledge proof (or argument) systems, without pairing-based assumptions, in the standard model?*

We point out that efficient NIZKs are known in the random oracle model (using the Fiat-Shamir heuristic), and inefficient NIZKs are known from (strong variants of) trapdoor permutations. However, apart from the breakthrough work of Groth and Sahai on pairing-based NIZKs, the research on efficient NIZKs in the standard model has proven elusive. In fact, using DDH-like assumptions, or lattice-based assumptions, we do not even know inefficient constructions. This suggests an alternative open question, which is more of theoretical interest, as it calls for a better understanding of the structure of non-interactive zero-knowledge:

Question 7.2. *Is it possible to build (possibly inefficient) public-verifiable non-interactive zero-knowledge proof (or argument) systems in the standard model, under DDH-like assumption (in pairing-free groups) or lattice-based assumptions?*

We note that a candidate answer to the above question based on indistinguishability obfuscation has been given in [BP15]. However, indistinguishability obfuscation implies the existence of multilinear maps where the multilinear analogue of DDH holds [AFH+16]; it therefore implies in particular the existence of pairing-friendly groups in which the Groth-Sahai methodology can be instantiated.

Implicit zero-knowledge arguments can be seen as a weak type of designated-verifier NIZKs (by interpreting the first flow as a word-dependent common reference string generated by the verifier). Whether full-fledged designated-verifier NIZK proof systems can be constructed without pairing-based assumptions seems to be a more tractable question, which remains a very interesting one in our opinion.

Question 7.3. *Is it possible to build efficient designated-verifier non-interactive zero-knowledge proof (or argument) systems in the standard model, under well-studied assumptions, for natural and interesting languages?*

Some results have already been obtained in this direction [DFN06; CG15], but numerous questions have been left open. In particular, designated-verifier NIZKs with unbounded soundness (where the soundness holds even if the prover received polynomially feedbacks on previous proofs) are only known in the generic group model, and while efficient designated-verifier NIZK *arguments* are known for interesting languages, we do not yet know of efficient designated-verifier NIZK *proofs*.

Other iZK-related questions can be envisioned, such as studying the relations of iZK to other cryptographic primitives (in particular, iZK seems to be closely-related to dual-mode encryption [PVW08]), and generalizing the results of Chapter 4 to the multiparty setting.

Turning our attention to zero-knowledge arguments for integer commitment schemes, let us quote a paragraph from the introduction of [CDP12]:

‘A multiplication protocol for integer commitments was proposed in [FO97; DF02]. This protocol has essentially optimal communication complexity $\Theta(\kappa + \ell + k)$, where k is the size in bits of the prover’s secret integers, but it requires an extra assumption, namely, the

strong RSA assumption. If we only want to assume what the commitment scheme requires (factoring), the best known complexity is $\Theta((\kappa + k)\ell)$.'

This observation was used in [CDP12] to motivate the design of efficient zero-knowledge proofs of integer commitment schemes in an *amortized* setting, where many proofs are performed, under the minimal assumptions required by the scheme. Our work in Chapter 5 improves this state of affair, by showing that the standard RSA assumption does in fact suffice for the zero-knowledge argument mentioned above (where amortization is not necessary to get an efficient proof). However, the interesting question of designing an efficient zero-knowledge argument system for relations between integer commitments under *minimal* assumptions remain open:

Question 7.4. *Is it possible to build an efficient zero-knowledge argument system for integer commitment schemes whose knowledge-extractability property can be based on the factorization assumption?*

More broadly, and without pointing specific open questions, RSA groups enjoy many non-trivial features that we usually expect to see in pairing-friendly groups – they allow for the construction of NIZKs (albeit inefficient ones), and identity-based encryption schemes [Coc01], both of which are more commonly built in pairing-friendly groups. It is also possible to build some form of (designated-verifier) decision Diffie-Hellman oracle in RSA groups, under the factorization assumption [HK09], which is again a natural feature of (symmetric) pairing-friendly groups. It seems likely that the rich structure of RSA groups remains insufficiently explored, and that many more applications that are known from pairing-friendly groups could be built in RSA groups.

Notation

Mathematical Notations

$\mathbb{N}$	set of non-negative integers
$\mathbb{Z}$	set of integers
$[\![a\,;b]\!]$	integer interval
$\{0,1\}^k$	set of length-k bitstrings
$\lvert s\rvert$	bit-length of s
$\lvert\lvert s\rvert\rvert$	absolute value of s
$(\mathbb{Z}_k,+)$	additive group of integers modulo k
$(\mathbb{Z}_k^*,\cdot)$	multiplicative groups of invertible integers modulo k
$(\mathbb{Z}_k,+,\cdot)$	ring of integers modulo k
$\mathbb{G}$	cyclic group
$\overrightarrow{x}$	row vector
$x \xleftarrow{\$} S$	uniformly random assignment of an element of S to x
p,q	primes
φ	Euler totient function
$\mathbb{J}_n$	the group of elements of $\mathbb{Z}_n^*$, with $n=pq$, with Jacobi symbol 1
QR_n	the group of quadratic residues modulo $n=pq$

Algorithms and Complexity

$y \leftarrow A(x)$	y is the output of the algorithm A on input x
$y \xleftarrow{\$} A(x)$	output of A on x, where A is a randomized algorithm
st	state of an algorithm
$\mathscr{L}$	language
P	the class of problems decidable by polytime algorithms
BPP	the class of problems decidable by randomized polytime algorithms with bounded error
NP	the class of decision problems with efficiently verifiable solutions
IP	the class of decision problems with interactive proofs
PSPACE	the class of problems decidable by polynomial space algorithms
PZK, SZK, CZK	the class of problems admitting a (perfect,statistical,computational) zero-knowledge proof
$(\mathsf{P},\mathsf{S},\mathsf{C})\mathsf{ZKA}$	the class of problems admitting a (perfect,statistical,computational) zero-knowledge argument
$(\mathsf{P},\mathsf{S},\mathsf{C})\mathsf{ZKPoK}$	the class of problems admitting a (perfect,statistical,computational) zero-knowledge proof of knowledge
$\mathrm{negl}(x)$	a negligible function in x
κ	the security parameter
$\mathscr{A}$, $\mathscr{A}^{\mathsf{O}}$	an adversary, without and with oracle access to O
$\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}$	experiment `sec` with adversary $\mathscr{A}$
$\mathsf{Succ}^{\mathsf{sec}}(\mathscr{A},\kappa)$	success of $\mathscr{A}$ in the experiment `sec`
$\mathsf{Adv}^{\mathsf{sec}}(\mathscr{A},\kappa)$	advantage of $\mathscr{A}$ in experiment `sec`

