A tale of SNARKs: quantum resilience, knowledge extractability and data privacy

Anca Nitulescu

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

A tale of SNARKs: quantum resilience, knowledge extractability and data privacy

Soutenue par
Anca NITULESCU
Le 01 avril 2019

Ecole doctorale n° 389
Sciences Mathématiques de Paris Centre

Spécialité
Mathématiques

ENS
ÉCOLE NORMALE SUPÉRIEURE

Composition du jury :

Benoît LIBERT École normale supérieure de Lyon	*Président*
Markulf KOHLWEISS University of Edinburgh	*Rapporteur*
Helger LIPMAA University of Tartu	*Rapporteur*
Rosario GENNARO City University of New York	*Examinateur*
Carla RÀFOLS University Pompeu Fabra	*Examinateur*
David POINTCHEVAL École normale supérieure	*Directeur de thèse*
Michel ABDALLA École normale supérieure	Co*Directeur de thèse*
Dario FIORE IMDEA Software Institute	Co*Directeur de thèse*

Un recueil de SNARKs: sécurité quantique, extractabilité et confidentialité des données

Anca Nitulescu

Dirigée par: Michel Abdalla, Dario Fiore, David Pointcheval

Résumé

Cette thèse est consacrée à une exploration des schémas de preuves succinctes de connaissance, les SNARKs. S'inscrivant dans un contexte marqué par le développement du Cloud et des technologies Blockchain, les SNARKs sont des primitives cryptographiques permettant la vérification de l'intégrité des calculs. Dans un modèle de type client-serveur, où un client à faible puissance de calcul délègue une tâche à un serveur à forte puissance de calcul, les SNARKs permettent au client de vérifier efficacement si le serveur a bien exécuté la tâche demandée.

Les principaux résultats acquis au cours de cette thèse suivent trois directions: la construction des preuves SNARKs résistant aux attaques cryptographiques post-quantiques, l'analyse de la notion d'extractabilité des protocoles SNARKs avec des applications; et la conception d'un schéma qui garantit non seulement l'intégrité des calculs, mais aussi la confidentialité des données en entrée du calcul.

D'abord, nous remarquons qu'actuellement, la plupart des SNARKs sont basées sur des hypothèses qui sont vulnérables face à une avancée technologique, comme la construction d'un ordinateur quantique. Nous proposons la construction des SNARKs avec des meilleures garanties de sécurité. Notre schéma est plus général et peut être implémenté avec des chiffrements à base de réseaux qui sont conjecturés sécurisés même en présence d'ordinateurs quantiques.

Nous nous intéressons aussi à des aspects plus théoriques des protocoles SNARKs. Grâce à une analyse des protocoles plus complexes utilisant les preuves SNARKs, on pose les bases d'une étude de la propriété d'extractabilité et la composabilité des SNARKs avec d'autres primitives cryptographiques.

Nous considérons le cas d'usage des signatures numériques homomorphes, et nous clarifions les limites et les propriétés des SNARKs dans ce contexte. Nous proposons une nouvelle définition, celle d'O-SNARK, qui prend en compte l'extractabilité dans le cas étendu où le serveur a accès à un oracle (qui peut par exemple générer des signatures valides). Malheureusement, nous montrons un résultat d'impossibilité pour certains schémas de signature. En revanche, nous offrons quelques solutions adaptées aux particularités du problème de conception des O-SNARKs.

Additionnellement, nous considérons le problème de confidentialité des données dans les calculs délégués. Nous nous intéressons à élaborer un protocole SNARK qui peut être utilisé pour prouver des calculs effectués sur des données chiffrées.

Cette nouvelle notion doit permettre d'obtenir des bénéfices similaires en terme d'efficacité que les preuves SNARKs classiques tout en restant compatible avec les schémas laborieux de chiffrement homomorphe.

Ensemble, ces résultats posent les bases d'une étude détaillée des preuves SNARKs et participent à la construction de nouveaux protocoles plus sécurisés et plus riches en fonctionnalités.

Abstract

The contributions detailed in this thesis focus on the design and the analysis of Succinct non-interactive arguments of knowledge, known as SNARKs. SNARKs enable a party with large computational resources to prove to a weaker party that a particular statement is true in an efficient way without further interaction and under a minimal communication requirement. SNARKs play a significant role both in the theory of proof systems and in practice as essential building blocks for many cryptographic protocols.

We will present along this manuscript several results that deal with three different aspects of SNARK protocols.

First, we remark the lack of SNARK constructions that take into account the advent of quantum cryptoanalytic efforts. To address this problem, we propose a new framework that optimizes some previous works and further allows the instantiation of a *quantum-resilient* SNARK scheme from lattice assumptions.

We are also interested in the theoretical aspects of SNARK schemes. We study the strong notion of soundness that made SNARKs very popular for a set of applications as delegating computation, homomorphic authentification protocols, electronic cryptocurrencies or anonymous credentials.

The knowledge soundness property of a SNARK can be formally stated in terms of some *knowledge extractability*: whenever the prover returns an accepting proof, there exists an extractor that is able to output a valid witness for the statement. We remark some limitations of this definition in a scenario where adversarial provers are given access to an oracle. To address this, we define a new notion, O-SNARKs, and we study the feasibility of such knowledge extraction for this new notion. We first give a negative result: a counterexample consisting of a "malign" signing oracle for which SNARKs are insecure. On the positive side, we present some candidates for O-SNARKs under specific restrictions.

We further recognize the need for using O-SNARKs in some well-known homomorphic authentication schemes with underlying succinct proofs: homomorphic signatures, functional signatures and aggregate signatures.

Last but not least, we address the problem of *data privacy* in delegated computation, that takes into consideration along with the integrity of the result, the confidentiality of the computation's inputs. To achieve this, we study the possibility of constructing SNARKs that enables verification of computations over encrypted data. The departing point is a generic solution that combines fully homomorphic encryption schemes with SNARKs. We improve the efficiency of this approach, by constructing a tailored SNARK designed to exploit the structure of the ciphertexts for a specific encryption scheme. Using this dedicated SNARK as a building block, we obtain a verifiable computation scheme over encrypted data, at a prover's costs that has a minimal dependence on the computation.

Together, these results strengthen the foundations of SNARK systems, give a deep understanding of their underlying security notions, and pave the way towards more secure and more practical SNARK schemes.

Acknowledgments

First of all, I would like to thank you, the reader, for your interest in this PhD thesis and hope you will learn something useful along these pages.

For this section, I would take the freedom to switch from one language to another in order to best express my gratitude to all the people I had the chance to meet during my Phd years and that contributed to realizing this great experience.

I would like to thank my supervisors David Pointcheval, Dario Fiore and Michel Abdalla for their direction, their availability and for the countless hours spent with me on different research problems during these years.

I would like to express my gratitude to my two reviewers Markulf Kohlzeiss and Helger Lipmaa who read and commented very carefully my thesis. I know this is a long work and I have to apologize for making it even harder with all my typos and omissions. I am very grateful for the feedback and the valuable suggestions they made, I learned a lot from their comments.

I am also thankful to each of my other committee members, Rosario Gennaro, Benoît Libert, Carla Ràfols, for accepting to evaluate my work and for their time and interest.

Ces ans m'ont donné l'occasion de nourrir ma curiosité et mon savoir dans le domaine de la cryptographie, j'ai pu explorer des nombreux défis et j'ai beaucoup partagé et appris. La richesse de cette aventure a pour protagonistes la vitalité, la passion et la curiosité des différentes personnes qui ont participé de façon active à la réalisation de ce projet de thèse. Je ne pourrai pas les citer toutes mais j'en suis très reconnaissante.

Je remercie David de m'avoir toujours guidé, non seulement dans ma recherche, mais aussi avec les autres aspects de la vie de thésard. Je lui suis reconnaissante pour m'avoir montré tant de bienveillance et de compréhension, étant donné que je ne suis pas le doctorant modèle, avec mes horaires improbables et ma façon chaotique de travailler. Je me rends compte de la chance que j'ai eu d'être sous sa direction pendant ces années et j'apprecie beaucoup tout son soutien.

I would like to thank Dario Fiore, who is somehow responsible of my decision of starting a PhD on the topic of SNARKs. I am glad I got to work with him and will always remember as a wonderful time the enriching collaboration that started as an intern at IMDEA and continued also during my PhD. Most of all, I am grateful for the passion and energy he inspired to me, for having the patience to advice me while I was still learning the most basic concepts of cryptography.

I thank Michel Abdalla for being my "social" co-advisor, for his cheerful, positive attitude, and for always looking after the well-being and the interests of our team.

Next, I would like to thank my other mentors, who supported me intellectually and morally during my visits abroad: Mariana Raykova, Rosario Gennaro, Tal Malkin. I want to acknowledge all my other co-authors: Mario Cornejo, Michele Orrù, Michele Minelli.

I am also happy I had the occasion to interact with all permanent members of the team, Georg Fuchsbauer, Hoeteck Wee, and thank them for the chats and for entrusting me with subreviews of several papers.

En ce qui concerne la vie au labo, je suis nostalgique en pensant aux nombreux moments passés avec les autres doctorants, post-docs et jeunes chercheurs de l'équipe crypto. Je remercie pour les pauses café à tous ceux dont je ne me rappelle probablement pas les prénoms. Je fais un effort de mémoire: Thomas Peters, Pooya Farshim, Céline Chevalier, Houda Ferradi, Florian Bourse, Geoffroy Couteau, Rafaël Del Pino, Răzvan Roşie, Aurélien Dupin, Michele Minelli, Jérémy Chotard.

Je remercie particulièrement Romain Gay d'avoir été un si bon collegue de bureau, toujours ouvert aux distractions, Balthazar Bauer pour ses recommendations cinéma si nichés, Michele Orrù pour les longues soirées passées au labo et à la K-Fêt, Dahmun Goudarzi pour son grand sourire et son bon-humeur, Chloé Hébant, Louiza Khati et Mélissa Rossi pour le girl-power, Adrian Thillard pour darder sans cesse, Pierrick Méaux pour ses crevettes et ses efforts d'animer le labo, Aisling Connolly, pour les discussions si interessantes et le soutient pendant ma redaction, Julia Hesse car "Ich bin nicht einverstanden!".

Je tiens également à exprimer toute ma reconnaissance à l'équipe administrative du DI, notamment à Joëlle Isnard, Lise-Marie Bivard, Valérie Mongiat et Sophie Jaudon.

Thank you all my crypto-friends from US for the road trips, the city breaks, the stories and laughts we shared: Sophia Yakoubov, Ghada Almashaqbeh, Brent Carmer, Saleet Klein.

I reserve a special thank to my Paparouna, Marios Georgiou, for his kind friendship, his encouragements, his support, his ability to always raise my moral and for the great crypto exchanges we had these years and the fun time we spent togheter in many occasions.

Je souhaite également profiter pour remercier mes amis non-crypto, qui ont contribué plus qu'ils le croient au bon déroulement de ma thèse pendant ces années.

Je tiens à remercier mon ami Antonia et Chewby, Mandu et Mario et petite Éloise qui a été longtemps sujet de conversation. Je leurs remercie pour tous les moments partagés ensemble à Paris et ailleurs. Merci à Antonia de si bien me comprendre et de m'avoir toujours soutenue et encouragée. Aussi d'avoir su monter mon moral dans les moments difficiles par son optimisme et sa confiance en moi.

Merci Raphaël de m'inspirer le soif de savoir – je ne parle pas de nos échanges pendant les cours des graphes au MPRI, mais plutôt de nos soirées autour d'une bonne bière belge avec des longues discussions culturels et politiques passionnantes. Merci Alice pour ta joie de vivre, tes magnifiques histoires de voyage. Merci également pour des histoires autour d'un verre à Flavien, Renata, Alex, Bruno, Élodie.

Merci Anthony, mon maître cuisiner et mon disciple dans la danse, toujours un bon confident pour des problèmes de cœur. Merci à mon partenaire de danse, mon ami des fringues, complice dans la coquetterie, Laurent.

Merci aux personnes qui ont été plus qu'amis et qui ont occupées une place spéciale pour moi: le fou d'amour et de drame Pierre-Hélie, mon cher cactus Jesse, "el piloto automático" Vanesa, le moustachoux Alexandre.

Un grand merci à mes amis de la Cité U: Marta, Mehdi, Pamela, Angie, Paula, Alma, Marion, Céleste, Camilo, Flore. Et merci aux metteurs en scène, danseurs et gens du théâtre avec qui j'ai eu la chance de participer à des projets chouettes. Merci d'avoir partagé leur talent et leur inspiration à: David Garel pour les soirées d'improvisation de lundi, Guillaume Barbot pour la fête de l'amour, Jessica Dalle pour son super pouvoir de raconter des histoires, créer des personnages, nous plonger dans la fiction, François Lanel pour le ludique et l'inédit, Vincent Delétang pour faire de nos corps des symphonies en mouvement.

Je remercie mes colocs Guilhem et Lambert pour avoir partagé des agréables moments ensemble, les soirées jeux de plateau, les discussions interdisciplinaires autour d'une tisane. Et merci à François pour la synchronisation parfaite pour la chambre, aussi pour les plantes et les BDs.

Merci à tous mes étudiants qui ont été une source d'énergie, de vitalité et d'inspiration pour moi. J'ai dédié une bonne partie de mon temps de recherche à préparer les cours et les TDs, mais j'ai été toujours récompensée par la passion et l'appréciation que certaines m'ont montré.

Je souhaite aussi rappeler l'endroit qui a fait germer cette graine de curiosité scientifique qui s'est développée plus tard en une passion pour la cryptographie et pour la recherche académique. Je remercie mes professeurs de l'Université de Bucarest, un endroit très cher à mon cœur, où j'ai appris non seulement l'algèbre, la théorie de nombres et la géométrie, mais surtout des valeurs morales, de la passion pour l'enseignement, le respect, la rigueur. Merci à Voica Cristian, Cătălin Gherghe, Șerban Strătilă, Ornea Liviu, Victor Vuletescu, Marius Vlădoiu, Constantin Năstăsescu.

Et une section pour mes amis inanimés: Je remercie tous ceux qui se sont occupés de mon ami vert César pendant mon absence et je remercie César d'avoir résisté aux intempéries. Je remercie aux crevettes anonymes qui m'ont tenu compagnie dans leurs aquariums à eau trouble. Je remercie mon vélo Rose, qui m'a amené chaque jour au bureau malgré les conditions météorologiques.

Nu în ultimul rând, îmi doresc să mulțumesc familiei mele, cuplului Floreta, lui Fabian, Moșicăi, verișorilor vechi și noi, Diana, David și lui Florian.

Doresc să le mulțumesc special părinților mei fantastici care mi-au fost neîncetat aproape, m-au susținut necondiționat și care cu siguranță știu că în ciuda absențelor mele lungi, eu îi am mereu alături de mine în gând.

Writing the acknowledgements section was for me like riding a rollercoaster of feelings. Remembering all this long adventure of my PhD, lovely times, sometimes tough, sometimes challenging, sometimes happy and so many different faces and places! I sit here, in front of this manuscript, filled with nostalgia, but with a smile on my face, embracing this moment of reminiscence...

Contents

Chapter 1

Introduction

Context. The modern digital world that we live in today is the result of many innovations and technology advances. Some of the most remarkable features we benefit from nowadays are the automated systems for storing, processing, and distributing information. The designated term used for these systems is Information Technology (IT) and refers mainly to computers and communication networks. Since the widespread use of IT facilities, it is evident that the standards of life and education have improved, as well as many other things that include our needs and wants. Connectivity increases access to information, ease of sharing and fast processing of data, but it also increases the possibility of undesired behaviour. Unfortunately, the negative impact of these new technologies are often overlooked.

A fascinating article in philosophy [vdHBPW18] addresses the relationship between the new technologies and the security concerns they raise. The authors emphasise that recent advances in information technology have reduced the amount of control over personal data and open up the possibility of a range of negative consequences as a result of unreliable infrastructures.

How can we avoid the new technologies to be a threat to our security?

1.1 The Role of Cryptography

Human beings have always valued their privacy and security, the protection of their personal sphere of life. Even if unconsciously people expose themselves to many risks in their online activities, they will always claim their right to privacy.

Alan Westin believes that new technologies alter the balance between privacy and disclosure, he defines privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [Wes67].

We certainly do not want our personal information to be accessible to just anyone at any time, our outsourced data to be altered without us noticing, or our identities to be stolen over the Internet. Meanwhile, it would be preferable to be able to enjoy the benefits of using the latest technologies.

Cryptography addresses these and more concerns by designing systems that help us keep our information secret to unintended receivers, prove our identity to digital interlocutors,

and allowing secure communication in the presence of untrusted parties.

By using Cryptography, we can dream of regaining the comfort and privacy we seem to lose as an information intensive society.

1.2 History of Cryptography

Cryptography has a long and fascinating history that begins together with the invention of writing. In its early years, the focus was mainly on protecting the confidentiality of information and ensuring secrecy in communications, such as those of spies, military leaders, lovers. In recent decades, the field has expanded beyond confidentiality concerns to include techniques for message integrity checking, sender/receiver identity authentication, digital signatures, interactive and non-interactive proofs of knowledge and secure computation, among others.

Early Ages. The earliest forms of secret writing required little more than local pen and paper analogues, as most people could not read. Simple versions of ciphers have never offered much confidentiality from enterprising opponents. An early substitution encryption method was the Caesar cipher, in which each letter in the plaintext was replaced by another letter some fixed number of positions further down the alphabet. It was named after Julius Caesar who is reported to have used it, with a shift of 3, to communicate with his generals during his military campaigns.

These methods of secure communication and authentication were only guaranteed by ad-hoc constructions, and most of the time by keeping secret the design of the systems. Later, the industrial revolution and the apparitions of machines changed both the possible functionalities and the theoretical treatment of cryptosystems.

Military Cryptography. In 1883 Auguste Kerckhoffs wrote two journal articles on La Cryptographie Militaire, [Ker83] in which he stated six design principles for military ciphers. Translated from French, they are:

1. The system must be practical, if not mathematically indecipherable;
2. It should not require secrecy, and it should not be a problem if it falls into enemy hands;
3. It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will;
4. It must be applicable to telegraph communications;
5. It must be portable, and should not require several persons to handle or operate;
6. Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

Some are no longer relevant given the ability of computers to perform complex encryption, but his second axiom, now known as Kerckhoffs's principle, is still critically important.

The development of digital computers and electronics after WWII made possible much more complex ciphers. Furthermore, computers allowed for the encryption of any kind of data representable in any binary format, unlike classical ciphers which only encrypted written language texts; this was new and significant.

The American mathematician Claude Shannon reformulated Kerckhoffs's principle in 1949

[Sha49] as *"the enemy knows the system"*, i.e., "one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them". In that form, it is called Shannon's maxim. In contrast to *security through obscurity*, it is widely embraced by cryptographers.

Modern Cryptography. Extensive open academic research into cryptography is relatively recent; it began only in the mid-1970s. Modern cryptology is a research field that lives in the intersection of mathematics, computer science and more recently, quantum physics according to [HPS08]. The security of most nowadays cryptosystems is related to hard mathematical problems and to some assumptions we make on the limitations of the machines designed to break them. For those cryptographic techniques, there are no absolute proofs of security, as one can always consider the *brute-force attack*, a method consisting in trying all possible solutions to a problem until the correct one is found. At best, there are proofs that some techniques are secure if some problem is difficult to solve, given a specific infrastructure.

The modern cryptosystems were developed and formalized beginning with the work of Feistel [Fei73] at IBM in the early 1970s and culminating in 1977 with the adoption as a U.S. Federal Information Processing Standard for encrypting unclassified information, the *Data Encryption Standard* (DES). Nowadays, DES remains the standard cryptosystem for securing electronic commerce for many financial institutions around the world.

This kind of encryption technique is called *symmetric-key* cryptosystems or *secret-key* cryptosystems. They use the same key to encrypt a message and later on to decrypt the ciphertext. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them is a considerable practical obstacle for users.

The most striking development in the history of cryptography that changed the way encryption was perceived came in 1976 when Diffie and Hellman published *"New Directions in Cryptography"* [DH76]. This work introduced the revolutionary concept of *Public-key cryptography* and also provided a new and ingenious method for key exchange, which solves the key management problem mentioned before. The security of this key exchange protocol is based on the difficulty of solving the discrete logarithm problem (with the existent means of computation).

Although the authors had no practical realization of a public-key encryption scheme at the time, the idea was clear, and it generated extensive interest and activity in the cryptographic community.

Public-key Cryptography. The historian David Kahn, author of the famous book "The Codebreakers" [Kah67], described public-key cryptography as "the most revolutionary new concept in the field since polyalphabetic substitution emerged in the Renaissance".

In public-key cryptography, each party has a pair of keys: a public one and a private (or secret) one. The public one can be published, e.g., on the Internet, and allows anyone to encrypt a message, that can only be decrypted with the corresponding private key. In order to explain this concept, a simple analogy is often used: the public key corresponds to an open lock, whereas the private key corresponds to the lock's key. Publishing the public key is equivalent to making the open lock available; then anyone can write a message, put it in a box, and close the box with the provided lock. The sealed box is then sent to the recipient, who can open it with the appropriate key.

In 1978 Rivest, Shamir, and Adleman discovered the first practical public-key encryption and signature scheme, now referred to as RSA [RSA78].

The RSA scheme is based on another hard mathematical problem, the intractability of factoring large integers. This application of a hard mathematical problem to cryptography revitalized efforts to find more efficient methods to factor. The 1980s saw significant advances in this area but none which rendered the RSA system insecure. Another class of powerful and practical public-key schemes was found by ElGamal in 1984 [ElG84]. These are also based on some problem assumed intractable, the discrete logarithm problem.