Abbreviations

Mathematical Concepts

CRT	Chinese Remainder Theorem

Assumptions

CDH	Computational Diffie-Hellman
DDH	Decisional Diffie-Hellman
DLIN	Decision Linear
MDDH	Matrix Decisional Diffie-Hellman
RSA	Rivest-Shamir-Adleman

Cryptographic Notions

2PC	Two-party Computation
MPC	Multiparty Computation
iZK	Implicit Zero-Knowledge
SSiZK	Simulation-Sound Zero-Knowledge
PPT	Probabilistic Polynomial-Time
IND-CPA	Indistiguishability against Chosen Plaintext Attacks
IND-CCA	Indistiguishability against Chosen Ciphertext Attacks
SPHF	Smooth Projective Hash Function
KV-SPHF	Katz-Vaikuntanathan SPHF
GL-SPHF	Gennaro-Lindell SPHF
CS-SPHF	Cramer-Shoup SPHF
ZK	Zero-Knowledge
HVZK	Honest-Verifier Zero-Knowledge
NIZK	Non-Interactive Zero-Knowledge
CRS	Common Reference String
ABP	Arithmetic Branching Program

List of Illustrations

Figures

2.1 Template for representing an experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathscr{A}}(1^\kappa)$ with a challenger and an adversary $\mathscr{A}$ . . . 17
2.2 Experiment $\mathsf{Exp}^{\mathsf{dlog}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the discrete logarithm problem over a group $\mathbb{G}$ of order p with generator g . . . 18
2.3 Experiment $\mathsf{Exp}^{\mathsf{cdh}}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the computational Diffie-Hellman problem over a group $\mathbb{G}$ with generator g . . . 20
2.4 Experiments $\mathsf{Exp}^{\mathsf{ddh}-0}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ and $\mathsf{Exp}^{\mathsf{ddh}-1}_{\mathscr{A}}(\mathbb{G}, g, 1^\kappa)$ for the decisional Diffie-Hellman problem over a group $\mathbb{G}$ with generator g . . . 21
2.5 Experiments $\mathsf{Exp}^{\mathsf{dlin}-0}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$ and $\mathsf{Exp}^{\mathsf{dlin}-1}_{\mathscr{A}}(\mathbb{G}, g, h, 1^\kappa)$ for the decision linear assumption over a group $\mathbb{G}$ with generators (g, h) . . . 21
2.6 Experiment $\mathsf{Exp}^{\mathsf{fact}}_{\mathscr{A}}(1^\kappa)$ for the factorization assumption . . . 23
2.7 Experiment $\mathsf{Exp}^{\mathsf{rsa}}_{\mathscr{A}}(1^\kappa, \mathtt{Dist}_n)$ for the RSA family of assumptions, parametrized by a distribution $\mathtt{Dist}_n$ over P_n . . . 24
2.8 Experiment $\mathsf{Exp}^{\mathsf{srsa}}_{\mathscr{A}}(1^\kappa)$ for the Strong-RSA assumption . . . 25
2.9 Experiments $\mathsf{Exp}^{\mathsf{hiding}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathsf{hiding}-1}_{\mathscr{A}}(1^\kappa)$ for the hiding property of a commitment scheme Π . . . 27
2.10 Experiment $\mathsf{Exp}^{\mathsf{binding}}_{\mathscr{A}}(1^\kappa)$ for the binding property of a commitment scheme Π 28
2.11 Experiments $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-0}_{\mathscr{A}}(1^\kappa)$ and $\mathsf{Exp}^{\mathtt{ind\text{-}cpa}-1}_{\mathscr{A}}(1^\kappa)$ for the IND-CPA security property of a public-key encryption scheme Π . . . 29

3.1 The Schnorr Σ-protocol for proving knowledge of a discrete logarithm . . . 47
3.2 Σ-protocol for proving knowledge of a witness for a DDH tuple . . . 49
3.3 Σ-protocol for proving the disjunction of statements $S = S_0 \vee S_1$ . . . 50
3.4 The common reference string ideal functionality [CF01] . . . 51

4.1 Enforcing semi-honest behavior of Alice (A) . . . 63
4.2 Experiments $\mathsf{Exp}^{\mathtt{iZK\text{-}zk}-b}$ for zero-knowledge of iZK, and $\mathsf{Exp}^{\mathtt{iZK\text{-}ss}-b}$ for simulation-soundness of SSiZK . . . 68
4.3 Three-round zero-knowledge from iZK for a word $\mathsf{x} \in \mathsf{i}\mathscr{L}$ and a witness iw . . 69
4.4 Construction of iZK . . . 74
4.5 Experiments $\mathsf{Exp}^{\mathtt{iZK\text{-}zk}-b}$ for zero-knowledge of iZK . . . 79
4.6 Semi-honest to malicious compilers . . . 86
4.7 Semi-Honest and Malicious Protocols for Secure Inner Product Computation 88
4.8 Ideal Functionality for Inner Product $\mathcal{F}_{\mathsf{IP}}$ . . . 95

5.1 Distributions for the Zero-Knowledge Property . . . 104
5.2 Lipmaa's Compact Argument for Positivity . . . 112
5.3 Variant of Lipmaa's Compact Argument for Positivity . . . 113

6.1 Three-Square Range Proof (3SRP) 123
6.2 Three-Square Range Proof with Knowledge-Delayed Order (3SRP-KDO) . . . 124

Tables

4.1 Comparison of the costs of various approaches for UC-secure two-party computation of the inner product 88
4.2 Costs for computing exponentiations and pairings in different curves 88

6.1 Complexities of 3SRP and 3SRP-KDO 125
6.2 Comparison between the 3SRP and the 3SRP-KDO 125

Bibliography

[Aar13] Scott Aaronson. *Quantum computing since Democritus*. Cambridge University Press, 2013 (cit. on p. 3).

[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 pp. 64, 65).

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Disjunctions for Hash Proof Systems: New Constructions and Applications". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 69–100. DOI: 10.1007/978-3-662-46803-6_3 (cit. on pp. 65, 71).

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. *A New Approach to Round-Optimal Secure Multiparty Computation*. Cryptology ePrint Archive, Report 2017/402. http://eprint.iacr.org/2017/402. 2017 (cit. on p. 66).

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. "Smooth Projective Hashing for Conditionally Extractable Commitments". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 671–689 (cit. on p. 65).

[AFH+16] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. "Multilinear Maps from Obfuscation". In: *TCC 2016-A, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 446–473. DOI: 10.1007/978-3-662-49096-9_19 (cit. on p. 128).

[AFK87] Martín Abadi, Joan Feigenbaum, and Joe Kilian. "On Hiding Information from an Oracle (Extended Abstract)". In: *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 195–203 (cit. on p. 18).

[AH91] William Aiello and Johan Hastad. "Statistical zero-knowledge languages can be recognized in two rounds". In: *Journal of Computer and System Sciences* 42.3 (1991), pp. 327–345 (cit. on p. 42).

[AHL05] Luis von Ahn, Nicholas J. Hopper, and John Langford. "Covert two-party computation". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 513–522 (cit. on p. 10).

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. "Priced Oblivious Transfer: How to Sell Digital Goods". In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 119–135 (cit. on p. 66).

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108 (cit. on p. 17).

[AM76] Leonard Adleman and Kenneth Manders. "Diophantine Complexity". In: *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*. SFCS '76. Washington, DC, USA: IEEE Computer Society, 1976, pp. 81–88. DOI: 10.1109/SFCS.1976.13. URL: http://dx.doi.org/10.1109/SFCS.1976.13 (cit. on p. 54).

[AMA05] Adelsbach André, Rohe Markus, and Sadeghi Ahmad-Reza. "Non-interactive Watermark Detection for a Correlation-Based Watermarking Scheme". In: *Communications and Multimedia Security: 9th IFIP TC-6 TC-11 International Conference, CMS 2005*. Ed. by springer. 2005, pp. 129–139 (cit. on p. 122).

[ARS04] André Adelsbach, Markus Rohe, and Ahmad-Reza Sadeghi. "Overcoming the obstacles of zero-knowledge watermark detection". In: *Proceedings of the 2004 workshop on Multimedia and security*. ACM. 2004, pp. 46–55 (cit. on p. 101).

[Bab85] László Babai. "Trading Group Theory for Randomness". In: *17th ACM STOC*. ACM Press, May 1985, pp. 421–429 (cit. on pp. 37–39).

[BB04] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 56–73 (cit. on p. 101).

[BBC+13] 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. 31, 32, 64, 65, 69, 72).

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. "An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 171–188 (cit. on p. 62).

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 41–55 (cit. on p. 21).

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2016, pp. 327–357 (cit. on p. 44).

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum Disclosure Proofs of Knowledge". In: *J. Comput. Syst. Sci.* 37.2 (1988), pp. 156–189. DOI: 10.1016/0022-0000(88)90005-0. URL: http://dx.doi.org/10.1016/0022-0000(88)90005-0 (cit. on p. 44).

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. "Secure Multiparty Computation Goes Live". In: *FC 2009*. Ed. by Roger Dingledine and Philippe Golle. Vol. 5628. LNCS. Springer, Heidelberg, Feb. 2009, pp. 325–343 (cit. on p. 5).

[BCDv88] Ernest F. Brickell, David Chaum, Ivan Damgård, and Jeroen van de Graaf. "Gradual and Verifiable Release of a Secret". In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Heidelberg, Aug. 1988, pp. 156–166 (cit. on p. 101).

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. "Homomorphic Secret Sharing: Optimizations and Applications". In: *ACM CCS 17*. ACM Press, 2017, pp. 1–2 (cit. on pp. 9, 10).

[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. 6, 73).

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. "Non-interactive zero-knowledge". In: *SIAM Journal on Computing* 20.6 (1991), pp. 1084–1118 (cit. on p. 7).

[Ben16] Fabrice Benhamouda. "Diverse modules and zero-knowledge". PhD thesis. PSL Research University, 2016 (cit. on pp. 69, 73).

[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. 19, 61, 62, 88, 89).

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 213–229 (cit. on p. 22).

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. *Batch Groth-Sahai*. Cryptology ePrint Archive, Report 2010/040. http://eprint.iacr.org/2010/040. 2010 (cit. on pp. 57, 92).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112 (cit. on pp. 7, 56).

[BG12] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 263–280 (cit. on p. 122).

[BGG+90] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. "Everything Provable is Provable in Zero-Knowledge". In: *CRYPTO'88*. Ed. by Shafi Goldwasser. Vol. 403. LNCS. Springer, Heidelberg, Aug. 1990, pp. 37–56 (cit. on p. 42).

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Breaking the circuit size barrier for secure computation under DDH". In: *Annual Cryptology Conference*. Springer. 2016, pp. 509–539 (cit. on p. 9).

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Group-Based Secure Computation: Optimizing Rounds, Communication, and Computation". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 163–193 (cit. on pp. 9, 10).

[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 p. 62).

[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 pp. 88, 89).

[BGR98a] Mihir Bellare, Juan A. Garay, and Tal Rabin. "Batch Verification with Applications to Cryptography and Checking". In: *LATIN 1998*. Ed. by Claudio L. Lucchesi and Arnaldo V. Moura. Vol. 1380. LNCS. Springer, Heidelberg, Apr. 1998, pp. 170–191 (cit. on pp. 121, 123).

[BGR98b] Mihir Bellare, Juan A. Garay, and Tal Rabin. "Fast Batch Verification for Modular Exponentiation and Digital Signatures". In: *EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, May 1998, pp. 236–250 (cit. on pp. 121, 123).