One of the most significant contributions provided by public-key cryptography is the *digital signature*; this is the digital analogous of a handwritten signature or stamped seal, and it is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures can provide the assurances of origin, identity and status of an electronic document, transaction or message.

In 1991, the first international standard for digital signatures (ISO/IEC 9796) was adopted. It is based on the RSA public-key scheme. In 1994 the U.S. Government adopted the Digital Signature Standard, a mechanism based on the ElGamal public-key scheme. The search for new public-key schemes, improvements to existing cryptographic mechanisms, and proofs of security continue at a rapid pace.

Nowadays, security products are being developed to address the security needs of an information intensive society. Cryptography has become a widely used tool in communications, computer networks, and in computer security generally.

Future of Cryptography. As well as being aware of the fascinating history of the field, the cryptographic community must also sensibly consider hypothetical future developments while working on their system designs. For instance, continuous improvements in computer processing power have increased the efficacy of brute-force attacks; thus the required key sizes are similarly advancing. Some cryptographic system designers are already considering potential effects of quantum computing. The announced imminence of implementations of these quantum machines should justify the need for this preemptive caution.

1.2.1 The Main Characters

To describe protocols functionalities, the roles of different parties and the type of interactions between them, the use of some fictional characters and scenarios is necessary. Alice and Bob are the world's most famous cryptographic couple. Since their invention in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman in their paper "A method for obtaining digital signatures and public-key cryptosystems" [RSA78], they have become familiar archetypes in the cryptography field.

Through this thesis, various cryptographic scenarios will be imagined between Alice and Bob. The two either want to secretly exchange messages, either authenticate on some platform, sign documents and exchange them, etc. Moreover, a thing to remember about Alice is that she is an Internet addict and she uses various Cloud services, storage for her personal data, outsourced computation, remote software, etc. Occasionally a third party will appear in the story, his name will be Charlie or Oscar. Other remarkable characters in this crypto journey are the villains; Eve is the passive and submissive eavesdropper, Mallory the active malicious attacker. Often, an adversary will interfere and make the lives of Alice and Bob complicated.

List of Characters

Alice. The main character, a tech addict, always connected.
Bob. Alice's secret lover, he wants to privately exchange messages or cryptographic keys with Alice.
Charlie. A generic third participant.
Eve. An eavesdropper, who is usually a passive attacker. While she can listen in on messages between Alice and Bob, she cannot modify them.
Mallory. A malicious attacker. Unlike passive Eve, Mallory is an active attacker, who can modify messages, substitute messages, or replay old messages. The difficulty of securing a system against Mallory is much higher than against Eve.
Oscar. An opponent, similar to Mallory, but not necessarily malicious. He is not trustworthy. Nevertheless, he is delegated data to; he performs tasks for other parties, such as large computations.
Arthur and Merlin. Arthur asks questions and Merlin provides answers. Merlin has unbounded computational ability (like the wizard Merlin). In interactive proof systems, Merlin claims the truth of a statement, and Arthur (like King Arthur), questions him to verify the claim.
SNARK. *"For the Snark's a peculiar creature, that won't*
Be caught in a commonplace way.
Do all that you know, and try all that you don't:
Not a chance must be wasted to-day!"
Lewis Carroll

1.2.2 The Main Security Goals

Cryptographic primitives and protocols are designed to maintain a desired set of security goals even under attempts at making them deviate from the expected functionality. The most common security requirements bursting from applications are:

Privacy/Confidentiality. This is the main idea people associate to the term of *cryptography*. Keeping information secret from all, but those who are authorized to see it. Confidentiality is the protection of transmitted data from passive attacks, by preventing disclosure of information to unauthorized parties. In a nutshell, a cryptographic scheme or protocol between Alice and Bob achieves confidentiality if Alice is able to send messages to Bob in such a way that only Bob can read the messages and Eve is not able to see the actual content of their communication. Encryption is the main cryptographic primitive to provide confidentiality.

Authentication. The process of proving one's identity. This property can refer to both data and user authentication. In the case of user authentication, this functionality ensures that Alice is whom she claims to be. For message authentication, the goal is to provide some additional information that guarantees to Bob that Alice originated the message he received. In particular, no undesired third party, Mallory, should be able to

impersonate Alice. Digital Signatures (DS) and Message Authentication Codes (MAC) are the two cryptographic primitives that guarantee data authentication.

Integrity. Ensuring the information has not been altered by unauthorized or unknown means. Data integrity provides assurance that data has not been modified in an unauthorized manner after it was created, transmitted or stored. This means that there has been no insertion, deletion or substitution done with the data. In a communication between Alice and Bob, one must have the ability to detect data manipulation by adversarial parties. Cryptographic hash functions are a fundamental building block for integrity.

Non-repudiation. Non-repudiation prevents either Alice or Bob from denying a message. Thus, when Bob sends a message to Alice, he cannot change his mind and Alice can prove that the message was, in fact, sent by the alleged sender.

1.3 Outsourcing Computation

Cloud computing is a revolutionary movement in the area of IT industry that provides storage, computing power, network and software as an abstraction and as a service on demand over the Internet, which enables its clients to access these services remotely from anywhere, anytime via any terminal equipment. Moreover, the clients are no longer restricted to their limited CPU, storage, and bandwidth; they are empowered to use the cloud resources to execute large computations.

Since the Cloud has modified the definition of data storage and computation from personal computers to the huge data centers, security and integrity of data have become some of the major concerns that should be taken into account by the developers of the Cloud. While clients are willing to outsource many tasks and resources to Cloud providers for the sake of several benefits, how can they be assured that the Cloud will operate correctly? The business providing the computing services may have a strong financial incentive to return incorrect answers, if such answers require less work and are unlikely to be detected by the clients. On the other side, some services are provided for "free", in exchange for the possibility to harvest users information in order to create tailored advertising campaigns, extract statistics or trade data to generate profits. The clients may not agree to lose control of their personal data or to divulge sensitive information to unknown parties.

Two main challenges for cryptographers arise here: find solutions that are able to guarantee the integrity of the results for the outsourced computations and/or the confidentiality of client's data.

More importantly, while privacy and integrity were long studied questions in cryptography, the key difference is that cryptographic solutions for the cloud setting must preserve the benefits of cloud computing. This means, for example, that these solutions must make sure that the users do less work than what they outsourced to the cloud. It is also desirable to keep the work performed by the workers as close as possible to the amount of work needed to complete the original task.

These questions motivated the study of secure delegation of computation and further research on proof systems, especially on a specific class of protocols, called SNARKs. SNARKs are proof systems that enable users to securely outsource computations to untrusted cloud providers in a verifiable manner. These protocols elegantly solve the problem of limited

computational power and public trust, providing to the worker a way of proving correctness of computations to the client in a way so the client can efficiently verify the result.

The other concern that we highlighted is the privacy of data while outsourcing storage and computation. The user's data can carry sensitive information such as personal health records, financial records, trends of stock, scientific research records, just to list a few. Therefore, we would like to use cryptography, for example, to encrypt the data before outsourcing it to the Cloud server to maintain confidentiality. However, the traditional encryption schemes would not allow the untrusted machine to perform the desired computation on the encrypted values.

More advanced techniques, called fully-homomorphic encryption allow a worker to compute arbitrary functions over encrypted data, but they do not suffice to secure the outsourced computation. Indeed, fully-homomorphic encryption provides no guarantee that the worker performed the correct computation and they are very inefficient to use together with a general verification tool such as the above mentionned SNARK.

We will examine all the current known approaches, their limitations and try to find better solutions along this thesis.

1.4 Proof Systems in Cryptography

Goldreich argued in [Gol95] that the notion of proof in cryptography is somewhat different from the concept of a proof in a strict mathematical sense and more similar to a dynamical way of understanding the proof by interaction and interpretation used by humans. Mathematical proofs are more strict; they have a static and formal nature as they are either self-evident or are obtained from earlier rules and axioms.

However, humans tend to use a more intuitive sense of proofs where the soundness of a statement is established through the process. In a similar sense, in a cryptographic proving protocol, instead of presenting a static proof for the claim, the prover tries to convince the verifier by exchanging messages with it, a kind of questions & answers process.

A *proof system* in the cryptographical sense, is an interactive protocol by which one party (called the prover) wishes to convince another party (called the verifier) that a given statement is true. In zero-knowledge proof, we require further that the proof does not reveal anything more than the truth of the statement. At a first glimpse, it sounds counter-intuitive, being able to prove something is correct, without revealing any extra detail.

Let's see that it is perfectly possible by a very simple day-to-day example:

Example 1.4.1 (Playing card)**.** *Imagine that we pick a card* **A♢** *from a complete deck of playing cards and we want to prove to an adversary that we have a* ***red*** *card in our hand. We can prove that by revealing more information than expressed in the statement, just by showing our card,* **A♢**. *Alternatively, we can choose to prove nothing more than the colour of our card by revealing to the adversary all the black cards* ♣, ♠ *left in the deck. Our opponent should now be convinced we have a red card in our hands, but it did not learn anything else about the value of our card.*

Researches in zero-knowledge proofs have been prompted by authentication systems where one party wants to prove its identity to a second party via some secret information such as a password but doesn't want to disclose anything about its secret password to the second party. This is called a zero-knowledge *proof of knowledge*.

A proof of knowledge is an interactive protocol in which the prover succeeds in "convincing" a verifier that it knows something (a password, the steps of a computation, etc.) associated

with the statement. For example, if the statement is "I am Alice.", the prover should show knowledge of the secret password of Alice; if the statement is "I computed the function $f(x)$ and obtained y.", then the prover must convince its verifier that it knows all the steps of this computation that lead to the result y.

What it means for a machine to *have knowledge* is defined formally in terms of an extractor. As the program of the prover does not necessarily spit out the knowledge itself (as is the case for zero-knowledge proofs), we will invoke another machine, called the knowledge extractor that, by having access to the prover, can extract this witness (the knowledge).

The next step is the introduction of non-interactive proof systems, which reduce the number of rounds of interaction between the prover and the verifier to only one. Some non-interactive protocols consist in only one message from the prover to verifier; others need the verifier to generate some setting information, called CRS, that can be made publicly available ahead of time and independently of the statement to be proved later. To enforce security, and avoid cheating from the verifier, this CRS is often generated by a third trusted party.

1.5 SNARKs

In the class of non-interactive proofs, a particularly interesting concept for proving integrity of results for large computations is that of SNARK, i.e., *succinct non-interactive argument of knowledge*. By this term, we denote a proof system which is:

***succinct*:** the size of the proof is very small compared to the size of the statement or the witness, i.e., the size of the computation itself,

***non-interactive*:** it does not require rounds of interaction between the prover and the verifier,

***argument*:** we consider it secure only for provers that have bounded computational resources, which means that provers with enough computational power can convince the verifier of a wrong statement,

***knowledge-sound*:** it is not possible for the prover to construct a proof without knowing a certain so-called witness for the statement; formally, for any prover able to produce a valid proof, there is an extractor capable of extracting a witness ("the knowledge") for the statement.

SNARK systems can be further equipped with a *zero-knowledge* property that enables the proof to be done without revealing anything about the intermediate steps (the witness). We will call these schemes zk-SNARKs.

A (zk-)SNARK protocol (as any other non-interactive proof system) is described by three algorithms that work as follows:

- Gen is the setup algorithm, generating a necessary string crs used later in the proving process and some verification key vrs, sometimes assumed to be secret to the verifier only. It is typically run by a trusted party.
- Prove is the proving algorithm that takes as input the crs, the statement u and a corresponding witness w and outputs the proof π.
- Verify is the algorithm that takes as input the verification key vrs, the statement u and the proof π, and returns 1 "accept" the proof or 0, "reject".

The SNARK schemes can be used for delegating computation in the following way: a server can run a computation for a client and non-interactively prove the accuracy of the

result. The client can verify the result's correctness in nearly-linear time in the size of the input (instead of running the entire computation itself).

The goal of this thesis is to study these protocols, try to construct new efficient and secure schemes and broaden their use to more applications.

1.6 Our Contributions

In this manuscript, we actively study the SNARK protocols, by proposing new constructions, by revisiting the security analysis of existing schemes, by finding new applications, or by combining SNARKs with other primitives in order to obtain further functionalities.

We expand below on the main contributions that are developed in this thesis and outline some of their implications. The results discussed along this manuscript have been published (or are in review process) in [GMNO18] (co-authored with Rosario Gennaro, Michele Minelli, Michele Orrù), [FN16] (co-authored with Dario Fiore) and [FNP18] (co-authored with Dario Fiore and David Pointcheval).

All along the five chapters, we place ourselves in the context of verifiable computation (VC): we study the setting where a party with limited storage and computation resources outsources its data and asks a powerful server located in the Cloud to run computation jobs on it maintaining verifiable results.

1.6.1 A Survey on SNARKs

The first part of this thesis – Chapter 3 – introduces the object of our study, the SNARKs. This part provides a broad overview, starting from explaining the motivations for proof systems in cryptography and giving some early context, then recalling the SNARK history from the first plausible construction, through successive improvements, to today most efficient instantiations. We present in this chapter some of the most outstanding constructions in the recent years. Our focus is mainly in detailing the idea of the framework presented by Gennaro et al. for SNARKs from Quadratic Span Programs (QSP) – QSP is a polynomial problem used to represent computations. This part is of independent interest and can be seen as a background necessary to understand the further contributions presented in this thesis.

1.6.2 Post-Quantum SNARKs

In Chapter 4, we propose a new framework for building SNARKs that exploits the advantages of previous constructions. It uses a new representation of a boolean circuit satisfiability problem, SSP of Danezis et al. that offers some improvements over the QSP of Gennaro et al. Also, this new framework minimizes the hardness assumptions necessary for proving the security of our SNARK scheme. In a nutshell, in order to construct succinct proofs of knowledge using our framework, one must use the following building blocks:

- an SSP, a way of *translating* the computation into a polynomial division problem, meaning that we reduce the proof of the computation to the proof of a solution to this SSP problem,
- an Enc encoding scheme that hides scalar values, but allows linear operations on the encodings for the prover to evaluate polynomials, and some quadratic check property for the verifier to validate the proofs,

- a CRS generator that uses this encoding scheme to hide a secret random point s and all the necessary powers needed later by the prover to compute polynomial evaluations on s.

The idea of the SNARK is simple; the prover has to convince the verifier that it knows some polynomials, such that a division property between them holds (a solution to SSP problem). Instead of sending the entire polynomials as a proof, it evaluates them in a secret point s (hidden by the encoding) to obtain some scalars. The verifier, instead of checking a polynomial division, has only to check a division between scalars, which makes the task extremely fast.

Besides the improvements mentioned above, one of the most remarkable features of this framework is the fact that it can accommodate for lattice-based encodings, meaning that we can use it to obtain a quantum-resilient SNARK.

We show indeed how to build zero knowledge SNARKs from lattice assumptions, that are claimed to withstand quantum attacks. Compared with previous works, our result achieves zero-knowledge, relies on weaker assumptions, and is simpler and more efficient. Also, for a reasonable set of parameters, we achieve post-quantum security. On the downside, the size of our proofs is considerably larger than in previous works. Moreover, our construction is designated-verifier, meaning that the verification procedure requires access to some secret key, as opposed to publicly verifiable, where the verification procedure can be run by anyone based solely on public information.

1.6.3 O-SNARKs

This result was motivated by the multiple applications of the SNARK protocols. SNARKs are nowadays used not only for delegating computation but also in electronic cryptocurrencies, authentication protocols, anonymous credentials, etc.

As mentioned while introducing SNARKs, one remarkable property that makes it so appealing for many applications is the ”knowledge soundness”. Recall that assuming the prover knows the witness, can be formally stated in terms of some *knowledge extraction*: there exists an extractor, that whenever the prover returns an accepting proof, is able to output a valid witness for the statement.

In this definition, we consider the extractor as a non-black-box algorithm that takes the same input as the prover and any auxiliary information that the prover also had when computing the proof. It can intuitively be seen as a machine that has access to the code of the prover and can extract from it essentially all the knowledge.

In Chapter 5 we study the feasibility of extraction by looking at a scenario in which adversarial provers are given *black-box access*, or what we call access to an *oracle*. For this setting, we study if and under what assumptions such provers can admit an extractor.

We define a new notion, O-SNARKs, that is different from the well-studied SNARKs, by allowing adversaries against the scheme to have Oracle access. We inspect if this notion is worth being defined and we give some satisfactory positive results, candidates for O-SNARK, in the random oracle model, in the limited setting where the prover makes non-adaptive queries to the Oracle, or when the query space is bounded.

Unfortunately, on the negative side, we show an impossibility for extraction with oracles in the *standard model*. We build a counterexample for a special signing oracle and then further analyse different signing oracles to understand the limitation of those.

This study of O-SNARKs is of high theoretical importance, as it establishes a framework that eases the analysis and the use of SNARK extractors when adversarial provers are given access to an oracle.

1.6.4 Applications of O-SNARK

In the following chapter – Chapter 6 – we are recognizing the need of using O-SNARKs by explicitly applying them to larger protocols where this new definition is imperative. We present some well-known homomorphic authentication schemes that can be instantiated with O-SNARKs and that are impossible to prove secure using classical SNARKs (unless some limitations are imposed on provers and signing oracles).

In the case of aggregate signatures, we further examine different ways to construct such schemes, ones that require O-SNARKs and other weaker versions that can be instantiated only with classical SNARKs. We develop non-standard proof techniques for our aggregator scheme in order to bypass the impossibility results for SNARKs with Oracles.

1.6.5 SNARKs for Data Privacy

In this last part, concentrated in Chapter 7, we try to answer to another concern that arises when we outsource computation, the privacy of the client's data.

We try to find solutions for delegated computation, that take into consideration along with the integrity problem, the confidentiality of the data.

To achieve this, we study the possibility of constructing cryptographic protocols that enable verification of computations over encrypted data. The building blocks are an encryption scheme that allows for homomorphic computations over the encrypted data and some proofs of knowledge that allow verification of these computations. In this scenario, the client must be able to efficiently verify the correctness of the result despite not knowing (anymore) the inputs of the delegated computation.

This problem is addressed by verifiable computation (VC) with input privacy, a notion that is close to both the active areas of homomorphic encryption and verifiable computation. However, in spite of the efficiency advances in the respective areas, VC protocols that guarantee the privacy of the inputs are still expensive. The only exception is a protocol by Fiore, Gennaro and Pastro (CCS'14) that supports arithmetic circuits of degree at most 2. In this chapter, we propose new efficient protocols for VC on encrypted data that improve over the state of the art solution of Fiore et al. in multiple aspects. First, we can support computations of degree higher than 2. Second, we achieve public delegability and public verifiability whereas Fiore et al. need the same secret key to encode inputs and verify outputs. Third, we achieve a new property that guarantees that verifiers can be convinced about the correctness of the outputs without learning information on the inputs. The key tool to obtain our new protocols is a new SNARK that can efficiently handle computations over a quotient polynomial ring, such as the one used by Ring-LWE somewhat homomorphic encryption schemes.

1.6.6 Other Contributions

In addition to the contributions outlined above and developed in this manuscript, we have worked during our thesis on other problems related to Cloud Computing and secure delegation

of data, in [ACNP16]. This result is not described in this thesis as it is not related to the main topic, the SNARK protocols.

For better security, as we have seen, it is recommended that users encrypt their data before outsourcing it to the Cloud. However, this leads to a key management issue: Users have to remember cryptographic keys. Humans cannot remember large secret keys, but just low-entropy passwords (and not too many).

Password-protected secret sharing (PPSS) schemes allow a user to publicly share its high-entropy secret across different servers and to later recover it by interacting with some of these servers using only his password without requiring any authenticated data [Jab01]. In particular, this secret will remain safe as long as not too many servers get corrupted. However, servers are not always reliable, and the communication can be altered. To address this issue, a robust PPSS should additionally guarantee that a user can recover his secret as long as enough servers provide correct answers, and these are received without alteration.

In [ACNP16], we propose new robust PPSS schemes which are significantly more efficient than the existing ones. Our contributions are two-fold: First, we propose a generic technique to build a Robust Gap Threshold Secret Sharing Scheme (RGTSSS) from some threshold secret sharing schemes.

Then, we use this new approach to design two new robust PPSS schemes that are efficient, from two OPRFs. Our schemes are proven in the random-oracle model, just because our RGTSSS construction requires random non-malleable fingerprints, which are provided by an ideal hash function.

Our main contribution in designing PPSS schemes is the efficient realization of the robustness, that consists of a single check at the very end of the protocol, during the secret reconstruction. We point out that, in order to achieve robustness in a PPSS protocol, one does not need to distinguish between correct and incorrect shares at each individual evaluation with a server as required in prior works.

1.7 Organization of the manuscript

This thesis is organized into seven chapters as follows:

Chapter 1 is the present introduction.

Chapter 2 introduces the notation used in the manuscript, and gives some definitions and some general notions.

Chapter 3 presents proof systems and notably SNARKs in detail, and constitutes a sort of survey on the area; it is mostly presented as a high-level introduction to the subject and contains many simplifications and examples to help the reader understand step-by-step the frameworks of diverse SNARK constructions.

Chapter 4 is a new construction of SNARK from lattices that also introduces a new general framework for such protocols that realizes a trade-off between the number of underlying hardness assumptions and the proof size.

Chapter 5 is more theoretical, it represents an extensive analysis of the security notion of knowledge soundness and the extraction for the SNARK. It is a very technical chapter, a good understanding of the concept is needed.