[BHJ+13] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Jae Hong Seo, and Christoph Striecks. "Practical Signatures from Standard Assumptions". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 461–485. DOI: 10.1007/978-3-642-38348-9_28 (cit. on p. 102).

[BKK+15] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. *Students and Taxes: a Privacy-Preserving Social Study Using Secure Computation*. Cryptology ePrint Archive, Report 2015/1159. http://eprint.iacr.org/2015/1159. 2015 (cit. on p. 5).

[BLS11] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. "On the Correct Use of the Negation Map in the Pollard rho Method". In: *PKC 2011*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. LNCS. Springer, Heidelberg, Mar. 2011, pp. 128–146 (cit. on p. 19).

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. “Lower Bounds for Non-Black-Box Zero Knowledge”. In: *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 384–393 (cit. on p. 50).

[BLW08] Dan Bogdanov, Sven Laur, and Jan Willemson. “Sharemind: A Framework for Fast Privacy-Preserving Computations”. In: *ESORICS 2008*. Ed. by Sushil Jajodia and Javier López. Vol. 5283. LNCS. Springer, Heidelberg, Oct. 2008, pp. 192–206 (cit. on p. 5).

[Bou00] Fabrice Boudot. “Efficient Proofs that a Committed Number Lies in an Interval”. In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 431–444 (cit. on pp. 54, 101).

[BP15] Nir Bitansky and Omer Paneth. “ZAPs and Non-Interactive Witness Indistinguishability from Indistinguishability Obfuscation”. In: *TCC 2015, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. LNCS. Springer, Heidelberg, Mar. 2015, pp. 401–427. DOI: 10.1007/978-3-662-46497-7_16 (cit. on p. 128).

[BP97] Niko Bari and Birgit Pfitzmann. “Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees”. In: *EUROCRYPT’97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 480–494 (cit. on pp. 25, 100).

[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 pp. 64, 65).

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. “How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios”. In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 626–643. DOI: 10.1007/978-3-642-34961-4_38 (cit. on p. 61).

[BR09] Mihir Bellare and Thomas Ristenpart. “Simulation without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters’ IBE Scheme”. In: *EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, Heidelberg, Apr. 2009, pp. 407–424 (cit. on p. 77).

[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 pp. 4, 7, 57, 60).

[Bro13] Jon Brodkin. *Satellite Internet faster than advertised, but latency still awful*. http://arstechnica.com/information-technology/2013/02/satellite-internet-faster-than-advertised-but-latency. Feb. 2013 (cit. on p. 61).

[Bro16] Daniel R. L. Brown. “Breaking RSA May Be As Difficult As Factoring”. In: *Journal of Cryptology* 29.1 (Jan. 2016), pp. 220–241. DOI: 10.1007/s00145-014-9192-y (cit. on p. 25).

[BS02] Emmanuel Bresson and Jacques Stern. "Proofs of Knowledge for Non-monotone Discrete-Log Formulae and Applications". In: *ISC 2002*. Ed. by Agnes Hui Chan and Virgil D. Gligor. Vol. 2433. LNCS. Springer, Heidelberg, Sept. 2002, pp. 272–288 (cit. on p. 119).

[BTW12] Dan Bogdanov, Riivo Talviste, and Jan Willemson. "Deploying Secure Multi-Party Computation for Financial Data Analysis - (Short Paper)". In: *FC 2012*. Ed. by Angelos D. Keromytis. Vol. 7397. LNCS. Springer, Heidelberg, Feb. 2012, pp. 57–64 (cit. on p. 5).

[BV98] Dan Boneh and Ramarathnam Venkatesan. "Breaking RSA May Not Be Equivalent to Factoring". In: *EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, May 1998, pp. 59–71 (cit. on p. 25).

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *Journal of Cryptology* 13.1 (2000), pp. 143–202 (cit. on p. 94).

[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 p. 50).

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. "Efficient Protocols for Set Membership and Range Proofs". In: *ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Vol. 5350. LNCS. Springer, Heidelberg, Dec. 2008, pp. 234–252 (cit. on pp. 101, 123, 124).

[CCT07] Sébastien Canard, Iwen Coisel, and Jacques Traoré. "Complex Zero-Knowledge Proofs of Knowledge Are Easy to Use". In: *ProvSec 2007*. Ed. by Willy Susilo, Joseph K. Liu, and Yi Mu. Vol. 4784. LNCS. Springer, Heidelberg, Nov. 2007, pp. 122–137 (cit. on pp. 101, 111).

[CD04] Ronald Cramer and Ivan Damgård. "Secret-Key Zero-Knowlegde and Non-interactive Verifiable Exponentiation". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Heidelberg, Feb. 2004, pp. 223–237 (cit. on p. 56).

[CDP12] Ronald Cramer, Ivan Damgård, and Valerio Pastro. "On the Amortized Complexity of Zero Knowledge Protocols for Multiplicative Relations". In: *ICITS 12*. Ed. by Adam Smith. Vol. 7412. LNCS. Springer, Heidelberg, Aug. 2012, pp. 62–79. DOI: 10.1007/978-3-642-32284-6_4 (cit. on pp. 128, 129).

[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 p. 48).

[CF01] Ran Canetti and Marc Fischlin. "Universally Composable Commitments". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 19–40 (cit. on p. 51).

[CFT98] Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. "Easy Come - Easy Go Divisible Cash". In: *EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. LNCS. Springer, Heidelberg, May 1998, pp. 561–575 (cit. on p. 101).

[CG08] Jan Camenisch and Thomas Groß. “Efficient attributes for anonymous credentials”. In: *ACM CCS 08*. Ed. by Peng Ning, Paul F. Syverson, and Somesh Jha. ACM Press, Oct. 2008, pp. 345–356 (cit. on p. 101).

[CG15] Pyrros Chaidos and Jens Groth. “Making Sigma-Protocols Non-interactive Without Random Oracles”. In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Heidelberg, Mar. 2015, pp. 650–670. DOI: 10.1007/978-3-662-46447-2_29 (cit. on p. 128).

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited”. In: *J. ACM* 51.4 (July 2004), pp. 557–594. ISSN: 0004-5411. DOI: 10.1145/1008731.1008734. URL: http://doi.acm.org/10.1145/1008731.1008734 (cit. on p. 62).

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited (Preliminary Version)”. In: *30th ACM STOC*. ACM Press, May 1998, pp. 209–218 (cit. on p. 7).

[CGOS07] Nishanth Chandran, Vipul Goyal, Rafail Ostrovsky, and Amit Sahai. “Covert Multi-Party Computation”. In: *48th FOCS*. IEEE Computer Society Press, Oct. 2007, pp. 238–248 (cit. on pp. 10, 66).

[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. 94).

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. “Compact E-Cash”. In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 302–321 (cit. on pp. 101, 124).

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. “Black-box concurrent zero-knowledge requires Omega (log n) rounds”. In: *33rd ACM STOC*. ACM Press, July 2001, pp. 570–579 (cit. on p. 50).

[CL01] Jan Camenisch and Anna Lysyanskaya. “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation”. In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, Heidelberg, May 2001, pp. 93–118 (cit. on pp. 101, 124).

[CM99a] Jan Camenisch and Markus Michels. “Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes”. In: *EUROCRYPT’99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Heidelberg, May 1999, pp. 107–122 (cit. on pp. 101, 124).

[CM99b] Jan Camenisch and Markus Michels. “Separability and Efficiency for Generic Group Signature Schemes”. In: *CRYPTO’99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 413–430 (cit. on p. 101).

[Coc01] Clifford Cocks. “An Identity Based Encryption Scheme Based on Quadratic Residues”. In: *8th IMA International Conference on Cryptography and Coding*. Ed. by Bahram Honary. Vol. 2260. LNCS. Springer, Heidelberg, Dec. 2001, pp. 360–363 (cit. on p. 129).

[Cou16a] Geoffroy Couteau. *Efficient Secure Comparison Protocols*. Cryptology ePrint Archive, Report 2016/544. http://eprint.iacr.org/2016/544. 2016 (cit. on p. 10).

[Cou16b] Geoffroy Couteau. *Revisiting Covert Multiparty Computation*. Cryptology ePrint Archive, Report 2016/951. http://eprint.iacr.org/2016/951. 2016 (cit. on p. 10).

[CPP15a] Geoffroy Couteau, Thomas Peters, and David Pointcheval. *Encryption Switching Protocols*. Cryptology ePrint Archive, Report 2015/990. http://eprint.iacr.org/. 2015 (cit. on pp. 101, 124).

[CPP15b] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Secure Distributed Computation on Private Inputs". In: *International Symposium on Foundations and Practice of Security*. Springer. 2015, pp. 14–26 (cit. on p. 9).

[CPP16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Encryption Switching Protocols". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 308–338. DOI: 10.1007/978-3-662-53018-4_12 (cit. on p. 9).

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. "Removing the Strong RSA Assumption from Arguments over the Integers". In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Heidelberg, May 2017, pp. 321–350 (cit. on p. 6).

[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. 7, 30, 31, 63–65).

[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. 65, 87).

[Dam00] Ivan Damgård. "Efficient Concurrent Zero-Knowledge in the Auxiliary String Model". In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Heidelberg, May 2000, pp. 418–430 (cit. on p. 51).

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-Malleable Cryptography (Extended Abstract)". In: *23rd ACM STOC*. ACM Press, May 1991, pp. 542–552 (cit. on p. 60).

[den90] Bert den Boer. "Diffie-Hellman is as Strong as Discrete Log for Certain Primes (Rump Session)". In: *CRYPTO'88*. Ed. by Shafi Goldwasser. Vol. 403. LNCS. Springer, Heidelberg, Aug. 1990, pp. 530–539 (cit. on p. 20).

[DF02] Ivan Damgård and Eiichiro Fujisaki. "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order". In: *ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. LNCS. Springer, Heidelberg, Dec. 2002, pp. 125–142 (cit. on pp. 54, 100–102, 104, 107, 128).

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. “Non-interactive Zero-Knowledge from Homomorphic Encryption”. In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Heidelberg, Mar. 2006, pp. 41–59 (cit. on pp. 4, 128).

[DH76] Whitfield Diffie and Martin Hellman. “New directions in cryptography”. In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on p. 20).

[DM10] Ivan Damgård and Gert Læssøe Mikkelsen. “Efficient, Robust and Constant-Round Distributed RSA Key Generation”. In: *TCC 2010*. Ed. by Daniele Micciancio. Vol. 5978. LNCS. Springer, Heidelberg, Feb. 2010, pp. 183–200 (cit. on pp. 101, 124).

[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. 56).

[DPR61] Martin Davis, Hilary Putnam, and Julia Robinson. “The decision problem for exponential diophantine equations”. In: *Annals of Mathematics* (1961), pp. 425–436 (cit. on p. 54).

[ECR] ECRYPT II. *eBATS*. http://bench.cr.yp.to/results-dh.html (cit. on p. 61).

[EG14] Alex Escala and Jens Groth. “Fine-Tuning Groth-Sahai Proofs”. In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, Mar. 2014, pp. 630–649. DOI: 10.1007/978-3-642-54631-0_36 (cit. on p. 57).

[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. 22, 70).

[ElG85] Taher ElGamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. In: *IEEE Transactions on Information Theory* 31 (1985), pp. 469–472 (cit. on pp. 17, 29).

[FGM89] Martin Furer, Oded Goldreich, and Yishay Mansour. “On completeness and soundness in interactive proof systems”. In: (1989) (cit. on p. 39).

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. “Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract)”. In: *31st FOCS*. IEEE Computer Society Press, Oct. 1990, pp. 308–317 (cit. on pp. 56, 57, 64).

[FNO15] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. “Privacy-Free Garbled Circuits with Applications to Efficient Zero-Knowledge”. In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 191–219. DOI: 10.1007/978-3-662-46803-6_7 (cit. on p. 53).

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Heidelberg, Aug. 1997, pp. 16–30 (cit. on pp. 25, 100–102, 128).

[For87] Lance Fortnow. "The complexity of perfect zero-knowledge". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 204–209 (cit. on p. 42).