Chapter 6 gives various applications of the previously defined notion. It is highly recommended to read Chapter 5 before.

Chapter 7 considers the other side of the coin for outsourced computations, and try to

answer to a more challenging problem, achieving privacy of the data together with guarantees on the computation. It uses techniques and tools from all the previous chapters to construct a new dedicated SNARKs for this particular need.

Chapter 8 draws some conclusions and outlines several questions that remain open, and that can be the topic for extending the research presented in this manuscript.

Alice in Cryptoland

To make the reading of this manuscript more enjoyable, all the chapters will have a parallel storyline, a ludic way of motivating and describing the interest of the crypto notions discussed.[a]
The main character of our story is Alice, a young girl discovering a new world, Cryptoland, an allusion to Lewis Caroll famous story. A reader not interested in the technical and scientifical content of this thesis can still follow the great adventures of Alice in this magic Cryptoland and identify itself with the character. Most of the problems and struggles encountered by our fictional Alice character can be also day-to-day life concerns for most of us and surprisingly, the answer to them comes from Cryptography.[b] I hope that this childish, naive story will raise awareness of the role and importance of our work in the real world and arouse some curiosity to better understand the concepts and techniques used in the field of cryptography.

[a] This attempt of crypto outreach was inspired to me by the novel *Sophie's World* [Gaa96] by Norwegian writer Jostein Gaarder.
[b] We warn the reader that Alice is not always a moral example to follow.

Chapter 2

Preliminaries

Notations, Assumptions and first Primitives. In this chapter, we introduce the notation and basic assumptions and primitives employed throughout this manuscript. We start by recalling some standard mathematical and computational concepts, and by briefly explaining provable security. We also remind some well-known number-theoretic assumptions, and define most of the cryptographic primitives involved in this work, including one-way functions, lattice-based encryption, digital signatures, commitment schemes. Most of the notions introduced in this chapter are rather usual; thus it can be easily skimmed through.

Contents

2.1 Notation and Preliminaries

2.1.1 Mathematical Notions

Sets, Integers, Functions, Bit Strings. We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$ and the set of non-negative integers by $\mathbb{N}$. If S is a set, then $\mid S \mid$ denotes its size. For two sets D and R, we denote by $\mathcal{F}(D, R)$ the set of all functions $f : D \to R$ with domain D and range R.

For two integers a, b such that $a < b$, we write $\{a, \ldots, b\}$ or $[a, b]$ to denote the set of integers between a and b (a and b included). If $a = 1$, then we use $[b]$ for $\{1, \ldots, b\}$.

The set of all bit strings is denoted by $\{0,1\}^*$, while the set of bit strings of length $n \in \mathbb{N}$ is $\{0,1\}^n$. A bit string is often denoted by a lowercase letter, such as x or y. If $x \in \{0,1\}^*$, its length is $|x|$ and the i-bit of x is x_i (for $i \in [|x|]$).

Modular Arithmetic. For a positive integer n, we denote by $(\mathbb{Z}_n, +)$ the additive group of integers modulo n and by $(\mathbb{Z}_n, +, \cdot)$ the ring of integers modulo n. We often abuse this notation by using $\mathbb{Z}_n$. For an integer $x \in \mathbb{Z}$, the reduction of x modulo n, denoted $x \mod n$, is the remainder of the Euclidean division of x by n. Furthermore, we denote by $(\mathbb{Z}_n^*, \cdot)$ or simply $\mathbb{Z}_n^*$ the group of units of the ring $\mathbb{Z}_n$ called also the multiplicative group of integers modulo n. Note that when n is a prime number, denoted by p, which will often be the case in this thesis, $\mathbb{Z}_p$ is also a field, and $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$. For arbitrary integers n, the size of $\mathbb{Z}_n^*$ (the number of invertible elements modulo n) is given by Euler's totient function, which we denote $\sigma(n)$. It corresponds to the number of integers k between 1 and n such that $\gcd(k, n) = 1$, where gcd denotes the greatest common divisor.

Cyclic Groups. We recall that a cyclic group is a finite group generated by a single element. In particular, a cyclic group is commutative (or abelian). Throughout this manuscript, we denote by a tuple $(p, \mathbb{G}, g)$ a multiplicative cyclic group of prime order p generated by an element g, so that $\mathbb{G} = \langle g \rangle = \{1, g, ..., g^{p-1}\}$, where 1 denotes the identity element. We often abuse this notation by using the notation $\mathbb{G}$ when the order and the generator are clear from the context. We also assume that the multiplication over $\mathbb{G}$ is an efficient operation, so given g and $x \in \mathbb{Z}_p$, one can efficiently compute g^x.

Bilinear Groups. A bilinear group is given by a description $(n, \mathbb{G}, \mathfrak{G}, \mathbb{G}_T, \mathsf{e})$ such that

- n is prime or composite;
- $\mathbb{G}, \mathfrak{G}$ are groups of order kn and $\mathbb{G}_T$ of order ln for $k, l \in \mathbb{N}$ and all the elements of these groups are of order at most n;
- $\mathsf{e} : \mathbb{G} \times \mathfrak{G} \to \mathbb{G}_T$ is a bilinear asymetric map (pairing), which means that $\forall a, b \in \mathbb{Z}_n$: $\mathsf{e}(g^a, \mathfrak{g}^b) = \mathsf{e}(g, \mathfrak{g})^{ab}$;
- if $\mathbb{G}$ and $\mathfrak{G}$ are cyclic and g and $\mathfrak{g}$ are the generators of $\mathbb{G}$ and $\mathfrak{G}$ respectively, then $\mathsf{e}(g, \mathfrak{g})$ generates $\mathbb{G}_T$;
- membership in $\mathbb{G}, \mathfrak{G}, \mathbb{G}_T$ can be efficiently decided, group operations and the pairing e are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each have linear size.

Polynomials, Rings. For any ring $\mathbf{R}$ and any integer k, we denote by $\mathbf{R}[X_1, \ldots, X_k]$ the ring of multivariate polynomials in indeterminates $X_1, \ldots, X_k$ over $\mathbf{R}$ and coefficients in $\mathbf{R}$. We call $\mathbf{R}[X_1, \ldots, X_k]_{\leq d}$ its subspace containing only polynomials whose degree in each

indeterminate is at most d. For a polynomial $P \in \mathbf{R}[X_1, \ldots, X_k]$, we denote by $P(a_1, \ldots, a_k)$, its evaluation in $(a_1, \ldots, a_k) \in \mathbf{R}^k$ by setting $X_1 = a_1, \ldots, X_k = a_k$.

In the same manner, we use $\mathbf{R}[X]$ for polynomials in the variable X with coefficients in $\mathbf{R}$. For some polynomial $R \in \mathbb{Z}[X]$ of degree d, we consider the quotient ring $\mathbb{Z}[X]/\langle R(X)\rangle$. For a prime $q \gg d$ we define $\mathbb{F} = \mathbb{Z}_q$ a finite field and $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$.

When there is no risk of confusion or conflict with other notations, we may denote the mono-variate polynomials by lowercase letters $v(x) \in \mathbb{F}[x]$ (we employ this notation in Chapters 3 and 4).

Vectors, Matrices. Vectors are denoted by lower-case bold letters, like $\mathbf{v}$, or in some situations with an arrow as in $\vec{c}$. We indicate a vector $\mathbf{v}$'s i-th entry by v_i (not in bold). We use $\mathbf{0}$ for the zero-vector. We use $\mathbf{v}^\intercal$ for the transpose of a vector $\mathbf{v}$, which is then a row. For a vector $\mathbf{v}$, we define the L_p norm as

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p},$$

and for $p = \infty$, we define $\|\mathbf{v}\|_\infty = \max_i |v_i|$. When we omit the subscript and simply write $\|\mathbf{v}\|$, we refer to the L_2 norm of the vector $\mathbf{v}$. Given two vectors of the same length $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, we use either $\langle \mathbf{a}, \mathbf{b}\rangle$ or $\mathbf{a} \cdot \mathbf{b}$ to denote their inner (or dot) product $\sum_{i\in[n]} a_i\, b_i$.

Matrices are denoted by upper-case bold letters, like $\mathbf{A} \in \mathbb{F}^{n\times m}$. When a square matrix is invertible, we denote by $\mathbf{A}^{-1}$ its inverse.

For a d_1-dimensional vector $\mathbf{a}$ and a d_2-dimensional vector $\mathbf{b}$, we write $(\mathbf{a}\|\mathbf{b})$ to denote the $(d_1 + d_2)$-dimensional vector obtained by concatenating $\mathbf{a}$ and $\mathbf{b}$. We will use a similar notation for concatenating matrices with matrices or matrices with vectors.

Asymptotic Behaviors. When f and g are two functions $f : \mathbb{N} \to \mathbb{R}$, we write $f = \mathcal{O}(g)$ or $g = \Omega(f)$ to indicate that there exist a constant c and an integer $n_0 \in \mathbb{N}$ such that for any integer $n \geq n_0$, $|f(n)| \leq c \cdot |g(n)|$. We say that a function $\varepsilon : \mathbb{N} \to [0, 1]$ is *negligible* (or $1 - \varepsilon$ is *overwhelming*) if, for any constant $c \in \mathbb{N}$, there exists $\eta_0 \in \mathbb{N}$ such that for any $\eta \geq \eta_0$, $\varepsilon \leq \frac{1}{\eta^c}$.

Distributions and Probabilities. Given a set $\mathcal{S}$, we write $x \leftarrow\!\!\$\, \mathcal{S}$ to indicate that x is sampled uniformly at random from $\mathcal{S}$ (independently of everything else). Similarly, if $\mathcal{D}$ is a probability distribution, then we write $x \leftarrow \mathcal{D}$ to denote that x is sampled according to $\mathcal{D}$. We write $\Pr[X = x]$ to indicate the probability of a random variable X taking the value x. Given two distributions X, Y over a finite or countable domain D, their statistical distance is defined as $\Delta(X, Y) = \frac{1}{2}\sum_{v\in D} |X(v) - Y(v)|$. When it is clear from the context, we sometimes write "random" for "uniformly at random".

Miscellaneous. The value *"true"* is represented by 1, while the value *"false"* is represented by 0. Logarithm log is always in basis 2: $\log = \log_2$.

2.1.2 Algorithms

Probabilistic Turing Machines. A Turing machine is an abstract model of computations in which a tape head reads symbols on the tape and performs operations (writing, moving left, moving right) that depend on the symbol that is read.

A probabilistic Turing machine is a multi-tape Turing machine 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. The 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. For simplicity, unless otherwise stated, all algorithms discussed in this work will be probabilistic Turing machines.

Random Access Machines. A random access machine (RAM) is a simple model of computation. Its memory consists of an unbounded sequence of registers. Each of the registers may hold an integer value. This model assumes a single processor. In the RAM model, instructions are executed one after the other, with no concurrent operations. This model of computation is an abstraction that allows us to compare algorithms on the basis of performance. The assumptions made in the RAM model to accomplish this are: Each simple operation takes one-time step. Loops and subroutines are not simple operations. Each memory access takes one-time step, and there is no shortage of memory. For any given problem the running time of an algorithm is assumed to be the number of time steps. The space used by an algorithm is assumed to be the number of accessed RAM memory cells.

Polynomial-Time Algorithms. We will call a PPT algorithm, or equivalently an efficient algorithm, 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.

For an algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x)$ to denote the fact that we run $\mathcal{A}$ on input x with fresh random coins and we obtain an output y.

For two PPT algorithms $\mathcal{A}, \mathcal{E}$, with the writing $(\mathcal{A}\|\mathcal{E})(x)$ we denote the execution of $\mathcal{A}$ followed by the execution of $\mathcal{E}$ on the same input x and with the same random coins. The output of the two is separated with a semicolon, e.g., $(\mathsf{out}_{\mathcal{A}}; \mathsf{out}_{\mathcal{E}}) \leftarrow (\mathcal{A}\|\mathcal{E})(x)$.

2.2 Provable security

Provable security dates back at least to Shannon's proof that the one-time pad hides all information about the encrypted message [Sha49], but only with the rise of public-key cryptography, which requires constructions with a rich mathematical structure that can also be exploited by cryptanalysts, did provable security really take off. Modern security definitions for cryptographic schemes usually consider a *security game* or experiment, which models how a potential adversary can attack the system. Classic examples are the definitions of CPA/CCA secure public-key encryption schemes [GM84, RS92], unforgeability for signatures schemes [GMR88] or pseudorandomness [GMY82, BM82].

Almost any problem can be solved if enough computational power or enough time is available, e.g., by enumerating and trying every possible solution until finding the right one (this approach is known as *brute-force attack*).

With this in mind, the goal of provable security is then to assess the amount of effort required to solve a certain problem (i.e., break a particular system).

2.2.1 Security Reductions

A cryptosystem is deemed "secure" if breaking it would require an unreasonable effort (with regards to computations, elapsed time, data storage, ...) from a reasonable attacker. The choice of what is reasonable depends on the security model (e.g., what kind of adversary we are considering), and on the security guarantees that we require from the scheme.

To prove the security of a cryptographic scheme, one has to specify

- the algorithmic assumptions: consider a computationally hard underlying mathematical problem which is well known to be intractable by any probabilistic polynomial time algorithm.
- the security notions to be guaranteed: modelized as an experiment where an adversary attacking the scheme is called one or several times with various inputs
- a polynomial reduction: an adversary can be used to construct an algorithm that breaks the underlying assumption

Security Parameter. The previous considerations are numerically represented by a *security parameter*, that we denote with λ: A cryptosystem provides λ bits of security if it requires 2^λ elementary operations to be broken. Although usually omitted in order to ease notation, all the integers, vectors, and matrices that we define will implicitly be a function of λ. However, sometimes we explicitly write that an algorithm (e.g., Setup) takes 1^λ as an argument. The reason for doing this is that we want the algorithm to run in a time that is polynomial in λ, so its input must have size λ.

The parameters of the system will be chosen so that the system is estimated to provide λ bits of security – i.e., such that the best known attack on the system requires 2^λ 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 classical computers.

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 is 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. Running times of Oracles are not taken into account in the running time of the algorithms: a query to an oracle always only counts for one clock cycle.

Adversaries. Adversaries are probabilistic algorithms or Turing machines, which will be denoted with calligraphic letters ($e.g., \mathcal{A}, \mathcal{B}$). They will be usually modeled as efficient algorithms taking 1^λ as input. We will sometimes also consider security against *unbounded* adversaries, which can run in arbitrary time.

Experiments. We often define our security notions or assumptions using experiments, parametrized by the security parameter λ, and during which an adversary is called one or several times with various inputs. An experiment can be seen as a *game* between an adversary $\mathcal{A}$ and an implicit challenger which provides its input to the adversary as well as some oracle access. We write $\mathcal{A}^{\mathcal{O}}(x)$ to say that the adversary $\mathcal{A}$ is called with input x and has access to the oracle $\mathcal{O}$.

Assumptions, Advantage, Indistinguishability. To define an assumption or a security notion, we then define a problem as an experiment or as distinguishing two experiments. We say that the problem is hard if no PPT algorithm can solve it.

The advantage of an adversary $\mathcal{A}$ in an experiment Exp is the probability that this adversary outputs 1 in this experiment:

$$\mathsf{Adv}^{\mathsf{Exp}}_{\mathcal{A}} = \Pr\left[\mathsf{Exp}_{\mathcal{A}}(1^\lambda) = 1\right].$$

When the experiment Exp corresponds to a cryptographic assumption or to a security notion we say that this assumption or property *computationally* holds if the above advantage is negligible negl for any PPT adversary. We further say that it *statistically* holds if the above advantage is negligible for any (even unbounded) adversary. We finally say that it *perfectly* holds if the above advantage is 0 for any (even unbounded) adversary.

Similarly, we define the advantage of an adversary $\mathcal{A}$ in distinguishing two variants of an experiment Exp^0 and Exp^1 over the random guess (we denote the global experiment Exp):

$$\mathsf{Adv}^{\mathsf{Exp}}_{\mathcal{A}} = \left|\Pr\left[\mathsf{Exp}^1_{\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^0_{\mathcal{A}}(1^\lambda) = 1\right]\right|.$$

As before, we say that two games are *computationally* indistinguishable if the advantage of any PPT adversary in distinguishing these two games is negligible. We say that two experiments are *statistically* indistinguishable (resp. *perfectly* indistinguishable or equivalent) if the advantage of any (unbounded) adversary in distinguishing these two games is negligible (resp. 0). For both types of qualities, probabilities are over the random coins of the challenger and of the adversary.

Hybrid Arguments. Most of the security proofs in this thesis are proofs by games (also called hybrid arguments) as defined in [Sho02, KR01, BR06]: to bound an advantage in some game experiment corresponding to some security notion, we construct a sequence of games. The first game is *Hybrid 0* is the experiment itself, while the last game corresponds to some security notion or is such that the adversary just cannot win. Furthermore, we prove that two consecutive games are indistinguishable either perfectly, statistically, or computationally. In other words, we bound the difference of advantages by a negligible quantity. Similarly, to bound an advantage of an adversary in distinguishing two experiments, we construct a sequence of indistinguishable games starting with the first experiment and ending with the second experiment.

Hybrid 0	Hybrid 1
Step 1	Step 1
Step 2	**Step 2 is different**
Step 3	Step 3

Figure 2.1: Two indistinguishable consecutive games in a hybrid argument

2.2.2 Abstract Models of Computation

Computational security proofs in cryptography, without unproven intractability assumptions, exist today only if one restricts the computational model as specified by Maurer [Mau05].

Proving the security of a particular cryptographic system means to exhibit a lower bound on the hardness of a certain computational problem. Unfortunately, for general models of computation no useful lower bound proofs are known, and it is therefore interesting to investigate reasonably restricted models of computation, if one can prove relevant lower bounds for them. In a restricted model one assumes that only certain types of operations are allowed. For example, the generic model which assumes that the properties of the representation of the elements of the algebraic structure (e.g., a group) under consideration cannot be exploited.

Other models used invoke trusted third parties to perform some task without cheating; for example, the public key infrastructure (PKI) model requires a certificate authority, which if it were dishonest, could produce fake certificates and use them to mount a man in the middle attack to read encrypted messages.

Standard Model. Schemes which can be proven secure using only complexity assumptions are said to be secure in the standard model. In this model of computation, the adversary is only limited by the amount of time and computational power available. Security proofs are notoriously difficult to achieve in the standard model, so in many proofs, cryptographic primitives are replaced by idealized versions.

Random Oracle Model. The most usual example of the technique of idealizing is known as the random oracle model (ROM) [BR93, CGH98] and it assumes the existence of a truly random function to which all parties involved in a protocol have access. Since in reality, no such ideal function exists, random oracles are instantiated with hash functions, and one heuristically assumes that a hash function behaves well enough to be a replacement for random oracles. Random oracles allow proving protocols are secure while they are still practically efficient. On the negative side, Canetti, Goldreich and Halevi prove in [CGH98] that there exist signature and encryption schemes that are secure in the ROM, but for which any implementation of the random oracle results in insecure schemes.

Generic Group Model. The generic group model [Sho97, Mau05] is another idealised cryptographic model, where algorithms do not exploit any special structure of the representation of the group elements and can thus be applied in any cyclic group.

In this model, the adversary is only given access to a randomly chosen encoding of a group, instead of efficient encodings, such as those used by the finite field or elliptic curve groups used in practice. The model includes an oracle that executes the group operation. This oracle takes two encodings of group elements as input and outputs an encoding of a third element. If the group should allow for a pairing operation this operation would be modeled as an additional oracle.

One of the primary uses of the generic group model is to analyse computational hardness assumptions. An analysis in the generic group model can answer the question: ”What is the fastest generic algorithm for breaking a cryptographic hardness assumption”. A generic algorithm is an algorithm that only makes use of the group operation, and does not consider the encoding of the group.

Common Reference String Model. The common reference string (CRS) model, introduced by Damgård [Dam00], captures the assumption that a trusted setup in which all involved parties get access to the same string crs taken from some distribution $\mathcal{D}$ exists. Schemes proven secure in the CRS model are secure given that the setup was performed correctly. The common reference string model is a generalization of the common random string model, in which $\mathcal{D}$ is the uniform distribution of bit strings.

2.3 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 specific mathematical problems.

The assumptions we will discuss along this work can be divided into two main categories, discrete-logarithm-based assumptions and lattice-based assumptions. We will also make a separation between *falsifiable* assumptions and *non-falsifiable* or extractable ones.

The discrete-logarithm-based assumptions were for a long time the assumptions underlying most constructions of public-key cryptography (with some noticeable exceptions [McE78]). Even though the last decade has witnessed the emergence of new types of cryptographic assumptions (the most prominent being lattice-based assumptions [Ajt96, Reg05a, HPS98, LPR10a]), 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 the discrete logarithm 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, their security against classical computers is quite well understood, and they enjoy a number of algebraic properties which make them well suited for a vast number of applications and amenable to practical instantiations.

2.3.1 Discrete-Logarithm-Based Assumptions

Given a cyclic group $\mathbb{G}$ of order $n \in \mathbb{N}$ 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 $x \in [n]$ such that $h = g^x$. Generic algorithms to solve discrete logarithm, which are independent of the particular structure of the underlying group $\mathbb{G}$, have a running time proportional to $\sqrt{n}$. In spite of more than four decades of intense cryptanalytic effort, there exist certain groups in which we still do not know any 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 is known. As previously mentioned, the discrete logarithm assumption (hence all assumptions discussed in this section) does not hold against quantum polynomial-time adversaries [Sho99].