[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 pp. 57, 60).

[Gen04] Rosario Gennaro. "Multi-trapdoor Commitments and Their Applications to Proofs of Knowledge Secure Under Concurrent Man-in-the-Middle Attacks". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 220–236 (cit. on pp. 115, 116).

[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 p. 66).

[GH98] Oded Goldreich and Johan Håstad. "On the complexity of interactive proofs with bounded communication". In: *Information Processing Letters* 67.4 (1998), pp. 205–214 (cit. on p. 44).

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. "Protecting Data Privacy in Private Information Retrieval Schemes". In: *30th ACM STOC*. ACM Press, May 1998, pp. 151–160 (cit. on p. 66).

[GK16] Shafi Goldwasser and Yael Tauman Kalai. "Cryptographic Assumptions: A Position Paper". In: *TCC 2016-A, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 505–522. DOI: 10.1007/978-3-662-49096-9_21 (cit. on p. 25).

[GKZ14] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. "Breaking '128-bit Secure' Supersingular Binary Curves - (Or How to Solve Discrete Logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$)". 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. 62).

[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 pp. 64, 65).

[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. 31, 64, 65).

[GLLM05] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. "On Private Scalar Product Computation for Privacy-Preserving Data Mining". In: *ICISC 04*. Ed. by Choonsik Park and Seongtaek Chee. Vol. 3506. LNCS. Springer, Heidelberg, Dec. 2005, pp. 104–120 (cit. on p. 101).

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption & how to play mental poker keeping secret all partial information". In: *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM. 1982, pp. 365–377 (cit. on p. 3).

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304 (cit. on p. 38).

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208 (cit. on pp. 3, 4, 39).

[GMS10] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. "Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis". In: *FC 2010*. Ed. by Radu Sion. Vol. 6052. LNCS. Springer, Heidelberg, Jan. 2010, pp. 375–382 (cit. on pp. 101, 124).

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)". In: *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 174–187 (cit. on p. 41).

[GMW87a] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 218–229 (cit. on pp. 5, 6, 60, 65).

[GMW87b] 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. 5, 6, 60, 65, 85).

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. "Strengthening Zero-Knowledge Protocols Using Signatures". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Springer, Heidelberg, May 2003, pp. 177–194 (cit. on p. 93).

[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, 46).

[GO94] Oded Goldreich and Yair Oren. "Definitions and Properties of Zero-Knowledge Proof Systems". In: *Journal of Cryptology* 7.1 (1994), pp. 1–32 (cit. on p. 50).

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. New York, NY, USA: Cambridge University Press, 2004. ISBN: 0521830842 (cit. on pp. 46, 85).

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. New York, NY, USA: Cambridge University Press, 2006. ISBN: 0521035368 (cit. on pp. 2, 42).

[Gol98] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*. Vol. 17. Springer Science & Business Media, 1998 (cit. on p. 37).

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 97–111 (cit. on p. 57).

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 339–358 (cit. on p. 57).

[Gro04] Jens Groth. *Honest verifier zero-knowledge arguments applied*. BRICS, 2004 (cit. on pp. 42, 46).

[Gro05] Jens Groth. "Non-interactive Zero-Knowledge Arguments for Voting". In: *ACNS 05*. Ed. by John Ioannidis, Angelos Keromytis, and Moti Yung. Vol. 3531. LNCS. Springer, Heidelberg, June 2005, pp. 467–482 (cit. on pp. 101, 122, 124).

[Gro09] Jens Groth. "Linear Algebra with Sub-linear Zero-Knowledge Arguments". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 192–208 (cit. on pp. 53, 122).

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 431–448 (cit. on pp. 53, 101, 124).

[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. 4, 7, 57, 60, 62, 65, 68, 72, 73, 87, 88).

[GS86] Shafi Goldwasser and Michael Sipser. "Private Coins versus Public Coins in Interactive Proof Systems". In: *18th ACM STOC*. ACM Press, May 1986, pp. 59–68 (cit. on p. 38).

[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. 42).

[GSV99] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. "Can Statistical Zero Knowledge Be Made Non-interactive? or On the Relationship of SZK and NISZK". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 467–484 (cit. on p. 56).

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. "Groth-Sahai Proofs Revisited". In: *PKC 2010*. Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. LNCS. Springer, Heidelberg, May 2010, pp. 177–192 (cit. on p. 57).

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "On interactive proofs with a laconic prover". In: *Computational Complexity* 11.1 (2002), pp. 1–53 (cit. on p. 44).

[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. 28).

[HJ12] Dennis Hofheinz and Tibor Jager. "Tightly Secure Signatures and Public-Key Encryption". In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 590–607 (cit. on p. 91).

[HJK11] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. "Short Signatures from Weaker Assumptions". In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 647–666 (cit. on p. 102).

[HK09] Dennis Hofheinz and Eike Kiltz. "The Group of Signed Quadratic Residues and Applications". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 637–653 (cit. on p. 129).

[HK12] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications". In: *Journal of Cryptology* 25.3 (July 2012), pp. 484–527. DOI: 10.1007/s00145-011-9102-5 (cit. on p. 76).

[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. 65).

[HMRT12] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. "Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting". In: *CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. LNCS. Springer, Heidelberg, Feb. 2012, pp. 313–331 (cit. on p. 101).

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. "NTRU: A ring-based public key cryptosystem". In: *International Algorithmic Number Theory Symposium*. Springer. 1998, pp. 267–288 (cit. on p. 17).

[HR07] Iftach Haitner and Omer Reingold. "Statistically-hiding commitment from any one-way function". In: *39th ACM STOC*. Ed. by David S. Johnson and Uriel Feige. ACM Press, June 2007, pp. 1–10 (cit. on pp. 28, 44).

[HW09] Susan Hohenberger and Brent Waters. "Short and Stateless Signatures from the RSA Assumption". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 654–670 (cit. on pp. 25, 102).

[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. 65).

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. "Efficient arguments without short PCPs". In: *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*. IEEE. 2007, pp. 278–291 (cit. on p. 44).

[IY88] Russell Impagliazzo and Moti Yung. "Direct Minimum-Knowledge Computations". In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Heidelberg, Aug. 1988, pp. 40–51 (cit. on p. 42).

[Jar14] Stanislaw Jarecki. "Practical Covert Authentication". In: *PKC 2014*. Ed. by Hugo Krawczyk. Vol. 8383. LNCS. Springer, Heidelberg, Mar. 2014, pp. 611–629. DOI: 10.1007/978-3-642-54631-0_35 (cit. on p. 66).

[Jar16a] Stanis law Jarecki. "Efficient covert two-party computation". In: (2016) (cit. on p. 61).

[Jar16b] Stanislaw Jarecki. *Efficient Covert Two-Party Computation*. Cryptology ePrint Archive, Report 2016/1032. http://eprint.iacr.org/2016/1032. 2016 (cit. on p. 66).

[JG02] Ari Juels and Jorge Guajardo. "RSA Key Generation with Verifiable Randomness". In: *PKC 2002*. Ed. by David Naccache and Pascal Paillier. Vol. 2274. LNCS. Springer, Heidelberg, Feb. 2002, pp. 357–374 (cit. on pp. 101, 124).

[JKK14] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. "Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model". In: *ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, Heidelberg, Dec. 2014, pp. 233–253. DOI: 10.1007/978-3-662-45608-8_13 (cit. on p. 101).

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. "Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently". In: *ACM CCS 13*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 955–966 (cit. on p. 53).

[JOP14] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. "The past, evolving present, and future of the discrete logarithm". In: *Open Problems in Mathematics and Computational Science*. Springer, 2014, pp. 5–36 (cit. on p. 19).

[Jou00] Antoine Joux. "A one round protocol for tripartite Diffie–Hellman". In: *International algorithmic number theory symposium*. Springer. 2000, pp. 385–393 (cit. on p. 22).

[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 p. 19).

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. "Efficient Two-Party Secure Computation on Committed Inputs". In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Heidelberg, May 2007, pp. 97–114 (cit. on p. 101).

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman J. J. te Riele, Andrey Timofeev, and Paul Zimmermann. "Factorization of a 768-Bit RSA Modulus". In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Heidelberg, Aug. 2010, pp. 333–350 (cit. on p. 23).

[Kd22] Maurice Kraitchik and Maurice d'Ocagne. *Théorie des nombres*. Vol. 1. Gauthier-Villars Paris, 1922 (cit. on p. 19).

[Kil92] Joe Kilian. “A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)”. In: *24th ACM STOC*. ACM Press, May 1992, pp. 723–732 (cit. on p. 44).

[KLC12] Myungsun Kim, Hyung Tae Lee, and Jung Hee Cheon. “Mutual Private Set Intersection with Linear Complexity”. In: *WISA 11*. Ed. by Souhwan Jung and Moti Yung. Vol. 7115. LNCS. Springer, Heidelberg, Aug. 2012, pp. 219–231 (cit. on p. 124).

[KLL+15] Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa, Kateryna Pavlyk, and Qiang Tang. “Optimal rate private information retrieval from homomorphic encryption”. In: *Proceedings on Privacy Enhancing Technologies* 2015.2 (2015), pp. 222–243 (cit. on p. 101).

[Kob87] Neal Koblitz. “Elliptic curve cryptosystems”. In: *Mathematics of computation* 48.177 (1987), pp. 203–209 (cit. on p. 19).

[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 p. 31).

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. “Traceable Signatures”. In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 571–589 (cit. on pp. 101, 111).

[LAN01] Helger Lipmaa, N. Asokan, and Valtteri Niemi. *Secure Vickrey Auctions without Threshold Trust*. Cryptology ePrint Archive, Report 2001/095. http://eprint.iacr.org/2001/095. 2001 (cit. on p. 101).

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. “Algebraic methods for interactive proof systems”. In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868 (cit. on p. 38).

[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. 65).

[Lip03] Helger Lipmaa. “On Diophantine Complexity and Statistical Zero-Knowledge Arguments”. In: *ASIACRYPT 2003*. Ed. by Chi-Sung Laih. Vol. 2894. LNCS. Springer, Heidelberg, Nov. 2003, pp. 398–415. DOI: 10.1007/978-3-540-40061-5_26 (cit. on pp. 5, 54, 100–102, 109, 111–113, 116, 119, 122).

[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. 65).

[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. 65).

[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 p. 77).

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 1–23 (cit. on p. 17).

[Mau09] Ueli M. Maurer. "Unifying Zero-Knowledge Proofs of Knowledge". In: *AFRICACRYPT 09*. Ed. by Bart Preneel. Vol. 5580. LNCS. Springer, Heidelberg, June 2009, pp. 272–286 (cit. on pp. 92, 93).

[Mau94] Ueli M. Maurer. "Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Algorithms". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Heidelberg, Aug. 1994, pp. 271–281 (cit. on p. 20).

[McE78] Robert J McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116 (cit. on p. 17).

[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. 19).

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. "Signatures Resilient to Continual Leakage on Memory and Computation". In: *TCC 2011*. Ed. by Yuval Ishai. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011, pp. 89–106 (cit. on p. 64).

[Nao91] Moni Naor. "Bit Commitment Using Pseudorandomness". In: *Journal of Cryptology* 4.2 (1991), pp. 151–158 (cit. on p. 28).

[NOV06] Minh-Huyen Nguyen, Shien Jin Ong, and Salil P. Vadhan. "Statistical Zero-Knowledge Arguments for NP from Any One-Way Function". In: *47th FOCS*. IEEE Computer Society Press, Oct. 2006, pp. 3–14 (cit. on pp. 28, 44).

[NY90] Moni Naor and Moti Yung. "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks". In: *22nd ACM STOC*. ACM Press, May 1990, pp. 427–437 (cit. on p. 60).

[Oka96] Tatsuaki Okamoto. "On Relationships between Statistical Zero-Knowledge Proofs". In: *28th ACM STOC*. ACM Press, May 1996, pp. 649–658 (cit. on p. 42).

[Ore87] Yair Oren. "On the Cunning Power of Cheating Verifiers: Some Observations about Zero Knowledge Proofs (Extended Abstract)". In: *28th FOCS*. IEEE Computer Society Press, Oct. 1987, pp. 462–471 (cit. on p. 56).