The Discrete Logarithm Assumption. Let $\mathbb{G}$ be a cyclic group of order $n \in \mathbb{N}$, with a generator g. The discrete logarithm assumption states that:

Assumption 2.3.1 (DLog)**.** *For any efficient algorithm $\mathcal{A}$, Discrete Logarithm Assumption holds relative to a group $\mathbb{G}$ if:*

$$\mathsf{Adv}^{\mathsf{dlog}}_{\mathcal{A}} = \Pr\left[\mathsf{DLog}_{\mathcal{A}}(\mathbb{G}, g, 1^\lambda) = 1\right] = \mathsf{negl}.$$

where $\mathsf{DLog}_{\mathcal{A}}$ is the experiment depicted in Figure 2.2.

$\underline{\mathsf{DLog}_{\mathcal{A}}(\mathbb{G}, g, 1^\lambda)}$	$\underline{\mathsf{CDH}_{\mathcal{A}}(\mathbb{G}, g, 1^\lambda)}$
$x \leftarrow\!\!\$\, \mathbb{Z}_n$	$(x, y) \leftarrow\!\!\$\, \mathbb{Z}_n^2$
$X := g^x$	$(X, Y) := (g^x, g^y)$
$x' \leftarrow \mathcal{A}(X)$	$Z \leftarrow \mathcal{A}(X, Y)$
return $(x' = x)$	**return** $(Z = g^{xy})$

Figure 2.2: Experiments for DLog and for CDH assumptions.

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 instances over $\mathbb{G}$.

Generic Attacks. A generic deterministic algorithm for computing discrete logarithms in arbitrary groups was designed by Shanks [Sha71], and it is known under the name of *baby-step-giant-step algorithm*. This algorithms needs $\mathcal{O}(\sqrt{n})$ group operations (and comparable space complexity) to solve the problem. An improvement was made by Pollard in [Pol78], whose solution uses only constant space (at the cost of being probabilistic rather than deterministic), and still $\mathcal{O}(\sqrt{n})$ group operations. Pollard's rho algorithm, for which practical improvements were suggested in [BLS11], is the state-of-the-art algorithm for computing arbitrary discrete logarithm in generic groups. However, more efficient algorithms can exist for specific groups, 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 particular groups in which no attack better than Pollard's rho algorithm is known.

Instantiating $\mathbb{G}$ with Finite Fields. One of the most common instantiations of $\mathbb{G}$: Let p be a random large prime. Then $\mathbb{F}_p = \mathbb{Z}_p$ is the field with p elements, and the multiplicative group $\mathbb{F}_p^*$ is a cyclic group of order $p - 1$ where DLog is conjectured to hold.

Instantiating $\mathbb{G}$ with Elliptic Curves. Alternatively, it is common to instantiate the DLog assumptions over elliptic curves, which are algebraic curves defined by an equation of the form $y^2 = x^3 + ax + b$ over a field on which a group operation can be defined. Elliptic curves have been the subject of a very rich study. Group elements are up to ten times smaller over elliptic curves than over finite fields for comparable security requirements.

The Computational Diffie-Hellman Assumption. The CDH 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. Formally, the assumption is stated as follows:

Assumption 2.3.2 (CDH). *For any efficient algorithm $\mathcal{A}$,* CDH *holds relative to a group $\mathbb{G}$ if:*

$$\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{A}} = \Pr\left[\mathsf{CDH}_{\mathcal{A}}(\mathbb{G}, g, 1^{\lambda}) = 1\right] = \mathsf{negl}.$$

where $\mathsf{CDH}_{\mathcal{A},1^{\lambda}}$ is the experiment depicted in Figure 2.2.

It is easy to see that hardness of DLog is implied by hardness of CDH. 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 [dB88] that CDH is as hard as the discrete logarithm problem over $\mathbb{F}_p^*$ if $\phi(p-1)$ is smooth (i.e., its prime factors are small). A general algorithm for solving DLog given access to a CDH oracle in an 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.

2.3.2 Assumptions on Bilinear Groups

As already defined, a bilinear group (see Figure 2.3) is given by a description $(n, \mathbb{G}, \mathfrak{G}, \mathbb{G}_T, \mathsf{e})$ where e is a pairing application or *bilinear map* from $\mathbb{G} \times \mathfrak{G}$ to $\mathbb{G}_T$ with the interesting property that for all exponents a, b we have that $\mathsf{e}(g^a, \mathfrak{g}^b) = \mathsf{e}(g, \mathfrak{g})^{ab}$. Elliptic curves equipped with bilinear maps are commonly used in cryptography, either in the symmetric pairing setting, *i.e.*, $\mathbb{G} = \mathfrak{G}$ or in the asymmetric pairing setting where $\mathbb{G} \neq \mathfrak{G}$.

Bilinear Group $\mathsf{gk} := (p, \mathbb{G}, \mathfrak{G}, \mathbb{G}_T, \mathsf{e})$

- p is a λ-bit prime
- $\mathbb{G}, \mathfrak{G}, \mathbb{G}_T$ are cyclic groups of order q ($\mathfrak{G} = \mathbb{G}$ in the symmetric case)
- $\mathsf{e} : \mathbb{G} \times \mathfrak{G} \to \mathbb{G}_T$ is a bilinear map:

$$\forall a, b \in \mathbb{Z}_p : \mathsf{e}(g^a, \mathfrak{g}^b) = \mathsf{e}(g, \mathfrak{g})^{ab}$$

- if $\langle g \rangle = \mathbb{G}$ and $\langle \mathfrak{g} \rangle = \mathfrak{G}$, then $\langle \mathsf{e}(g, \mathfrak{g}) \rangle = \mathbb{G}_T$

Figure 2.3: Asymmetric bilinear group of prime order.

The q-type Assumptions. The classic standard assumptions (such as DLog, CDH) are not parametrized and always have constant size (are static). Consequently, the assumption, when used in a reductionist proof, is independent of any system parameters or oracle queries and only related to the security parameter. In contrast, non-static *q-type* assumptions are parametrized by q, and they, are actually, a family of assumptions. They may be used in a

static way for a fixed q, but if a reductionists proof relies on the non-static version, then q is usually related to the number of oracle queries an adversary makes, to the size of input or to the number of computational steps necessary in a protocol.

The q–Power Diffie-Hellman (q-PDH). Let the generator $\mathcal{G}$ denote the algorithm by which bilinear groups are generated. $\mathcal{G}$ inputs a security parameter λ and outputs a description of a bilinear group $\mathsf{gk} := (q, \mathbb{G}, \mathfrak{G}, \mathbb{G}_T, \mathsf{e}) \leftarrow_\$ \mathcal{G}(1^\lambda)$. Roughly speaking, the q-PDH assumption says that given $g, g^s, \ldots g^{s^q}, g^{s^{q+2}}, \ldots g^{s^{2q}}$ it is hard to compute the missing element $g^{s^{q+1}}$.

Assumption 2.3.3 (q-PDH)**.** *The q-Power Diffie-Hellman (q-PDH) assumption holds for the bilinear group generator $\mathcal{G}$ if for all* PPT *adversaries $\mathcal{A}$ we have, on the probability space $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $g \leftarrow_\$ \mathbb{G}$ and $s \leftarrow_\$ \mathbb{Z}_p$:*

$$\mathsf{Adv}^{\mathsf{q\text{-}pdh}}_{\mathcal{A}} := \Pr\left[q\text{-}\mathsf{PDH}_{\mathcal{A}}(\mathsf{gk}, 1^\lambda) = 1\right] = \mathsf{negl}.$$

where q-$\mathsf{PDH}_{\mathcal{A}}$ is defined as in Figure 2.4.

A heuristic argument for believing in the q-PDH hardness is given by Groth in [Gro10]. Groth gives a proof to show that the q-PDH assumption holds in the generic bilinear group model. Some variants to this assumptions exist. Sometimes the input of the adversary is extended, also containing the values $\widehat{g}, \ldots, \widehat{g}^{s^q}$, for $\widehat{g} \leftarrow_\$ \mathfrak{G}$ or other auxiliar inputs.

q-$\mathsf{SDH}_{\mathcal{A}}(\mathsf{gk}, 1^\lambda)$	q-$\mathsf{PDH}_{\mathcal{A}}(\mathsf{gk}, 1^\lambda)$
$g \leftarrow_\$ \mathbb{G}$	$g \leftarrow_\$ \mathbb{G}$
$s \leftarrow_\$ \mathbb{Z}_p$	$s \leftarrow_\$ \mathbb{Z}_p$
$\sigma \leftarrow (g, g^s, \ldots g^{s^q})$	$\tau \leftarrow (g, g^s, \ldots g^{s^q}, g^{s^{q+2}}, \ldots g^{s^{2q}})$
$(r, y) \leftarrow \mathcal{A}(\mathsf{gk}, \sigma)$	$y \leftarrow \mathcal{A}(\mathsf{gk}, \tau)$
return $(y = g^{1/(s-r)})$	**return** $(y = g^{s^{q+1}})$

Figure 2.4: Experiments for q-SDH and q-PDH assumptions.

The q–Strong Diffie-Hellman Assumption (q-SDH). The Strong Diffie-Hellman assumption [BB08] says that given $\mathsf{gk}, g \leftarrow_\$ \mathbb{G}$ and a set of powers $(g, g^s, \ldots g^{s^q})$ for a random exponent $s \leftarrow_\$ \mathbb{Z}_p$, it is infeasible to compute $y = g^{\frac{1}{s-r}}$ for a chosen $r \in \mathbb{Z}_p$.

Assumption 2.3.4 (q-SDH)**.** *The q–Strong Diffie-Hellman assumption holds relative to a bilinear group generator $\mathcal{G}$ if for all* PPT *adversaries $\mathcal{A}$ we have, on the probability space $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $g \leftarrow_\$ \mathbb{G}$ and $s \leftarrow_\$ \mathbb{Z}_p$:*

$$\mathsf{Adv}^{\mathsf{q\text{-}sdh}}_{\mathcal{A}} := \Pr\left[q\text{-}\mathsf{SDH}_{\mathcal{A}}(\mathsf{gk}, 1^\lambda) = 1\right] = \mathsf{negl}.$$

where q-$\mathsf{SDH}_{\mathcal{A}}$ is the experiment depicted in Figure 2.4.

A proof in Boneh and Boyen [BB08] shows that the q-SDH assumption holds in the generic bilinear group model.

As for the q-PDH assumption, variants with extended input to auxiliary values exist. We can adapt the q-SDH assumption to hold in the target group $\mathbb{G}_T$. The Target-group Strong Diffie-Hellman (q-TSDH) assumption says that given $(g, g^s, \ldots, g^{s^q})$ it is infeasible to find an $r \in \mathbb{Z}_p$ and to compute $y = \mathsf{e}(g, \mathfrak{g})^{\frac{1}{s-r}}$.

2.3.3 Falsifiable vs. Non-Falsifiable Assumptions

Falsifiable Assumptions. A desired characteristic of a computational hardness assumption is falsifiability, i.e., that if the assumption were false, then it would be possible to prove it. In particular, Naor [Nao03a] introduced a formal notion of cryptographic falsifiability. Roughly speaking, a computational hardness assumption is said to be falsifiable if it can be formulated in terms of a challenge: an interactive protocol between an adversary and a challenger (verifier), where an efficient adversary can convince the verifier to accept if and only if the assumption is false.

Furthermore, the complexity of checking the refutation of the assumption is of interest and should be a major consideration in how acceptable the assumption is. If the assumption is false, then it should be possible to solve the challenge (and the solution can be verified) in time related to the time it takes to break the assumption.

Most standard cryptographic assumptions are falsifiable (e.g., hardness of factoring, *dl*, RSA, CDH, LWE, etc.). Intuitively, assumptions that are not falsifiable are more laborious to reason about, and therefore we have significantly less confidence in them. However, in some cases, in order for a challenger to decide whether an adversary breaks some scheme, the challenger needs to decide whether an NP statement is true or false, which may not be efficient.

Non-Falsifiable Assumptions. The knowledge assumptions are the most common non-falsifiable assumptions that we use in cryptography. They are considered non-standard assumptions, also called *extractability* assumptions, in the sense that any efficient algorithm that succeeds in the associated experiment, there exists a knowledge extractor algorithm that efficiently recovers the said witness.

Knowledge assumptions capture our belief that certain computational tasks can be achieved efficiently only by (essentially) going through specific intermediate stages and thereby obtaining, along the way, some specific intermediate values. A number of different extractability assumptions exist in the literature, most of which are specific number-theoretic assumptions. Usually, these knowledge assumptions can be proven secure in the generic group model. Abstracting from such specific assumptions, one can formulate general notions of extractability for one-way functions and other basic primitives (see [CD09]).

2.3.4 Knowledge Assumptions

The framework of such an assumption is as follows: an extractability assumption considers any PPT algorithm M that, on input a security parameter λ and some *benign* auxiliary input z returns a secret output and a public output. Then, the assumption states that if M satisfies certain efficiency or hardness properties (to be defined later), then for any adversary algorithm $\mathcal{A}$ trying to simulate M, there exists an efficient algorithm $\mathcal{E}_{\mathcal{A}}$ that, given the security parameter, $\mathcal{A}$'s public output and random bits, can compute a matching secret output. Actually, it is more appropriate to talk about a class of extractability assumptions,

varying over the specific algorithms M, and the algorithms $\mathcal{Z}$ that generate the auxiliary input z taken as input by M.

The q-Power Knowledge of Exponent Assumption. This class of assumptions have the following flavor: if an efficient algorithm, given the description of a finite group along with some other public information, computes a list of group elements that satisfies a certain algebraic relation, then there exists a knowledge extractor that outputs some related values that "explain" how the public information was put together to satisfy the relation. The knowledge of exponent (KEA) assumption was the first of this type, introduced by Damgard [Dam92]. It says that given g, g^α in a group $\mathbb{G}$ it is infeasible to create $c, \widehat{c}$ so $\widehat{c} = c^\alpha$ without knowing a such that $c = g^a$ and $\widehat{c} = (g^\alpha)^a$.

$q\text{-}\mathsf{PKE}_{\mathsf{gk},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda)$

$g \leftarrow\!\!\$\ \mathbb{G}, \quad s \leftarrow\!\!\$\ \mathbb{Z}_p$

$\sigma \leftarrow (g, g^s, \ldots g^{s^q}, g^\alpha, g^{\alpha s} \ldots g^{\alpha s^q})$

$z \leftarrow \mathcal{Z}(\mathsf{gk}, \sigma)$

$(c, \widehat{c}; \{a_i\}_i^q) \leftarrow (\mathcal{A} \| \mathcal{E}_\mathcal{A})(\sigma, z)$

return $(\widehat{c} = c^\alpha) \ \wedge \ c \neq \prod_i^q (g^{s^i})^{a_i}$

Figure 2.5: Experiment for q-PKE assumption.

The q-power knowledge of exponent assumption (q-PKE) is a generalization of KEA. It says that given the successive powers of some random value $s \in \mathbb{Z}_p$ encoded in the exponent $\{g, g^s, g^{s^2}, \ldots, g^{s^q}, \ \widehat{g}, \widehat{g}^s, \widehat{g}^{s^2}, \ldots, \widehat{g}^{s^q}\}$ –where either $g \in \mathbb{G}$, $\widehat{g} \in \mathfrak{G}$ for the asymmetric bilinear groups, or $g \in \mathbb{G}$, $\widehat{g} \coloneqq g^\alpha \in \mathbb{G}$ in symmetric ones– it is infeasible to create $c, \widehat{c}$ where $\widehat{c} = c^\alpha$ without knowing $a_0, a_1, \ldots a_q$ that satisfy $c = \prod_{i=0}^q (g^{s^i})^{a_i}$. This is more formally defined (in the symmetric case) by the existence of an extractor:

Assumption 2.3.5 (q-PKE)**.** *The q-Power Knowledge of Exponent (q-PKE) assumption holds relative to a bilinear group given by the description* gk *and for the class* Z *of auxiliary input generators if, for every non-uniform* PPT *auxiliary input generator* $\mathcal{Z} \in \mathsf{Z}$ *and non-uniform* PPT *adversary* $\mathcal{A}$*, there exists a non-uniform* PPT *extractor* $\mathcal{E}_\mathcal{A}$ *such that:*

$$\mathsf{Adv}^{\mathsf{q\text{-}pke}}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}} \coloneqq \Pr\left[\mathsf{q\text{-}PKE}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}} = \mathbf{true}\ \right] = \mathsf{negl},$$

where $\mathsf{q\text{-}PKE}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}}$ *is the game depicted in Figure 2.5.*

These assumptions can be reformulated in terms of "encodings", a generalization of the exponential function in the bilinear group. Roughly speaking, an encoding is a way to hide values; we will formally define encodings and knowledge assumptions for encodings in Chapter 4.

2.3.5 Lattice Assumptions

For quantum computers, solving discrete logarithm problems is easy, so all the hardness assumptions stated until now are not quantum resilient. Fortunately, lattice problems we will discuss in this part are conjectured to be quantum-hard [Pei15]. This makes some lattice-based cryptosystems candidates for post-quantum cryptography.

To begin with, we present some basic definitions and background for lattices and we recall several fundamental computational problems used in post-quantum cryptography.

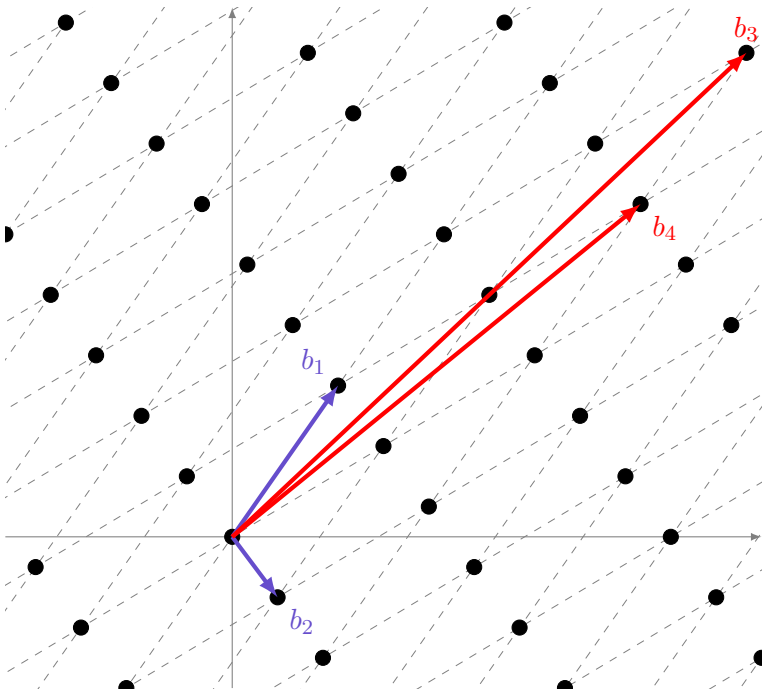


Figure 2.6: A two-dimensional lattice with two different bases.

Lattice Definition, Basis. An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. For an integer $k < n$ and a rank k matrix $\mathbf{B} \in \mathbb{R}^{n \times k}$, $\Lambda(\mathbf{B}) = \left\{\mathbf{Bx} \in \mathbb{R}^n \mid \mathbf{x} \in \mathbb{Z}^k\right\}$ is the lattice generated by the columns of $\mathbf{B}$. The columns of $\mathbf{B}$ form a *basis* of the lattice.

In this work, we are only interested in full-rank lattices, i.e., those for which $k = n$. A lattice basis $\mathbf{B}$ is never unique: for any matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ such that $\det(\mathbf{U}) = \pm 1$, the matrix $\mathbf{B} \cdot \mathbf{U}$ is also a basis of $\Lambda(\mathbf{B})$. In Figure 2.6 we show a two-dimensional lattice with two different bases $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2)$, $\mathbf{B} = (\mathbf{b}_3, \mathbf{b}_4)$.

The *minimum distance* of a lattice Λ is the length of a shortest nonzero lattice vector:

$$\lambda_1(\Lambda) \coloneqq \min_{\mathbf{0} \neq \mathbf{v} \in \Lambda} \|\mathbf{v}\|$$

Fundamental parallelepiped. For any lattice basis $\mathbf{B}$, we define the associated fundamental parallelipiped $\mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{R}^n, 0 \leq x_i < 1 \ \forall i\}$.

We define the determinant of Λ, denoted as $\det(\Lambda)$, as the n-dimensional volume of its fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.

Gaussian Function, Discrete Distribution. For any $\sigma > 0$ the spherical Gaussian function with mean 0 and parameter σ (omitted if 1) is defined as

$$\rho_\sigma(\mathbf{x}) \coloneqq \mathsf{Exp}\left(\frac{-\pi \|\mathbf{x}\|}{\sigma^2}\right)$$

for any $\mathbf{x} \in \mathbb{R}^n$.

For any discrete subset $A \subseteq \mathbb{R}^n$ we define $\rho_\sigma(A) \coloneqq \sum_{\mathbf{x}\in A} \rho_\sigma(\mathbf{x})$, the discrete integral of ρ_σ over A. We then define χ_σ, the discrete Gaussian distribution over A with mean 0 and parameter σ as:

$$\chi_\sigma : A \to \mathbb{R}^+ : \mathbf{y} \mapsto \frac{\rho_\sigma(\mathbf{y})}{\rho_\sigma(A)}.$$

We denote by χ_σ^n the discrete Gaussian distribution over $\mathbb{R}^n$ where each entry is independently sampled from χ_σ. Intuitively, one can think of χ_σ^n as a sphere of radius $\sigma\sqrt{n/(2/\pi)}$ centered around the origin.