[OV07] Shien Jin Ong and Salil P. Vadhan. "Zero Knowledge and Soundness Are Symmetric". In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Heidelberg, May 2007, pp. 187–209 (cit. on p. 43).

[OW93] Rafail Ostrovsky and Avi Wigderson. "One-way functions are essential for non-trivial zero-knowledge". In: *Theory and Computing Systems, 1993., Proceedings of the 2nd Israel Symposium on the*. IEEE. 1993, pp. 3–17 (cit. on p. 43).

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Heidelberg, Aug. 1992, pp. 129–140 (cit. on p. 28).

[Pol03] Chris Pollett. "On the bounded version of Hilbert's tenth problem". In: *Arch. Math. Log.* 42.5 (2003), pp. 469–488. DOI: 10.1007/s00153-002-0162-y. URL: http://dx.doi.org/10.1007/s00153-002-0162-y (cit. on p. 54).

[Pol78] John M Pollard. "Monte Carlo methods for index computation (mod p)". In: *Mathematics of computation* 32.143 (1978), pp. 918–924 (cit. on p. 19).

[PS00] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and Blind Signatures". In: *Journal of Cryptology* 13.3 (2000), pp. 361–396 (cit. on pp. 101, 106).

[PS96] David Pointcheval and Jacques Stern. "Security Proofs for Signature Schemes". In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 387–398 (cit. on pp. 101, 106).

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. "A Framework for Efficient and Composable Oblivious Transfer". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Springer, Heidelberg, Aug. 2008, pp. 554–571 (cit. on p. 128).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93 (cit. on p. 17).

[RS86] Michael O. Rabin and Jeffery O. Shallit. "Randomized algorithms in number theory". In: *Communications on Pure and Applied Mathematics* 39.S1 (1986), S239–S256. ISSN: 1097-0312. DOI: 10.1002/cpa.3160390713. URL: http://dx.doi.org/10.1002/cpa.3160390713 (cit. on pp. 54, 111).

[RSA78a] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126 (cit. on p. 17).

[RSA78b] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signature and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126 (cit. on pp. 24, 100).

[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 pp. 47, 61, 65).

[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. 19).

[Sha92] Adi Shamir. "Ip= pspace". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 869–877 (cit. on p. 38).

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Heidelberg, May 1997, pp. 256–266 (cit. on p. 19).

[Sho99] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332 (cit. on pp. 17, 18, 23).

[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. 65).

[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. 65).

[Tes01] Edlyn Teske. "On random walks for Pollard′s rho method". In: *Mathematics of computation* 70.234 (2001), pp. 809–825 (cit. on p. 19).

[Vad04] Salil P. Vadhan. "An Unconditional Study of Computational Zero Knowledge". In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 176–185 (cit. on p. 43).

[Vil17] Jorge L Villar. "Equivalences and Black-Box Separations of Matrix Diffie-Hellman Problems". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2017, pp. 435–464 (cit. on p. 22).

[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 pp. 64, 75, 77).

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)". In: *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167 (cit. on pp. 5, 6, 60).

[YSLM08] Tsz Hon Yuen, Willy Susilo, Joseph K. Liu, and Yi Mu. "Sanitizable Signatures Revisited". In: *CANS 08*. Ed. by Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong. Vol. 5339. LNCS. Springer, Heidelberg, Dec. 2008, pp. 80–97 (cit. on p. 101).

Résumé

Dans cette thèse, nous étudions les preuves à divulgation nulle de connaissance, une primitive cryptographique permettant de prouver une assertion en ne révélant rien de plus que sa véracité, et leurs applications au calcul sécurisé. Nous introduisons tout d'abord un nouveau type de preuves à divulgation nulle, appelées arguments implicites à divulgation nulle, intermédiaire entre deux notions existantes, les preuves interactives et les preuves non-interactives à divulgation nulle. Cette nouvelle notion permet d'obtenir les mêmes bénéfices en terme d'efficacité que les preuves non-interactives dans le contexte de la construction de protocoles de calcul sécurisé faiblement interactifs, mais peut être instanciée à partir des mêmes hypothèses cryptographiques que les preuves interactives, permettant d'obtenir de meilleures garanties d'efficacité et de sécurité. Dans un second temps, nous revisitons un système de preuves à divulgation nulle de connaissance qui est particulièrement utile dans le cadre de protocoles de calcul sécurisé manipulant des nombres entiers, et nous démontrons que son analyse de sécurité classique peut être améliorée pour faire reposer ce système de preuve sur une hypothèse plus standard et mieux connue. Enfin, nous introduisons une nouvelle méthode de construction de systèmes de preuves à divulgation nulle sur les entiers, qui représente une amélioration par rapport aux méthodes existantes, tout particulièrement dans un modèle de type client-serveur, où un client à faible puissance de calcul participe à un protocole de calcul sécurisé avec un serveur à forte puissance de calcul.

Mots Clés

cryptographie, sécurité prouvée, hypothèses calculatoire, preuves à divulgation nulle de connaissance, calcul sécurisé.

Abstract

In this thesis, we study zero-knowledge proofs, a cryptographic primitive that allows to prove a statement while yielding nothing beyond its truth, and their applications to secure computation. Specifically, we first introduce a new type of zero-knowledge proofs, called implicit zero-knowledge arguments, that stands between two existing notions, interactive zero-knowledge proofs and non-interactive zero-knowledge proofs. Our new notion provides the same efficiency benefits than the latter when used to design round-efficient secure computation protocols, but it can be built from essentially the same cryptographic assumptions than the former, which allows to get improved efficiency and security guarantees. Second, we revisit a zero-knowledge proof system that is particularly useful for secure computation protocols manipulating integers, and show that the known security analysis can be improved to base the proof system on a more well-studied assumption. Eventually, we introduce a new method to build zero-knowledge proof systems over the integers, which particularly improves over existing methods in a client-server model, where a weak client executes a secure computation protocol with a powerful server.

Keywords

cryptography, provable security, computational assumptions, zero-knowledge proofs, secure computation.