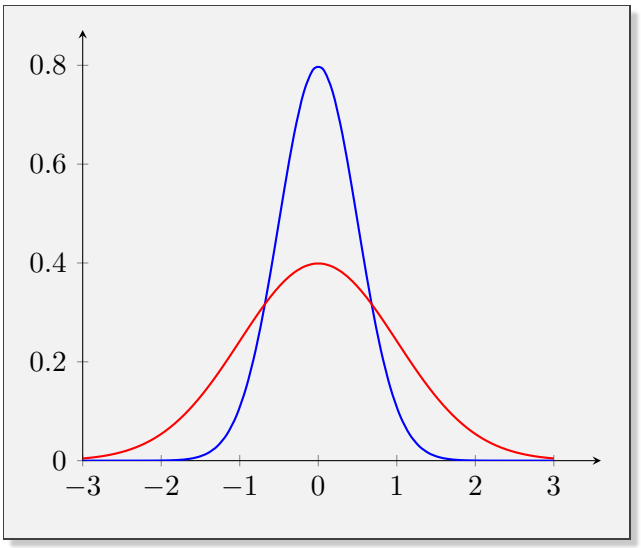


Figure 2.7: One-dimentional Gaussians, in blue $\chi^2_{\sigma_1}$ for parameter $\sigma_1 = \sqrt{2\pi}/2$, in red $\chi^2_{\sigma_2}$ for $\sigma_2 = \sqrt{2\pi}$.

We now report a very well known result about additivity of Gaussian distributions.

Lemma 2.3.6 (Pythagorean additivity of Gaussians)**.** *Let χ_1 and χ_2 be Gaussian distributions with parameters σ_1 and σ_2, respectively. Then χ^+, obtained by sampling χ_1 and χ_2 and summing the results, is a Gaussian with parameter $\sqrt{\sigma_1^2 + \sigma_2^2}$.*

Smoothing parameter. Another important quantity for a lattice Λ is its *smoothing parameter*. Roughly speaking, this can be seen as the minimum amount of Gaussian "blur" required to "smooth out" all the discrete structure of Λ. Consider that one picks a noise vector from a Gaussian distribution with radius at least as large as the smoothing parameter, and reduces the noise vector modulo the fundamental parallelepiped of the lattice, then the resulting distribution is very close to uniform.

This parameter plays a central role in the best known worst-case/average-case reductions for lattice problems, and in a wealth of lattice-based cryptographic constructions. We now give a more formal definition.

Definition 2.3.7 (Smoothing parameter [MR04])**.** *For a lattice* $\Lambda \subseteq \mathbb{R}^n$ *and a positive real* $\varepsilon > 0$*, the* smoothing parameter $\eta_\varepsilon(\Lambda)$ *is the smallest real* $r > 0$ *such that* $\rho_{1/r}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$*, where* Λ^* *is the* dual *of the lattice* Λ *defined as*

$$\Lambda^* \coloneqq \{\mathbf{v} : \forall\ \mathbf{y} \in \Lambda,\ \langle \mathbf{v}, \mathbf{y} \rangle \in \mathbb{Z}\}.$$

2.3.5.1 Computational Problems on Lattices

Lattice problems have received considerable attention as a potential source of computational hardness to be used in cryptography, after a breakthrough result of Ajtai [Ajt96] showing that if certain lattice problems are computationally hard to solve in the worst case, then average-case one-way functions (a fundamental cryptographic primitive that we will define in the next section) exist. The main worst-case lattice problem considered by Ajtai is that of finding a set of n-linearly independent lattice vectors in an arbitrary lattice of length within a polynomial (in n) factor from the shortest such set. This problem in turn, is related, using standard techniques, to various other lattice problems, like approximating the length of the shortest non-zero lattice vector in the worst case, within factors polynomial in n. No polynomial time algorithm is known to solve any of these worst-case problems, so it is reasonable to conjecture that the problems are hard for any polynomial approximation factor.

We present here a class of well-known computational problems on lattices and for ease of presentation we illustrate the solutions for 2-dimensional lattices.

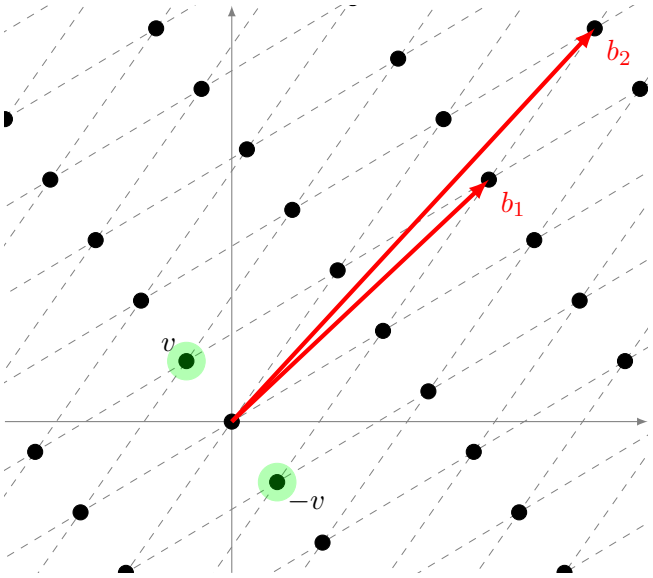


Figure 2.8: Shortest Vector Problem (SVP)

Shortest Vector Problem (SVP). Given an arbitrary basis $\mathbf{B}$ of some lattice $\Lambda = \Lambda(\mathbf{B})$, find a shortest nonzero lattice vector, i.e., a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

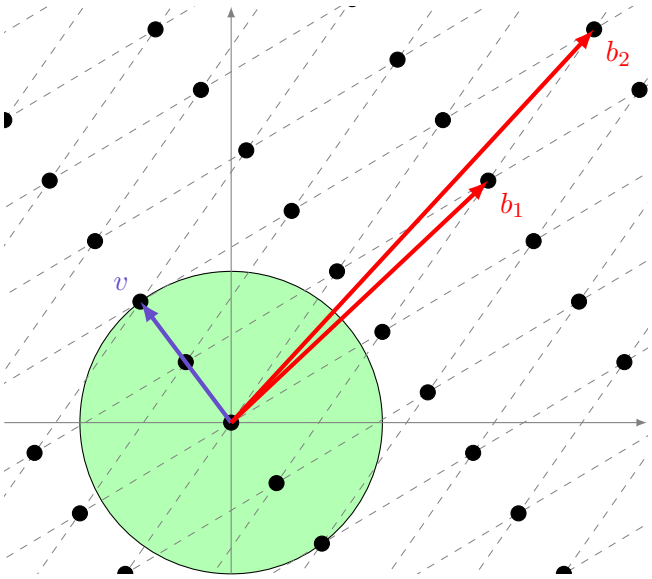


Figure 2.9: Approximate Shortest Vector Problem (SVP_γ).

Approximate Shortest Vector Problem (SVP_γ). Given an arbitrary basis $\mathbf{B}$ of a lattice $\Lambda = \Lambda(\mathbf{B})$, find a nonzero vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\Lambda)$.

Decisional Approximate SVP (GapSVP_γ). Given an arbitrary basis $\mathbf{B}$ of a lattice $\Lambda = \Lambda(\mathbf{B})$, where $\lambda_1(\Lambda) \leq 1$ or $\lambda_1(\Lambda) > \gamma$, determine which is the case.

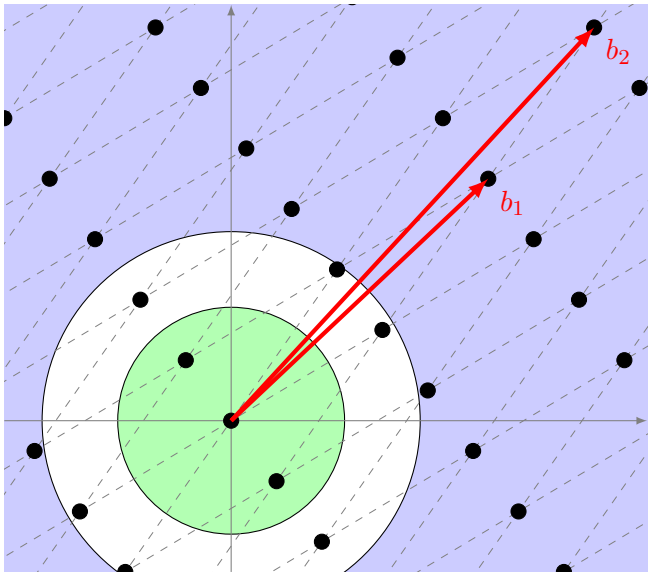


Figure 2.10: Decisional Approximate Shortest Vector Problem (GapSVP_γ).

Approximate Shortest Independent Vector Problem (SIVP_γ). Given an arbitrary basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \Lambda(\mathbf{B})$, output a set $\mathcal{S} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \Lambda$ of n linearly independent lattice vectors such that $\|\mathbf{v}_i\| \leq \gamma \cdot \lambda_1(\Lambda)$ for all $i \in [n]$.

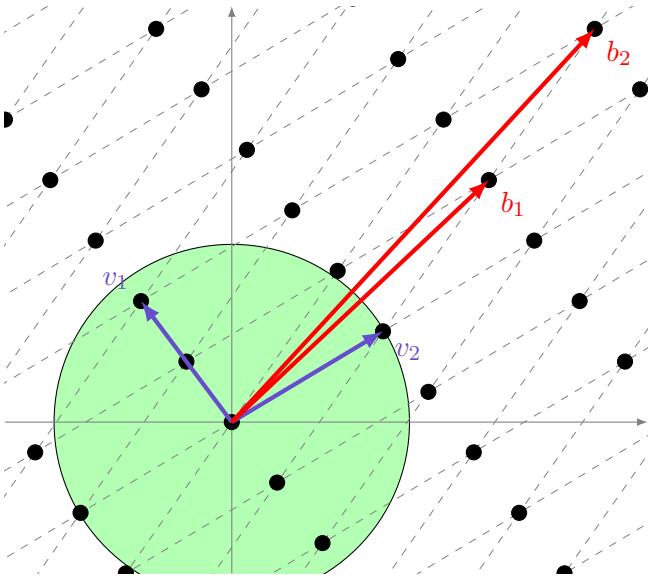


Figure 2.11: Approximate Shortest Independent Vector Problem (SIVP_γ).

Approximate Closest Vector Problem (CVP_γ). Given an arbitrary basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \Lambda(\mathbf{B})$ and a target vector $\mathbf{t} \in \mathbb{R}^n$, find a vector $\mathbf{v} \in \Lambda$ such that $0 < \|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \mathsf{Dist}(\mathbf{t}, \Lambda) = \gamma \cdot \inf_{\mathbf{x} \in \Lambda} \|\mathbf{x} - \mathbf{t}\|$.

Approximate Bounded Distance Decoding Problem (BDD_γ). Given an arbitrary basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = \Lambda(\mathbf{B})$ and a target vector $\mathbf{t} \in \mathbb{R}^n$ such that $\mathsf{Dist}(\mathbf{t}, \Lambda) \leq \gamma^{-1} \cdot \lambda_1(\Lambda)$, find a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v} - \mathbf{t}\| = \mathsf{Dist}(\mathbf{t}, \Lambda)$.

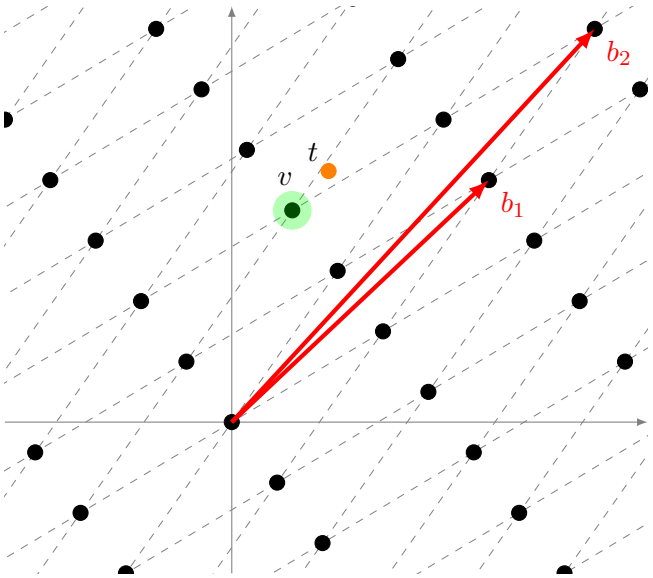


Figure 2.12: Approximate Bounded Distance Decoding Problem (BDD_γ).

The reader is referred to [MG02] for further discussion of these lattice problems. All the previous definitions were parameterized by a positive real valued function $\gamma = \gamma(n)$.

2.3.6 Learning With Errors

The learning with errors (LWE) problem was introduced by Regev in [Reg05b], and has become one of the most known problems in lattice-based cryptography. It has been used to construct several cryptosystems, and it is believed to be hard even for quantum computers. This problem comes in two flavors, search and decision: we present them both in the following.

Parameters, LWE Distribution. LWE is parameterized by two positive integers, n and q, and an error distribution χ over $\mathbb{Z}$, usually a discrete Gaussian of width αq, $0 < \alpha < 1$. We use the notation $\Gamma := (q, n, \alpha, \chi)$, with $q, n \in \mathbb{N}$ for the parameters.

Definition 2.3.8 (LWE distribution)**.** *Given a secret vector* $\mathbf{s} \in \mathbb{Z}_q^n$*, the* LWE *distribution* $\mathsf{LWE}_{\mathbf{s},\chi}$ *over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *is sampled by picking* $\mathbf{a} \leftarrow\!\!\$\, \mathbb{Z}_q^n$*, an error* $e \leftarrow \chi$*, and returning* $(\mathbf{a}, c = \langle \mathbf{s}, \mathbf{a} \rangle + e)$*.*

Search-LWE Assumption. This assumption states that, given m independent samples $(\mathbf{a}_i, b_i) \leftarrow \mathsf{LWE}_{s,\chi}$, for a fixed $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$, it is hard to find $\mathbf{s}$. More formally:

Assumption 2.3.9 (sLWE)**.** *The searching Learning With Errors assumption holds for a parameter generation algorithm* Pgen *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathsf{slwe}}_{\mathsf{Pgen},\mathcal{A}} := \Pr\left[\mathsf{sLWE}_{\mathcal{A}}(\mathsf{Pgen}, 1^\lambda) = \mathbf{true}\right] = \mathsf{negl},$$

where $\mathsf{sLWE}_{\mathcal{A}}(\mathsf{Pgen}, 1^\lambda)$ *is defined as in Figure 2.13.*

Decisional-LWE Assumption. Given m independent samples $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where every sample is either distributed according to $\mathsf{LWE}_{\mathbf{s},\chi}$ for some fixed $\mathbf{s} \in \mathbb{Z}_q^n$ or uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, decisional *lwe* assumption states that distinguishing which is the case is hard.

Assumption 2.3.10 (dLWE)**.** *The decisional Learning With Errors (dLWE) assumption holds for a parameter generation algorithm* Pgen *if for any* PPT *adversary* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathsf{dlwe}}_{\mathsf{Pgen},\mathcal{A}} := \Pr\left[\mathsf{dLWE}_{\mathcal{A}}(\mathsf{Pgen}, 1^\lambda) = \mathbf{true}\right] - 1/2 = \mathsf{negl},$$

where $\mathsf{dLWE}_{\mathcal{A}}(\mathsf{Pgen}, 1^\lambda)$ *is defined as in Figure 2.13.*

Most of the constructions presented in this manuscript will be based on the decisional LWE (dLWE) problem. In [Reg05b], Regev proved the following theorem on the hardness of dLWE:

Theorem 2.3.11 (Hardness of dLWE)**.** *For any* $m = \mathsf{poly}[n]$*, any modulus* $q \leq 2^{\mathsf{poly}[n]}$*, and any (discretized) Gaussian error distribution* χ_σ *of parameter* $\sigma = \alpha q \geq 2\sqrt{n}$ *(where* $0 < \alpha < 1$*), solving the* dLWE *for parameters* $\Gamma := (q, n, \alpha, \chi_\sigma)$ *problem is at least as hard as quantumly solving* GapSVP_γ *and* SIVP_γ *on arbitrary* n*-dimensional lattices, for some* $\gamma = \tilde{\mathcal{O}}(n/2\alpha)$*.*

2.3.6.1 Ring Learning With Errors

A version of the LWE problem over rings was introduced in [SSTX09, LPR10b]. The advantage of this variant instead of plain LWE is compactness and efficiency. In the case of ring-LWE, each sample gives a n-dimensional pseudorandom ring element $c \in \mathbf{R}$, instead of just a pseudorandom scalar $c \in \mathbb{Z}_q$. We can thus say that a single ring-*lwe* sample with $a \in \mathbf{R}$ takes the place of n LWE samples with vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$. Moreover, thanks to techniques like

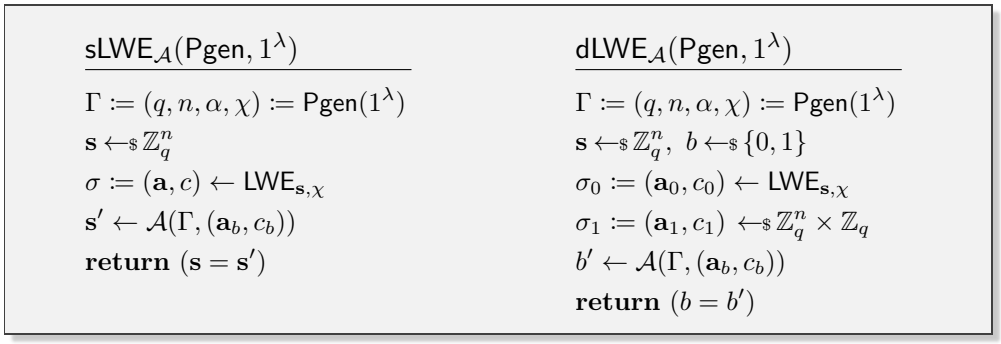


Figure 2.13: The search and decisional LWE games for parameters Γ

FFT, the multiplication between ring elements can be performed in quasi-linear time. The essential drawback of ring-LWE is its conjectured hardness, which is not as well-established as for classical LWE. We will not give details about this, but refer the interested reader to specialized literature (e.g., [APS15a]).

Definition 2.3.12 (Ring-LWE Distribution)**.** *Let $\mathbf{R} = \mathbb{Z}[X]/f(X)$ be a polynomial ring such that f is some cyclotomic polynomial of degree n. Let $q \geq 2$ be an integer modulus, and let $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$ be the quotient ring. Finally, let χ be an error distribution over $\mathbf{R}$. For a fixed secret $s \in \mathbf{R}_q$, the ring-LWE distribution $\mathsf{rLWE}_{s,\chi}$ is sampled by taking $a \leftarrow\!\!\$\ \mathbf{R}_q$, $e \leftarrow \chi$, and outputting $(a, b = s \cdot a + e)$.*

Analogously to what was done for LWE, we now roughly present the two versions of the ring-LWE problem.

Search ring-LWE. Given m independent samples $(a, b) \leftarrow \mathsf{rLWE}_{s,\chi}$, for a fixed $s \leftarrow\!\!\$\ \mathbf{R}$, find s.

Decisional ring-LWE. Given m independent samples $(a, b) \in \mathbf{R} \times \mathbf{R}$, where every sample is either distributed according to $\mathsf{rLWE}_{s,\chi}$ for some fixed $s \in \mathbf{R}$ or uniformly random in $\mathbf{R} \times \mathbf{R}$, distinguish which is the case.

2.3.6.2 Link Between LWE and Lattice-Based Problems

We now clarify the link between LWE and the lattice problems presented in Section 2.3.5.1.

Reduction from sLWE to BDD. Let $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}$, and let Λ be the following lattice

$$\Lambda = \Lambda(\mathbf{A}) := \left\{ \mathbf{A}^\intercal \mathbf{v} : \mathbf{v} \in \mathbb{Z}_q^n \right\} + q\mathbb{Z}^m$$

This is an m-dimensional q-ary lattice, since $\mathbf{A}^\intercal$ has m rows and the lattice is defined modulo the integer q.

Let $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, let χ be an LWE error distribution, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{c} = \mathbf{A}^\intercal \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$. Now it is easy to see that, for sufficiently small error terms, the vector $\mathbf{c}$ is rather close to a specific vector (or point) of the lattice Λ, namely $\mathbf{A}^\intercal \mathbf{s}$, whereas a random vector $\mathbf{u} \leftarrow\!\!\$\ \mathbb{Z}_q^m$ will be far from Λ with high probability. We can then conclude that solving search-LWE amounts to solving an average instance of the BDD problem on the lattice Λ. In fact, once the vector $\mathbf{A}^\intercal \mathbf{s} \in \Lambda$ has been found, $\mathbf{s}$ can be trivially recovered, e.g., by Gaussian elimination.

Reduction from dLWE to GapSVP_γ and SIVP_γ. We already mentioned a result by Regev Theorem 2.3.11 about the hardness of dLWE that reduces to the quantum hardness of GapSVP_γ and SIVP_γ on arbitrary n-dimensional lattices, for some $\gamma = \tilde{\mathcal{O}}(n/a)$. The reduction has been improved since, and the result generalized for rLWE; making the dLWE problem a fruitful assumption to build cryptographic primitives and relying on well-studied assumed-hard problems.

Among other things, LWE was used as the basis of public-key encryption schemes secure under chosen-plaintext attacks [Reg05b, KTX07, PVW07] and chosen-ciphertext attacks [PW08, Pei08], oblivious transfer protocols [PW08], identity-based encryption (IBE) schemes [GPV07, CHKP12, ABB10], various forms of leakage-resilient encryption (e.g., [AGV09, ACPS09, DGK+10, GKPV10]) and more. In addition, LWE is attractive as it typically leads to efficient implementations, involving low complexity operations (often mainly additions).

2.4 Cryptographic Primitives

In this section we give a basic introduction to public-key cryptographic primitives, namely one-way functions, encryption schemes, commitments, signature schemes. We outline some generic algorithms and provide some definitions related to the security of some of these schemes.

2.4.1 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). The existence of such one-way functions is still an open conjecture. In fact, their existence would prove that the complexity classes P and NP are not equal, thus resolving the foremost unsolved question of theoretical computer science. This is nevertheless an assumption that is necessary to prove the existence of almost any cryptosystem (at least from a theoretical point of view).

More formally, for a security parameter denoted by λ:

Definition 2.4.1 (One-Way Function)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if it satisfies the following two conditions:*

Efficient. *Given λ and $x \in \{0,1\}^*$ the value $f(x)$ can be computed in $\mathsf{poly}(\lambda)$ time.*

One-way. *For all PPT algorithms $\mathcal{A}$, the following is negligible (in λ):*

$$\Pr[\mathcal{A}(f(U_\lambda), 1^\lambda) \in f^{-1}(f(U_\lambda))] = \mathsf{negl}.$$

Primitives related to one-way functions exists and we will define one essential in this work, the universal one-way hash function.

Universal One-Way Hash Functions. The notion of universal one-way hash function (UOWHF) families was introduced in [NY89].

Definition 2.4.2 (Universal one-way hash family)**.** *A collection of function families* $\mathbb{H} = \{\mathcal{H}\}_\lambda$ *where each* $\mathcal{H}$ *is a function family* $\mathcal{H} = \{h : \{0,1\}^{q(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}$ *is an universal one-way hash family if:*

Efficient. *The functions* $q(\lambda)$ *and* $\ell(\lambda)$ *are polynomially-bounded; furthermore, given* λ *and* $x \in \{0,1\}^{q(\lambda)}$ *the value* $h(x)$ *can be computed in* $\mathsf{poly}(\lambda)$ *time.*

Compressing. *For all* λ *we have that* $q(\lambda) > \ell(\lambda)$*.*

Universal one-way. *For all PPT algorithms* $\mathcal{A}$*, the following is negligible (in* λ*):*

$$\Pr[x \leftarrow \mathcal{A}(1^\lambda); h \xleftarrow{\$} \mathcal{H}; x' \leftarrow \mathcal{A}(1^\lambda, h, x) : x, x' \in \{0,1\}^{q(\lambda)} \wedge x \neq x' \wedge h(x) = h(x')] = \mathsf{negl}.$$

Collision-Resistant Hash Functions. A collision-resistant hash function (CRHF) is a function ensemble for which it is hard to find two inputs that map to the same output. Formally:

Definition 2.4.3 (Collision Resistant Hash Family)**.** *A collection of function families* $\mathbb{H} = \{\mathcal{H}\}_\lambda$ *where each* $\mathcal{H}$ *is a function family* $\mathcal{H} = \{h : \{0,1\}^{q(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}$ *is collision-resistant if:*

Efficient. The functions $q(\lambda)$ and $\ell(\lambda)$ are polynomially-bounded; furthermore, given λ and $x \in \{0,1\}^{q(\lambda)}$ the value $h(x)$ can be computed in $\mathsf{poly}(\lambda)$ time.

Compressing. For all λ we have that $q(\lambda) > \ell(\lambda)$.

Collision resistant. For all PPT algorithms $\mathcal{A}$, the following probability is negligible (in λ):

$$\Pr[h \leftarrow_\$ \mathcal{H}, (x, x') \leftarrow \mathcal{A}(1^\lambda, h) : x, x' \in \{0,1\}^{q(\lambda)} \wedge x \neq x' \wedge h(x) = h(x')] = \mathsf{negl}.$$

Merkle Trees. We recall here what a Merkle tree or hash tree represents: A hash tree or Merkle tree is a (binary) tree in which every leaf node is labelled with the hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains.

The concept of hash trees is named after Ralph Merkle who patented it in 1979 [Mer79]. Merkle tree (MT) hashing enables a party to use a CRHF to compute a succinct commitment to a long string $\pi \in \{0,1\}^{q(\lambda)}$ and later to locally open to any bit of π (in a succinct manner).

The structure of the tree allows for efficient mapping of arbitrarily large amounts of data and enables easy identification of where changes in that data occur. This concept enables Merkle proofs, with which, someone can verify that the hashing of data is consistent all the way up the tree and in the correct position without having to actually look at the entire set of hashes. Instead, demonstrating that a leaf node is a part of a given binary hash tree requires computing a number of hashes proportional to the logarithm of the number of leaf nodes of the tree; this contrasts with hash lists, where the number is proportional to the number of leaf nodes itself.

2.4.2 Encryption Schemes

Secret-key Encryption. A secret-key encryption scheme consists of the following algorithms:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{k}, \Gamma)$: on input the security parameter λ, outputs the key k and some public parameters Γ;
$\mathsf{Enc}(\mathsf{sk}, m) \to c$: on input the key sk and a message m, outputs a ciphertext c;
$\mathsf{Dec}(\mathsf{sk}, c) \to m$: on input the key sk and a ciphertext c, outputs a message m.

While in a secret-key encryption scheme *the same key* is used both for encrypting and decrypting, in a public-key encryption scheme there are two separate keys, a public and a private (or secret) one, which are used for encrypting and decrypting, respectively.

Public-key Encryption. A public-key encryption scheme consists of the following algorithms:

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk}, \Gamma)$: on input the security parameter λ, outputs the public key pk, the secret key sk, and some public parameters Γ;
$\mathsf{Enc}(\mathsf{pk}, m) \to c$: inputs the public key pk and a message m, outputs a ciphertext c;
$\mathsf{Dec}(\mathsf{sk}, c) \to m$: inputs the secret key sk and a ciphertext c, outputs a message m.

Correctness. We give the correctness definition for public-key encryption, the one for the secret-key follows the same format:

We say that the public-key encryption scheme $\mathcal{E}nc = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is *correct* if, for any $(\mathsf{pk}, \mathsf{sk}, \Gamma) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and a message m, $\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \neq m] = 0$.

We now give a standard definition of security for an encryption scheme, namely that of *indistinguishability under chosen-plaintext attacks* (IND-CPA).

Indistinguishability Under Chosen-Plaintext Attacks. For a public-key encryption scheme $\mathcal{E}nc = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we define IND-CPA security via the game depicted in Figure 2.14. We say that $\mathcal{E}nc$ is IND-CPA secure if, for any PPT adversary $\mathcal{A}$, it holds that

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A}} = \Pr\left[\text{IND-CPA}_{\mathsf{Enc},\mathcal{A}}(1^\lambda)\right] - \frac{1}{2} = \mathsf{negl}.$$

where $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A}}$ is the advantage of an adversary $\mathcal{A}$ when playing the game $\text{IND-CPA}_{\mathcal{E}nc,\mathcal{A}}(1^\lambda)$.

$\underline{\text{IND-CPA}_{\mathcal{E}nc,\mathcal{A}}(1^\lambda)}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\left(1^\lambda\right)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
$b \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(\mathsf{pk}, c)$
return $b' = b$

Figure 2.14: Experiment for IND-CPA security notion.

An Example: Regev's Symmetric Encryption Scheme. We now present as example, a basic version of a well-known encryption scheme based on the LWE assumption (see Assumption 2.3.10), usually referred to as the Regev encryption scheme [Reg05b].

Construction 2.4.4 (Regev Encryption Scheme)**.** *This encryption scheme operates on bits, i.e., the message space is* $\mathcal{M} = \{0,1\}$*. It is composed of the following algorithms:*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \Gamma)$ **:** *given the security parameter in unary, choose an integer* $n = n(\lambda)$*, a modulus* $q = q(\lambda)$*,* $0 < \alpha < 1$*, an error distribution* χ_σ *of parameter* $\sigma = \alpha q \geq 2\sqrt{n}$*, and output a secret key* $\mathsf{sk} \leftarrow\!\!\$\, \mathbb{Z}_q^n$ *and public parameters* $\Gamma = (n, q, \chi_\sigma)$*. In the following,* Γ *is an implicit argument to all the algorithms.*

$\mathsf{Enc}(\mathsf{sk}, m \in \mathcal{M}) \to c$ **:** *given the secret key* sk *and a message* m*, sample* $\mathbf{a} \leftarrow\!\!\$\, \mathbb{Z}_q^n$*,* $e \leftarrow \chi$*, and return* $c = (\mathbf{a}, b = \langle \mathsf{sk}, \mathbf{a} \rangle + e + m\,\frac{q}{2}) \in \mathbb{Z}_q^{n+1}$*.*

$\mathsf{Dec}(\mathsf{sk}, c) \to m'$ **:** *given the secret key* sk *and a ciphertext* $c = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$*, compute* $b - \langle \mathsf{sk}, \mathbf{a} \rangle$ *and return* $m' = 0$ *if this quantity is closer to* 0 *than to* $\frac{q}{2}$*. Otherwise return* $m' = 1$*.*

Correctness and security of Construction 2.4.4 are straightforward: correctness holds as long as $|e| < \frac{q}{4}$, whereas security comes directly from the LWE assumption. In particular, the term $\langle \mathsf{sk}, \mathbf{a} \rangle + e$ plays the role of a random mask for the message m.

Remark 2.4.5 (Extending the message space)**.** *Construction 2.4.4 can be trivially extended to* $\mathcal{M} = \mathbb{Z}_p$*, for* $q > p \in \mathcal{N}$*. It is sufficient to multiply* m *by* $\frac{q}{p}$ *instead of* $\frac{q}{2}$ *during encryption, and round appropriately during decryption. Naturally, this implies a different condition on the error for correctness to hold: in this case, it is necessary that* $|e| < \frac{q}{2p}$*.*

Parameters. Going through the details of the Regev cryptosystem and the extended encryption, it is evident that choosing parameters for lattice-based cryptosystems is no trivial matter. Several authors have studied the problem, e.g., [GN08, RS10], and we refer to these works for more details. Choosing parameters is a matter of trade-offs. Generally one would like to have as small parameters as possible, while still maintaining correctness and security. Assuming a fixed p, choosing a bigger q and α will result in harder underlying lattice problems, thereby increasing security. On the other hand, choosing a larger α makes decryption harder, and in turn one will have to choose a bigger q, harming efficiency.

2.4.3 Homomorphic Encryption Schemes

Some encryption schemes have the additional property of homomorphism, that allows one to perform operations on ciphertexts based solely on publicly available information, and in particular without having access to any secret key. For example, an *additively homomorphic* encryption scheme allows anyone to take a ciphertext c_1 encrypting a message m_1, a ciphertext c_2 encrypting a message m_2, and produce a ciphertext c_+ that decrypts to $m_1 + m_2$. Analogously, a *multiplicatively homomorphic* encryption scheme allows one to produce a ciphertext $c_\times$ that decrypts to $m_1 \cdot m_2$. And this is possible *without having access* to either m_1 or m_2 or any secret information.

Definition 2.4.6 (Homomorphic Encryption)**.** *A homomorphic (public-key) encryption scheme $H.\mathcal{E}nc$ with message space $\mathcal{M}$ is composed of the following algorithms:*

$H.\mathsf{KGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$ *: given the security parameter, output a secret key* sk*, a public key* pk *and an evaluation public key* evk*.*

$H.\mathsf{Enc}(\mathsf{sk}, \mu) \to c$ *: given the public key* pk *and a message* $\mu \in \mathcal{M}$*, output a ciphertext* c*.*

$H.\mathsf{Dec}(\mathsf{sk}, c) \to \mu$ *: given the secret key* sk *and a ciphertext* c*, output a message* $\mu \in \mathcal{M}$*.*

$H.\mathsf{Eval}(\mathsf{evk}, f, c_1, \ldots, c_\ell)$ *: given the public key* evk*, a function* $f : \mathcal{M}^\ell \to \mathcal{M}$ *and ciphertexts* $c_1, \ldots, c_\ell$*, apply the function* f *on ciphertexts* c_i *and output a ciphertext* c_f*.*

We now give a fundamental definition for any homomorphic encryption scheme, i.e., evaluation correctness.

Evaluation Correctness. We say that the $H.\mathsf{Eval}$ algorithm correctly evaluates all functions in $\mathcal{F}$ if, for any function $f \in \mathcal{F} : \mathcal{M}^\ell \to \mathcal{M}$ and respective inputs $x_1, \ldots, x_\ell$, it holds that

$$\Pr[H.\mathsf{Dec}(\mathsf{sk}, H.\mathsf{Eval}(\mathsf{evk}, f, c_1, \ldots, c_\ell)) \neq f(x_1, \ldots, x_\ell)] = \mathsf{negl},$$

where $\mathsf{sk} \leftarrow H.\mathsf{KGen}(1^\lambda)$, and $c_i \leftarrow H.\mathsf{Enc}(\mathsf{pk}, x_i), \forall i \in [\ell]$.

Somewhat and Fully Homomorphic Encryption. In most of the homomorphic encryption schemes known to date, an error term is injected during the encryption procedure for security purposes. The reason is that these encryption schemes rely on the hardness of solving "noisy" problems, i.e., problems where the relations are not exact, but are perturbed by a moderate quantity of error as dLWE mentioned in Assumption 2.3.10. Combining multiple ciphertexts through homomorphic operations has the side effect of combining the noises as well, thus increasing the magnitude of the error in the resulting encryption. When the error grows beyond a certain threshold, correctness is lost, meaning that the decryption procedure will not return the expected result. We say that an encryption scheme is *somewhat* homomorphic if it can evaluate a certain number of homomorphic operations, before the error grows too much to maintain the correctness of the evaluation.

On the other hand, a *fully homomorphic encryption* (FHE) scheme is a homomorphic scheme that allows for the evaluation of arbitrarily complex computations over encrypted data. The problem of designing such scheme was suggested by Rivest, Adleman and Dertouzos in 1978 [RAD78] but, despite moderate progress [GM82, Pai99, BGN05, IP07], it remained the "Holy Grail of cryptography" ([Mic10]) until the breakthrough result of Gentry in 2009 [Gen09]. In the case of FHE, there is no need to set an a priori bound on the number of homomorphic operations, thus making the scheme more flexible. In contrast, FHE schemes tend to be considerably less efficient than their *leveled* relaxation[1], which, for specific applications, can be noticeably faster and, as a consequence, more appealing.

Over the years, homomorphic encryption has been a very active field for research, and this has led to a long list of works and improvements, (e.g., [vDGHV10, SS10, SV10, BV11a, BV11b, BGV12, GHS12, GSW13, BV14, AP14]). This comprised both theoretical works that put forth new ideas and constructions, and implementation works that optimized existing constructions with the goal of achieving the best possible efficiency.

The FHE constructions that we know of can roughly be divided into three groups, usually referred to as *generations*. "first generation" FHE usually denotes the one stemming directly

[1] In a "leveled" FHE scheme, the parameters of the scheme may depend on the depth of the circuits that the scheme can evaluate (but not on their size).

from Gentry's seminal work [Gen09] and based on ideal lattices and the approximate GCD problem; "second generation" usually indicates constructions proposed in a sequence of works by Brakerski and Vaikuntanathan [BV11a, BV11b] and based on the LWE problem; "third generation" usually denotes the GSW FHE scheme by Gentry, Sahai, and Waters [GSW13] and subsequent works (e.g., [AP14, HAO15]).

2.4.4 Digital Signatures

Digital Signatures enable the holder of a secret key to sign messages in such a way that anyone in possession of the corresponding public verification key can determine the validity of a given message-signature pair. For security, it is required that the signature is unforgeable, i.e., no efficient adversary can forge a valid signature (unless the adversary knows the secret key).

Definition 2.4.7 (Digital Signature)**.** *A digital signature scheme* Σ *consists of a triple of algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *working as follows:*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$ *the key generation takes as input the security parameter* λ *and returns a pair of keys* $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$ *on input a signing key* sk *and a message* m*, the signing algorithm produces a signature* σ.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to 0/1$ *given a triple* vk, m, σ *the verification algorithm tests if* σ *is a valid signature on* m *with respect to verification key* vk.

The standard security notion for digital signatures, *unforgeability* against chosen-message attacks (UF-CMA, for short) is defined by the experiment in Figure 2.15 where the adversary (the forger) $\mathcal{F}$ has access to the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

$\underline{\mathsf{UF\text{-}CMA}_{\Sigma,\mathcal{F}}(1^\lambda)}$

$(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$

$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{F}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})$

return $(\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*))$

$\wedge\ (m^*$ is "new")

Figure 2.15: Experiment for UF-CMA security notion.

A message m^* is said "new" if it is different from all the messages m_i that the adversary queried to the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ during the experiment.

Definition 2.4.8 (UF-CMA security)**.** *A digital signature scheme* Σ *is* UF-CMA*-secure if for any forger* $\mathcal{F}$ *running in time* $t = \mathsf{poly}(\lambda)$ *and making* $Q = \mathsf{poly}(\lambda)$ *signing queries, the following probability is negligible:*

$$\mathsf{Adv}^{\mathsf{uf\text{-}cma}}_{\Sigma,\mathcal{F}}(\lambda) = \Pr[\mathsf{UF\text{-}CMA}_{\Sigma,\mathcal{F}}(1^\lambda)] = \mathsf{negl}.$$

We say that an adversary $\mathcal{F}$ (ε, t, Q)-breaks the unforgeability of Σ if $\mathsf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathcal{F},\Sigma}(\lambda) \geq \varepsilon$ holds for an $\mathcal{F}$ running in time t and making Q queries.

A stronger notion of security is *strong unforgeability* against chosen-message attacks (SUF-CMA). This notion is defined by considering a security experiment slightly different than $\mathsf{UF\text{-}CMA}_{\mathcal{F},\Sigma}(1^\lambda)$. Instead of checking whether m^* is "new", one checks whether the pair (m^*, σ^*) is "new", i.e., if (m^*, σ^*) is different from all the pairs (m_i, σ_i) obtained by $\mathcal{F}$ from the signing oracle.

Homomorphic Signatures. The notion of homomorphic signatures was formalized by [BFKW08] and long studied since then.

A homomorphic signature scheme (HS) enable the holder of a secret key to sign messages $m_1, \dots m_n$ in such a way that anyone in possession of the corresponding signatures $\sigma_1, \dots \sigma_n$ and a function f can produce a valid signature σ for the message $f(m_1, \dots m_n)$. The key property of HS is succinctness: the size of the evaluated signature σ should be smaller than the concatenation of the initial signatures $\{\sigma_i\}_{i\in[n]}$ (and it is usually logarithmic in n, the number of messages). In homomorphic settings the definition of unforgeability depends on the class of functions f supported by the scheme. For schemes that support only linear functions on a vector space, e.g., [BF11b], unforgeability states that the adversary should not be able to derive a correct signature for a message (vector) which cannot be obtained as a linear combination of previously honestly signed messages. If we applied the same reasoning to linearly homomorphic signatures with messages in a field or to Fully Homomorphic Signature schemes (FHS), e.g., [BF11a] we would end up with a useless notion of security: because it is always possible to generate a valid signature for any message derived as a function from the initial set of $\{m_i\}_{i\in[n]}$, meaning that one can compute signatures for any message in the whole message space. A meaningful notion of unforgeability for FHS requires that the adversary should not be able to derive a valid signature σ^* for a value y^* that is not the correct output of $f(m_1, \dots m_n)$. This notion is achieved thanks to labelled programs [GW13], as in FHS the signatures, the homomorphically evaluated signatures and the verification procedure all depend on the labels. The unforgeability intuitions given in this section are approximations of the core meaning of the corresponding security notions that are detailed in Section 6.2.1 of Chapter 6.

2.4.5 Commitment Schemes

The notion of commitment is one of the most fundamental and widely used in cryptography. First, a non-interactive commitment scheme allows a sender to create a commitment to a secret value. It may later open the commitment and reveal the value in a verifiable manner. A commitment should be hiding and binding in the sense that a commitment does not reveal the secret value and cannot be opened to two different values:

Definition 2.4.9 (Non-Interactive Commitment)**.** *A non-interactive commitment scheme is a tuple of algorithms* $\mathcal{C}om = (\mathsf{ComGen}, \mathsf{Com}, \mathsf{ComVer}, \mathsf{OpenVer})$*:*

- $\mathsf{ComGen}(1^\lambda) \to \mathsf{ck}$*: Generates a commitment public key* ck*. It specifies a message space* M_{ck}*, a randomness space* R_{ck}*, and a commitment space* C_{ck}*. This algorithm is run by a trusted or distributed authority;*
- $\mathsf{Com}(\mathsf{ck}, m) \to (c, o)$*: Outputs a commitment* c *and an opening information* o*. This algorithm specifies a function* $\mathsf{Com}_{\mathsf{ck}} : M_{\mathsf{ck}} \times \mathcal{R}_{\mathsf{ck}} \to C_{\mathsf{ck}}$*. Given a message* $m \in M_{\mathsf{ck}}$*, the sender picks a randomness* $\rho \in R_{\mathsf{ck}}$ *and computes the commitment* $(c, o) = \mathsf{Com}_{\mathsf{ck}}(m, \rho)$*.*

$\mathsf{ComVer}(\mathsf{ck}, c) \to 0/1$: *Checks whether c is a well-formed commitment. If so, it outputs 1, otherwise it outputs 0;*

$\mathsf{OpenVer}(\mathsf{ck}, c, m, o) \to 0/1$: *Outputs 1 if the value $m \in M_{\mathsf{ck}}$ is the committed message in the commitment c and 0 if (m, o, c) does not correspond to a valid pair opening-commitment.*

We say $\mathcal{C}om = (\mathsf{ComGen}, \mathsf{Com}, \mathsf{ComVer}, \mathsf{OpenVer})$ is a secure commitment scheme if it satisfies the following properties:

Correctness. *Let $\mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda)$. Any commitment of $m \in M_{\mathsf{ck}}$ honestly generated $(c, o) \leftarrow \mathsf{Com}(\mathsf{ck}, m)$ is successfully verified by $\mathsf{ComVer}(\mathsf{ck}, c)$ and by $\mathsf{OpenVer}(\mathsf{ck}, c, m, o)$.*

Hiding. *It is statistically hard, for any adversary $\mathcal{A}$, to generate two messages $m_0, m_1 \in M_{\mathsf{ck}}$ such that $\mathcal{A}$ can distinguish between their corresponding commitments c_0 and c_1 where $(c_0, o_0) \leftarrow \mathsf{Com}(\mathsf{ck}, m_0)$ and $(c_1, o_1) \leftarrow \mathsf{Com}(\mathsf{ck}, m_1)$. Meaning that the advantage of the adversary $\mathcal{A}$ on winning the game depicted in Figure 2.16 is negligible:*

$$\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{C}om,\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Hiding}_{\mathcal{C}om,\mathcal{A}}(1^\lambda)\right] = \mathsf{negl}.$$

$\underline{\mathsf{Hiding}_{\mathcal{C}om,\mathcal{A}}(1^\lambda)}$

$\mathsf{ck} \leftarrow \mathsf{ComGen}\left(1^\lambda\right)$
$(m_0, m_1) \leftarrow \mathcal{A}\left(\mathsf{ck}\right)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(c_b, o_b) \leftarrow \mathsf{Com}(\mathsf{ck}, m_b)$
$b' \leftarrow \mathcal{A}\left(\mathsf{ck}, c_b\right)$
return $b' = b$

$\underline{\mathsf{Binding}_{\mathcal{C}om,\mathcal{A}}(1^\lambda)}$

$\mathsf{ck} \leftarrow \mathsf{ComGen}\left(1^\lambda\right)$
$(c, (m_0, o_0), (m_1, o_1)) \leftarrow \mathcal{A}(\mathsf{ck})$
return $\big((m_0 \neq m_1) \wedge$
$(\mathsf{OpenVer}(\mathsf{ck}, c, m_0, o_0) = 1) \wedge$
$(\mathsf{OpenVer}(\mathsf{ck}, c, m_1, o_1) = 1)\big)$

Figure 2.16: Experiments for Hiding and Binding.

Binding. *It is computationally hard, for any adversary $\mathcal{A}$, to come up with a collision (c, m_0, o_0, m_1, o_1), such that o_0 and o_1 are valid opening values for two different pre-images $m_0 \neq m_1$ for c. For any adversary $\mathcal{A}$, the following probability is negligible*

$$\mathsf{Adv}^{\mathsf{binding}}_{\mathcal{C}om,\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Binding}_{\mathcal{C}om,\mathcal{A}}(1^\lambda)\right] = \mathsf{negl}.$$

The game for the binding property is depicted in Figure 2.16.

Knowledge Binding. *For every adversary $\mathcal{A}$ that produces a valid commitment c associated to a message that verifies, i.e. such that $\mathsf{ComVer}(\mathsf{ck}, c) = 1$, there is an extractor $\mathcal{E}_{\mathcal{A}}$ that is able to output a pre-image m and a valid opening o of c, with overwhelming probability:*

$$\Pr\left[\begin{array}{l|r} \mathsf{OpenVer}(\mathsf{ck}, c, m, o) = 1 & \mathsf{ck} \leftarrow \mathsf{ComGen}(1^\lambda) \\ & (c; (m, o)) \leftarrow (\mathcal{A} \| \mathcal{E}_{\mathcal{A}})(\mathsf{ck}) \\ & \mathsf{ComVer}(\mathsf{ck}, c) = 1 \end{array}\right] = 1 - \mathsf{negl}.$$

For the sake of simplicity, throughout this work, we will omit the commitment key ck from the input of the algorithms, and with a slight abuse of notations, we will adopt the writing $\mathsf{Com}(m) \to (c, o)$.

Homomorphic Commitment Scheme. A commitment scheme can also be homomorphic, if for a group law $\oplus$ on the message space M_{ck}, from $(m_0, o_0) \leftarrow \mathsf{Com}(\mathsf{ck}, m_0)$ and $(m_1, o_1) \leftarrow \mathsf{Com}(\mathsf{ck}, m_1)$, one can efficiently generate c from c_0, c_1 as well as o from o_0 and o_1 so that $\mathsf{OpenVer}(\mathsf{ck}, c, o, m_0 \oplus m_1) = 1$.

An Example: 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:

Construction 2.4.10 (Pedersen Commitment)**.** *The Pedersen commitment scheme is defined as follows:*

$\mathsf{ComGen}(1^\lambda) \to \mathsf{ck}$*: Generates the description of a group* $\mathbb{G}$ *of prime order* p *together with two generators* $(g, h) \leftarrow_\$ \mathbb{G}^2$*. We let* $\mathsf{ck} \coloneqq (\mathbb{G}, p, g, h)$*;*

$\mathsf{Com}(\mathsf{ck}, m) \to (c, o)$*: Given* ck *and the message* $m \in \mathbb{Z}_p$ *it samples a randomness* $r \leftarrow_\$ \mathbb{Z}_p$ *and computes the commitment* $(c, o) \coloneqq (g^m h^r, r)$*.*

$\mathsf{ComVer}(\mathsf{ck}, c) \to 0/1$*: Checks whether* $c \in \mathbb{G}$*;*

$\mathsf{OpenVer}(\mathsf{ck}, c, m, o) \to 0/1$*: Outputs* 1 *if* $g^m h^r = c$*.*

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 $m \in \mathbb{Z}_p$, $c = g^m h^r$ 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 $h = g^{(m_0 - m_1)/(r_1 - r_0)}$, 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. □

Chapter 3

Succinct Non-Interactive Arguments of Knowledge

Survey of Proof Systems. In this chapter, we provide the reader with an overview of the context of our work and we introduce Succinct Non-interactive Arguments of Knowledge (SNARK), the main object of study in this thesis. We recall the history of proof systems in cryptography; we try to give an idea of their importance, the evolution of the soundness notion, the way SNARKs burst into cryptography and some well-known constructions.

The first part of the chapter is rather informal; it gives some intuition and the historical evolution of (zero-knowledge) proofs and arguments. A more detailed study of these notions that are tangential with the SNARK advancements can be read in some recent PhD thesis, such as the ones by Geoffroy Couteau [Cou18] and Fabrice Ben Hamouda [Ben16] and in the survey [Gol08] that I used for inspiration. This chapter can be seen as a (non-comprehensive) introduction to SNARKs, that recalls a variety of outstanding results, together with historical insights and concrete examples. The idea is for this chapter to be of independent interest as a brief survey on zero-knowledge proofs and SNARKs.

Contents

Tale one: Down the Rabbit-Hole

Back in the days when everyone was addicted to technology and connected to the internet, always checking their smartphones, downloading new apps, and constantly posting on social media... back in these times lived Alice, a little girl who had a simple life, going to school as every child of her age.
Day after day, Alice, gets more and more immersed in her technological gizmos and gadgets. These technologies made it possible for Alice to discover many functionalities and applications, to stay in touch with her friends and to find new hobbies.
Nevertheless, Alice still has a dilemma: She thinks that she spends too much time on doing her boring homework, she has to solve tons of redundant calculations and exercises on math, physics and other subjects. She considers this useless, even more in this modern world, where we can find online a variety of learning materials so much more exciting than old-school lecture books. She would like to find a way to overcome the school system and become a self-taught girl.
In this world where Alice is so connected, there may exist a solution to that! Searching over the Internet, she found the website of Oscar, a freelance who made some application that promises incredible functionalities: solving any school exercise, being math, chemistry, physics or any other science subject. This is exactly what she wanted! Alice starts daydreaming of herself being able to freely enjoy her passions while her homework is done without any effort.
She only has to install the app, give her homework as input, and then just enjoy her spare-time, surfing the Internet, watching movies, dancing, playing with her friends, reading her favorite books, writing stories, instead of doing her boring science exercises. But wait! That sounds too good to be true... So Alice, a foresighted and smart girl, has a hard time to fully trust this ”magical” service proposed by Oscar. Of course, she would like to try it, but letting someone solve complicated exercises for her and just receiving some answers without any guarantees of correctness seems too risky.
She needs to find a way to check that the received solutions are the correct ones. Of course she does not consider to try solving her homework again by herself in order to compare and check that the application was accurate. She needs something faster than that and also convincing.
Something like a proof from Oscar that the computation was performed as expected, a proof that Alice can verify fast and efficiently.
One way of doing so is by asking the solver specific questions, on how he achieved the solution, to detect that the process was correct. However, this seems an overhead for Alice, who does not want to spend her time to communicate with Oscar about this service, she wants just to receive a short and fast to check proof together with the solved homework. Also, from the point of view of Oscar, as a service provider, he has no interest to reveal to Alice his specific methods and tricks... He also fears that a competitor may steal his idea, so he may want to keep the details of the computation private.
She needs a fast solution, not requiring interaction and coming in a concise format. Would she be able to find such a proof? Through this chapter, Alice will get some possible answers to her problem, let’s see if any of them is good enough for her needs.

3.1 Introduction to Proofs and Arguments.

In modern cryptography, proofs play an essential dual role. Not only are they at the heart of provable security, but they are also both an important subject of study of cryptography and an important tool to construct advanced cryptosystems or cryptographic protocols. Proof systems introduced by [GMR89] are fundamental building blocks in cryptography. Extensively studied aspects of proof systems are the expressivity of provable statements and their efficiency.

Complexity Classes. In order to better understand the role of proofs and their classification, we will briefly and informally introduce some basic complexity notions. The complexity classes will be defined by the type of computational problem, the model of computation, and the resource that are being bounded and the bounds. The resource and bounds are usually stated together, such as "polynomial time", "logarithmic space", "constant depth", etc. We will introduce the main two fundamental complexity classes, P and NP. They are used to classify decision problems.

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

On the other hand, the class NP is the class of languages $\mathcal{L}$, such that there exists an algorithm, that takes as input two bit strings x and w and that can decide in polynomial time (in the size of x), whether w is a valid proof or witness that $x \in \mathcal{L}$. We suppose that for any statement $x \in \mathcal{L}$, there exists such a witness w, while otherwise ($x \notin \mathcal{L}$) no such witness exists. A formal definition is stated as follows:

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

$$\mathcal{L} = \{x | \exists\ w, |w| = \mathsf{poly}(|x|) \wedge \mathcal{R}_{\mathcal{L}}(x, w) = 1\}.$$

By restricting the definition of NP to witness strings of length zero, we capture the same problems as those in P. While the class P is clearly included in NP, finding whether NP is included in P is one of the most important open problems in computer science.

It basically asks whether being able to efficiently check a proof of a statement, is equivalent to being able to check if a statement is true or false efficiently. Even if we don't have any clear evidence for that, most researchers strongly believe that $\mathsf{P} \neq \mathsf{NP}$.

In cryptography, considerable atention is given to the NP-hard complexity class. NP-hard is the defining property of a class of problems that are, informally, "at least as hard as the hardest problems in NP".

We will often talk about NP-complete decision problems, the ones belonging to both the NP and the NP-hard complexity classes.

Example: Satisfiability Problems SAT**.** As an example for a problem in NP, let us consider the problem of boolean formula satisfiability (SAT). For that, we define a boolean formula using an inductive definition:

- any variable $x_1, x_2, x_3, \ldots$ is a boolean formula
- if f is a boolean formula, then $\neg f$ is a boolean formula (negation)
- if f and g are boolean formulas, then $(f \wedge g)$ and $(f \vee g)$ are boolean formulas (conjunction / and, disjunction / or).

The string $((x_1 \wedge x_2) \wedge \neg x_2)$ would be a boolean formula.

A boolean formula is satisfiable if there is a way to assign truth values to the variables so that the formula evaluates to true. The satisfiability problem SAT is the set of all satisfiable boolean formulas: $\mathsf{SAT}(f) := 1$ if f is a satisfiable boolean formula and 0 otherwise.

The example above, $((x_1 \wedge x_2) \wedge \neg x_2)$, is not satisfiable and thus does not lie in SAT. The witness for a given formula is its satisfying assignment and verifying that a variable assignment is satisfying is a task that can be solved in polynomial time.

The attractive property of this seemingly simple problem is that it does not only lie in NP, it is also NP-complete. It means that it is one of the hardest problems in NP, but more importantly – and that is the definition of NP-complete – an input to any problem in NP can be transformed to an equivalent input for SAT in the following sense:

For any NP-problem $\mathcal{L}$ there is a so-called reduction function f, which is computable in polynomial time such that:

$$\mathcal{L}(x) = \mathsf{SAT}(f(x)).$$

Such a reduction function can be seen as a compiler: It takes source code written in some programming language and transforms it into an equivalent program in another programming language, which typically is a machine language, which has the same semantic behaviour. Since SAT is NP-complete, such a reduction exists for any possible problem in NP.

Computational problems inside NP can be reduced to each other and, moreover, there are NP-complete problems that are basically only reformulations of all other problems in NP.

In the following section, we will discuss some other characterisations of NP class, such as Probabilistically Checkable Proofs (PCP) or (Boolean or Arithmetical) Circuit Satisfiability (Circuit-SAT).

3.1.1 Interactive Proofs

By Definition 3.1.1, the class NP contains all languages for which an unbounded prover can compute deterministic proofs, where a proof is viewed as a string of length polynomial in the statement x. An interactive proof relaxes these requirements in two directions: first, the parties are allowed to use random coins, second, the output of a proof verification should only match the actual truth of the statement with some reasonable enough probability and obviously, there is interaction between parties.

First Interactive Proofs. In two independent seminal papers, that won a Gödel prize, Babai [Bab85] and Goldwasser, Micali, and Rackoff [GMR85] introduced the notion of interactive proofs also known as *Arthur-Merlin proofs*.

Both works 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 these papers is regarding 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.

Zero-Knowledge. As pointed out by Goldwasser, Micali, and Rackoff in their seminal paper [GMR85], an essential question about interactive proofs in cryptography is whether the prover reveals more information (or knowledge) to the verifier than the fact that $x \in \mathcal{L}$. Indeed, in cryptography, we often want to hide information. A proof that does not reveal any information to the verifier besides the membership of the statement to the language is called a zero-knowledge proof. A way to formally define this property is to consider a simulator that is able to behave exactly as the prover in the protocol and to produce a "fake" proof without knowing the witness. This should be done in a way that a verifier will not be able to tell if it interacts with the real prover or with this simulator. Intuitively, we can then argue that a honestly generated proof looks indistinguishable from a simulated value produced independently of the witness, meaning that the proof reveals as much information about the witness as this value, so basically zero-knowledge.

This concept might seem very counter-intuitive and impossible to achieve. However, in [GMW86], Goldreich, Micali, and Wigderson constructed zero-knowledge proofs for any language in NP, under a very weak assumption, namely the existence of one-way functions.

Succinct Arguments. Related to efficiency and to optimization of communication complexity, it has been shown that statistically-sound proof systems are unlikely to allow for significant improvements in communication [BHZ87, GH98, GVW02, Wee05]. When considering proof systems for NP this means that, unless some complexity-theoretic collapses occur, in a statistically sound proof system any prover has to communicate, roughly, as much information as the size of the NP witness. The search for ways to beat this bound motivated the study of *computationally-sound* proof systems, also called *argument systems* [BCC88], where soundness is required to hold only against *computationally bounded* provers. Assuming the existence of collision-resistant hash functions, Kilian [Kil92] showed a four-message interactive argument for NP. In this protocol, membership of an instance x in an NP language with NP machine M can be proven with communication and verifier's running time bounded by $p(\lambda, |M|, |x|, \log t)$, where λ is a security parameter, t is the NP verification time of machine M for the instance x, and p is an *universal* polynomial. Argument systems of this kind are called *succinct*.

Zero-Knowledge Proofs and Arguments. A zero-knowledge proof or its relaxed version, argument, is a protocol between a prover P and a verifier V for proving that a statement x is in a language $\mathcal{L}$. Informally, such a protocol has to satisfy three properties:

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

Soundness. No unbounded/PPT adversary can make an honest verifier accept a proof of a word $x \in \mathcal{L}$ either statistically (for zero-knowledge proofs)/computationally (for zero-knowledge arguments).

Zero-knowledge It is possible to simulate (in polynomial-time) the interaction between a (potentially malicious) verifier and an honest prover for any word $x \in \mathcal{L}$ without knowing a witness w.

Honest-Verifier Zero-Knowledge. Honest-verifier zero-knowledge arguments or proofs are similar to the ones defined above, except that we assume that the verifier is not malicious. The zero-knowledge property applies only to verifiers that behave honestly and follow the protocol. This relaxation enables to construct even more efficient schemes.

3.1.2 Interactive Arguments of Knowledge

The proofs and arguments we discussed in the previous section are tools used for membership statements, *i.e.*, proving membership of an instance x in a language $\mathcal{L}$. Restricting our attention to NP-languages, such statements can be phrased as existential statements, of the form $\exists w, \mathcal{R}_{\mathcal{L}}(x, w) = 1$. 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 shows that the prover knows such a witness.

Several remarks are in order here. First, we have to define what it means for a prover to know such a witness. Intuitively, to make sure a prover has used the witness, it should be possible to "extract" this knowledge from that prover. Informally, this is done as follows: we say that an (efficient) algorithm $\mathcal{A}$ knows a value w if we can build a simulator Sim that, for any such $\mathcal{A}$ that produces an accepting transcript, Sim can extract the witness w from its interaction with $\mathcal{A}$.

Second, 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 3.1.2 (Discrete Logarithm Language)**.** *Let $\mathcal{L}_{\mathsf{DLog}}(\mathbb{G}, g)$ denote, for a cyclic group $(\mathbb{G}, \cdot)$ with a generator g, the following language:*

$$\mathcal{L}_{\mathsf{DLog}}(\mathbb{G}, g) = \{h \in \mathbb{G} | \exists\ x \in \mathbb{Z}, g^x = h\}\,.$$

As g is a generator of $\mathbb{G}$, this is a trivial language: all elements of $\mathbb{G}$ belong to $\mathcal{L}_{\mathsf{DLog}}$, $\forall\ h \in \mathbb{G}, \exists\ x \in \mathbb{Z}$ such that $g^x = h$, and this exponent x is not unique. However, computing such an integer x can be computationally infeasible (see the discussion on the discrete logarithm assumption, Section 2.3.1). 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 piece of non-trivial information.

Sigma-Protocols. In this part we will describe a specific class of zero-knowledge proof systems to which very efficient zero-knowledge protocols from the literature belong: Σ-protocols [CDS94].

Definition 3.1.3 (Sigma-Protocol)**.** *A Σ-protocol for a language $\mathcal{L}$ is a public-coin honest-verifier zero-knowledge proof of knowledge, with a particular three-move structure:*

Commit Phase. *P sends to V some commitment values to some randomness,*
Challenge Phase. *V sends to P a uniformly random challenge e,*
Response Phase. *P sends to V an answer $f(w, r, e)$ where f is some public function, and w is the witness held by P*

Example: The Schnorr Protocol. In Example 3.1.2, we were 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 in Figure 3.1. It was first described in [Sch90], and it is commonly used as an authentication protocol: given a public value h, the prover authenticates 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).

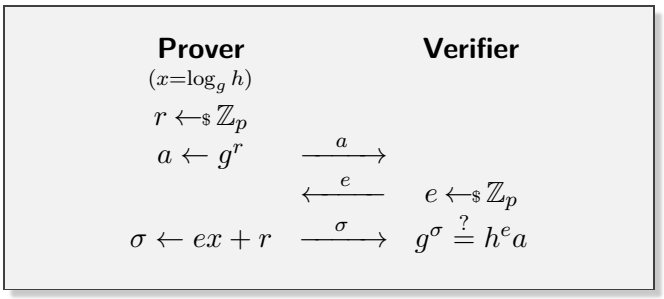


Figure 3.1: Schnorr Σ-Protocol for DLog Language.

Rewinding. The standard solution to prove the 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. By doing so, the simulator 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 its witness.

Security Analysis (Sketch). We show that the protocol given in figure Figure 3.1 is perfectly complete, knowledge-extractable, and honest-verifier zero-knowledge.

Perfect completeness. It follows immediately by inspection: if $\sigma = ex + r \mod p$, then $g^\sigma = g^{ex+r} = (g^x)^e g^r = h^e a$.

Honest-verifier zero-knowledge. Let Sim be a simulator which is given the common input $(\mathbb{G}, g, h)$ and the code of the verifier. Sim selects a uniformly random tape for the verifier algorithm and runs it with this random tape on a random input message $a \in \mathbb{G}$. Once the Verifier outputs a challenge e, Sim restarts the protocol, feeding Verifier algorithm with the same random tape and setting the input message a to $g^r h^{-e}$ for a uniformly random r. Note that a is distributed exactly as in an honest execution of the protocol. After the verifier outputs the challenge e (the verifier is assumed honest, so it uses only the coins of his random tape. Hence, this challenge is the same than the one extracted by Sim in the previous run of the verifier), Sim answers with $\sigma := r$. Observe that the equation $g^\sigma = h^e a$ is satisfied for the chosen values of σ and a, and that the answer is distributed exactly as in an honest run, hence the honest-verifier zero-knowledge property.

Knowledge-extraction. Consider a prover that runs in time T and produces an accepting answer with non-negligible probability ε, and let Sim' be a simulator which is given the code of the prover as input. Once the prover outputs the first flow a, Sim' writes a

random $e \in \mathbb{Z}_p$ on its message input tape, and gets an answer σ. Then, Sim' rewinds the prover to step 2 of the protocol, feeding it with a new random e' and receiving a corresponding new answer σ'. Observe that if both (σ, σ') are accepting answers, it holds that $g^{\sigma} = h^{e}a$, $g^{\sigma'} = h^{e'}a$, which gives $g^{\sigma-\sigma'} = h^{e-e'} = (g^{x})^{e-e'}$. In this case, Sim' can obtain x by computing $(\sigma - \sigma')(e - e')^{-1} \pmod{p}$ (as we have $e \neq e'$ with overwhelming probability). We argue the simulator Sim' for a prover that runs in time T and has success probability ε runs in $\mathcal{O}(T/\varepsilon)$ (the simulator repeats the rewinding procedure at most $1/\varepsilon$ times).

3.1.3 Non-Interactive Proofs and 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. In this section, we will focus on protocols that do not require more communication, than a sole message from prover to verifier. In a non-interactive proof or argument, the prover just sends one message (called the proof) to the verifier, and the latter can check it in order to accept it or not. This proof is similar to a witness of an NP language, except that sending a witness often gives too much knowledge to the verifier.

Non-Interactive Zero-Knowledge. Zero-knowledge proofs are randomized interactive proof systems satisfying a specific zero-knowledge property. All the results mentioned previously relied on interactive protocols with strong security guarantees without making any trust assumption whatsoever. This is known as the *standard model* (see Section 2.2.2), 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 proved impossible to achievable in the standard model. Consider 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 standard 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].

A natural theoretical question is to ask whether there are zero-knowledge randomized proofs that are completely non-interactive (no round of interaction is needed). 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. Other motivations for NIZK proofs are their applications to numerous cryptographic primitives.

Common Reference String. In light of the strong limitations discussed above, an interesting research direction is to find the minimal trust assumptions one could make that lead to a model in which practically efficient NIZK proof systems can be built. Some impossibility results and studies of lower-bounds were shown for various models, we refer to [Wee07] for more details. The common reference string model (CRS) described in Section 2.2.2 has proven very convenient to use for constructing a large variety of efficient primitives with strong security requirements. In this model, the prover and the verifier both have access to a common bit string chosen by some trusted party. In practice, such a bit string can be generated by a multi-party computation between users who are believed not to collude.

First NIZK Schemes. Blum et al. first study the non-interactive zero-knowledge proof system and present the common reference string model that is generally applied at present [BFM88, DMP90]. This first construction of [BFM88] is a bounded NIZK proof system, meaning that for different statements in NP language, the proof system has to use different CRSs and the length of the statement is controlled by the length of CRS.

Later, Blum et al. [DMP90] presented a more general (multi-theorem) NIZK proof system for 3SAT by improving the previous one, which allows to prove many statements with the same CRS.

Both [BFM88] and [DMP90] based their NIZK systems on certain number-theoretic assumptions (specifically, the hardness of deciding quadratic residues modulo a composite number). Feige, Lapidot, and Shamir [FLS90] showed later how to construct computational NIZK proofs based on any trapdoor permutation.

Much research has been devoted to the construction of efficient NIZK proofs [Dam93, KP98, BDP00], but back then, the only known method to do so has been the "hidden random bits" model. This hidden random bits model assumes the prover has a string of random bits, which are secret to the verifier. By revealing a subset of these bits, and keeping the rest secret, the prover can convince the verifier of the truth of the statement in question. Improvements in the efficiency of NIZK proofs have come in the form of various ways to set up a hidden random bits model and how to use it optimally.

Groth-Sahai Proofs. For a long time, two main 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 random bits model [FLS90, Dam93, KP98, BDP00], 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.

The Groth-Ostrovsky-Sahai proof system was the first perfect NIZK argument system for any NP language and the first universal composable secure NIZK argument for any NP language. This resolved a central open problem concerning NIZK protocols. The mechanism was dramatically different from the previous works, such as Blum-Feldman-Micali proof system [BFM88] and Blum-Santis-Micali-Persiano proof system [DMP90].

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 greatly improved the efficiency and practicability of NIZK and created a new line of research on the applications of NIZK.

Nevertheless, these schemes pose a limitation on the length of the proof statement in order to achieve *adaptive soundness* against dishonest provers who may choose the target statement depending on the CRS. Since the common reference string is public, it would be more natural to define soundness adaptively.

The first adaptively-sound statistical NIZK argument for NP that does not pose any restriction on the statements to be proven requires non-falsifiable assumptions (see [AF07]). Abe and Fehr [AF07] have demonstrated also an impossibility result: no adaptively-sound statistical zero-knowledge NIZK argument for an NP-complete language can have a "direct black-box" security reduction to a standard cryptographic assumption unless $\mathsf{NP} \subseteq \mathsf{P/poly}$.

Fiat-Shamir Heuristic. The Fiat-Shamir heuristic [FS87] is a heuristic method to convert Σ-protocols (see Section 3.1.2) into non-interactive zero-knowledge proofs. It proceeds as follows: to prove the membership of an instance x to a language $\mathcal{L}$ the prover first computes the first flow (the commitments) of a Σ-protocol for this statement. Let a denote this first flow. Then, the prover sets $e \leftarrow \mathsf{RO}(x, a)$, where RO is some hash function modeled by a random oracle, and computes the last flow (step 3 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 only in the random oracle model (see Section 2.2.2 for details). This model has its disadvantages, as it is seen more as heuristically secure since no trully random hash functions can be used in practice. Some failures of the random oracle methodology when implemented in practice are shown in [CGH98].

Still, an open question is whether there exist concrete hash families that are "Fiat-Shamir-compatible" (i.e., that can guarantee soundness and potentially also zero-knowledge for the transformed protocol). Initial results in this direction were negative. Indeed, Goldwasser and Kalai [GK03] (following Barak [Bar01]) demonstrated a three-round, public-coin argument scheme for which applying the Fiat-Shamir transform with any hash family never yields a sound protocol. Furthermore, Bitansky et al. [BDG+13] show that, even when starting with a three-round proof, soundness of the Fiat-Shamir transform with a concrete hash family cannot be proved via black-box reduction to standard, game-based assumptions. In contrast, a recent line of work [KRR17, CCRR18, HL18] circumvents the [BDG+13] impossibility result by using stronger than standard hardness assumptions to construct FS-compatible hash families. Kalai et al. [KRR17] gave the first construction of a hash family that is FS-compatible for arbitrary constant-round (public-coin) interactive proofs, albeit from complex obfuscation assumptions. Canetti et al. ([CCRR18]) then provide alternative families without obfuscation, but using complex KDM-security assumptions on secret-key encryption schemes. It is important to remark that the assumptions made by [KRR17, CCRR18] are non-falsifiable and highly complex in the following sense: both involve an adversary that is in part computationally unbounded.

In two recent companion articles, Canetti et al. [CCH+18, CLW18] construct explicit hash functions that are FS-compatible for a rich class of protocols, and they prove their security under assumptions that are qualitatively weaker than what was previously known. Using these hash families, new results can be obtained for delegation of computation and zero-knowledge.

SNARG: Succinct Non-Interactive Arguments. Starting from Kilian's protocol, Micali [Mic94] used the Fiat-Shamir heuristic to construct a *one-message* succinct argument for NP whose soundness is set in the random oracle model. New more efficient systems followed in the CRS model, they are called *succinct non-interactive arguments* (SNARGs) [GW11]. The area of SNARGs became quite popular in the last years with the proposal of several constructions in the CRS model, some of which gained significant improvements in efficiency [Gro10, Lip12, BCCT12, GGPR13, PHGR13, BCG+13, DFGK14, Gro16].

Non-Falsifiable Assumptions. Noteworthy is that all SNARG constructions are based on non-falsifiable assumptions [Nao03b], a class of assumptions that is likely to be inherent in proving the security of SNARGs (without random oracles), as stated by Gentry and Wichs in their work [GW11]. They show that no construction of SNARGs can be proven secure via a black-box reduction from any falsifiable assumption (unless that assumption is already false).

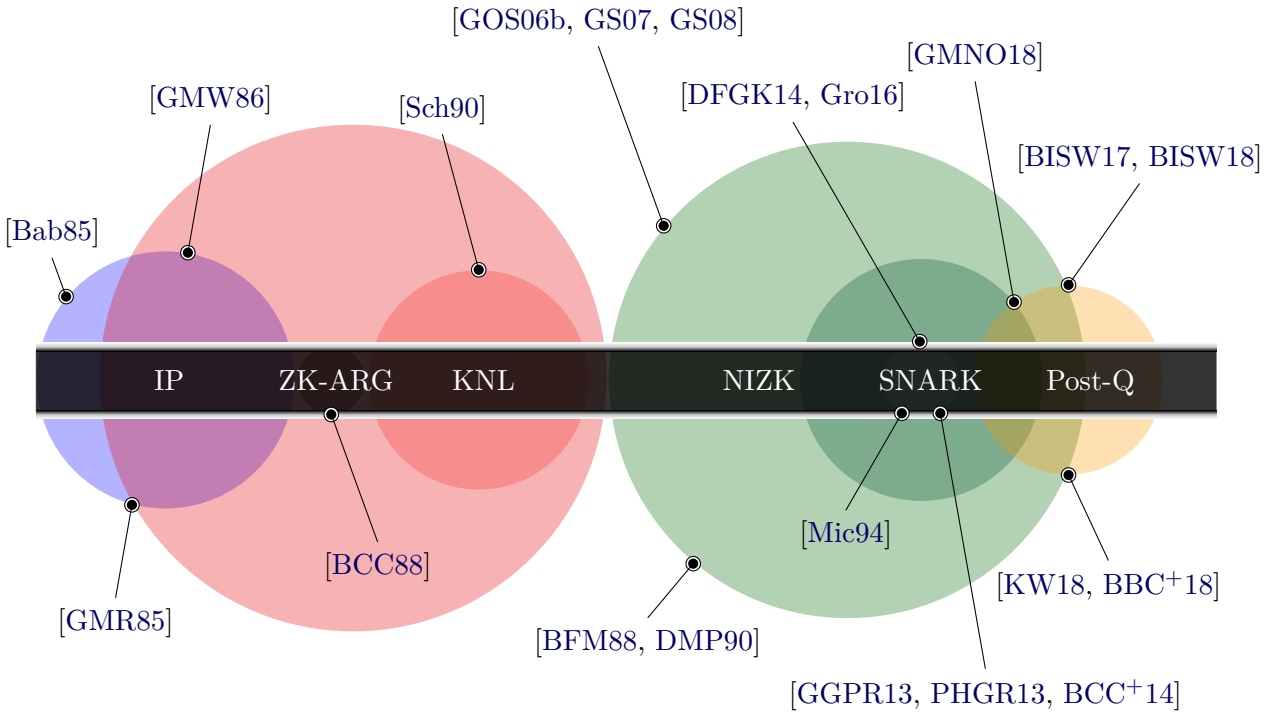


Figure 3.2: Proof systems. Some of the results mentioned in this chapter: Interactive schemes, starting with proofs (IP), zero-knowledge proofs and arguments (ZK-ARG), arguments of knowledge (KNL), then non-interactive zero-knowledge (NIZK), SNARKs and post-quantum schemes.

Knowledge Soundness. SNARGs have also been strengthened to become *succinct non-interactive arguments of knowledge* (SNARKs) [BCCT12, BCC+14]. SNARKs are SNARGs where computational soundness is replaced by *knowledge soundness*. Intuitively speaking, this property says that every prover producing a convincing proof must "know" a witness. On the one hand, knowledge soundness turns out to be useful in many applications, such as delegation of computation where the untrusted worker contributes its own input to the computation, or recursive proof composition [Val08, BCCT13].

On the other hand, the formalization of knowledge soundness in non-interactive protocols is a delicate point since rewinding techniques mentioned in Section 3.1.2 do not apply anymore. Typically, the concept that the prover "must know" a witness is expressed by assuming that such knowledge can be efficiently extracted from the prover by means of a so-called *knowledge extractor*. In SNARKs, extractors are inherently non-black-box, and the definition of knowledge soundness requires that for every adversarial prover $\mathcal{A}$ generating an accepting proof π there must be an extractor $\mathcal{E}_{\mathcal{A}}$ that, given non-black-box access to $\mathcal{A}$ (e.g., by getting the same input, including the random coins and the code of $\mathcal{A}$), outputs a valid witness.

Post-Quantum Proof Systems. Almost all the proof systems mentioned so far are based on discrete-log type assumptions, that do not hold against quantum polynomial-time adversaries [Sho99], hence the advent of general-purpose quantum computers would render insecure the constructions based on these assumptions. Efforts were made to design such systems based

on quantum resilient assumptions. We note that the original protocol of Micali [Mic94] is a zk-SNARK which can be instantiated with a post-quantum assumption since it requires only a collision-resistant hash function – however (even in the best optimized version recently proposed in [BSBHR18]) the protocol does not seem to scale well for even moderately complex computations.

Some more desirable assumptions that withstand quantum attacks are the aforementioned lattice assumptions [Ajt96, MR04]. Nevertheless, few non-interactive proof systems are built based on lattices. Some recent works that we can mention are the NIZK constructions for specific languages, like [KW18, LLNW18, BBC+18] and the two designated verifier SNARG constructions [BISW17, BISW18], designed by Boneh et al. using encryption schemes instantiated with lattices. We introduce in this thesis (in Chapter 4) the first lattice-based designated-verifier zk-SNARK [GMNO18] that assumes weaker properties on the encryption schemes than the non-interactive argument of Boneh et al. and additionally achieves knowledge-soundness.

3.2 SNARK: Definitions

The Universal Relation and NP Relations. A difficulty that arises when studying the efficiency of proofs for arbitrary NP statements is the problem of representation. Proof systems are typically designed for abstract NP-complete languages such as circuit satisfiability or algebraic constraint satisfaction problems, while in practice, many of the problem statements we are interested in proving are easier (and more efficient) to express via algorithms written in a high-level programming language. Modern compilers can efficiently transform these algorithms into a program to be executed on a random-access machine (RAM) [CR72, AV77]. Therefore, we choose to define SNARK protocols that efficiently support NP statements expressed as the correct execution of a RAM program.

We recall the notion of universal relation from [BG08], here adapted to the case of non-deterministic computations.

Definition 3.2.1. *The* universal relation *is the set $\mathcal{R}_\mathcal{U}$ of instance-witness pairs $(u, w) = ((M, x, t), w)$, where $|u|, |w| \leq t$ and M is a random-access machine such that $M(x, w)$ accepts after running at most t steps. The* universal language $\mathcal{L}_\mathcal{U}$ *is the language corresponding to $\mathcal{R}_\mathcal{U}$.*

For any constant $c \in \mathbb{N}$, $\mathcal{R}_c$ denotes the subset of $\mathcal{R}_\mathcal{U}$ of pairs $(u, w) = ((M, x, t), w)$ such that $t \leq |x|^c$. $\mathcal{R}_c$ is a "generalized" NP relation that is decidable in some fixed time polynomial in the size of the instance.

Universal Arguments vs. Weaker Notions. A SNARK for the relation $\mathcal{R} = \mathcal{R}_\mathcal{U}$ is called a universal argument. In the definition of $\mathcal{R}_\mathcal{U}$ we can replace the RAM machine M by a Turing machine, depending on the wished applications.

A weaker notion that we will also consider is a SNARK for the relation $\mathcal{R} = \mathcal{R}_c$ for a constant c. A SNARK can also be defined for a specific efficiently decidable binary relation $\mathcal{R}$, or for a boolean or arithmetic circuit $\mathtt{C}$. We will consider such definitions in some sections of this thesis, and we will mention if an adaptation for the specific settings is necessary.

Universal SNARK. We choose here to introduce the very general definition of universal SNARKs, only parametrized by a time bound T.

Definition 3.2.2 (SNARK for NP [BCC+14]). *A SNARK is defined by three algorithms:*

$\mathsf{Gen}(1^\lambda, T) \to (\mathsf{vrs}, \mathsf{crs})$**:** on input a security parameter $\lambda \in \mathbb{N}$ and a time bound $T \in \mathbb{N}$, this algorithm outputs a common reference string crs and a verification state vrs.;

$\mathsf{Prove}(\mathsf{crs}, u, w) \to \pi$**:** given a prover reference string crs, a statement u and a witness w s.t. $(u, w) \in \mathcal{R}$ and $t \leq T$, this algorithm produces a proof π;

$\mathsf{Ver}(\mathsf{vrs}, u, \pi) \to b$**:** on input a verification state vrs, an instance u, and a proof π, the verifier algorithm outputs $b = 0$ (reject) or $b = 1$ (accept);

satisfying *completeness*, *succinctness*, *knowledge-soundness* as described below:

$\mathsf{COMPL}_{\Pi,\mathcal{A}}$

$(\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{Gen}(1^\lambda, T)$
$(u, w) \leftarrow \mathcal{A}(\mathsf{crs})$
$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u, w)$
return $\big(\mathcal{R}(u, w) = 1 \ \wedge \ \mathsf{Ver}(\mathsf{vrs}, u, \pi) = 0\big)$

$\mathsf{KS}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}$

$(\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{Gen}(1^\lambda, T)$
$(u, \pi; w) \leftarrow (\mathcal{A} \| \mathcal{E}_\mathcal{A})(\mathsf{crs}, \mathsf{z})$
return $\big(\mathcal{R}(u, w) = 0 \ \wedge \ \mathsf{Ver}(\mathsf{vrs}, u, \pi) = 1\big)$

$\mathsf{ZK}_{\Pi,\mathsf{Sim},\mathcal{A}}$

$(\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{Gen}(1^\lambda, T)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{crs}}(1^\lambda, T)$
$b \leftarrow\!\!\$\ \{0, 1\}$
if $b = 1$ $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, u, w)$
if $b = 0$ $\pi \leftarrow \mathsf{Sim}_{\mathsf{proof}}(\mathsf{crs}, \mathsf{td}, u)$
$b' \leftarrow \mathcal{A}(\mathsf{vrs})$
return $(b = b')$

Figure 3.3: Games for completeness, knowledge soundness, and zero-knowledge.

- **Completeness.** For every time bound $T \in \mathbb{N}$, any relation $\mathcal{R}$ such that $t \leq T$ and any PPT adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{compl}}_{\Pi,\mathcal{A}} := \Pr\Big[\mathsf{COMPL}_{\Pi,\mathcal{A}}(\lambda) = \mathbf{true}\Big] = \mathsf{negl},$$

  where $\mathsf{COMPL}_{\Pi,\mathcal{A}}(\lambda)$ is the game depicted in Figure 3.3.

- **Succinctness.** There exists a fixed polynomial $p(\cdot)$ independent of $\mathcal{R}$ such that for every large enough security parameter $\lambda \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance $y = (M, x, t)$ such that $t \leq T$, we have
  - Gen runs in time $\begin{cases} p(\lambda + \log T) & \text{for a fully-succinct SNARG} \\ p(\lambda + T) & \text{for a pre-processing SNARG} \end{cases}$
  - Prove runs in time $\begin{cases} p(\lambda + |M| + |x| + t + \log T) & \text{for fully-succinct SNARG} \\ p(\lambda + |M| + |x| + T) & \text{for pre-processing SNARG} \end{cases}$
  - Ver runs in time $p(\lambda + |M| + |x| + \log T)$
  - a honestly generated proof has size $|\pi| = p(\lambda + \log T)$.

- **Knowledge Soundness.** For every PPT adversarial prover $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ such that for every large enough $\lambda \in \mathbb{N}$, every benign auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and every time bound $T \in \mathbb{N}$, for the relation $\mathcal{R}$ such that $t \leq T$ it holds:

$$\mathsf{Adv}^{\mathsf{ks}}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}} := \Pr\Big[\mathsf{KS}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda) = \mathbf{true}\Big] = \mathsf{negl},$$

where $\mathsf{KS}_{\Pi,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda)$ is defined in Figure 3.3.

We call a zk-SNARK, a SNARK for which the zero knowledge property holds:

- **Statistical Zero-knowledge.** There exists a stateful interactive polynomial-size simulator $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{crs}}, \mathsf{Sim}_{\mathsf{proof}})$ such that for every large enough security parameter $\lambda \in \mathbb{N}$, auxiliary input $\mathsf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, time bound $T \in \mathbb{N}$ and relation $\mathcal{R}$ and for all stateful interactive distinguishers $\mathcal{A}$, $\forall\ (u,w) \in \mathcal{R}$:

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathsf{Sim},\mathcal{A}} \coloneqq \Pr\left[\mathsf{ZK}_{\Pi,\mathsf{Sim},\mathcal{A}}(\lambda) = \mathbf{true}\ \right] = \mathsf{negl},$$

where $\mathsf{ZK}_{\Pi,\mathsf{Sim},\mathcal{A}}(\lambda)$ is defined in Figure 3.3.

Adaptive Soundness. A SNARG is called *adaptive* if the prover can choose the statement u to be proved after seeing the reference string crs and the argument remains sound.

SNARG vs. SNARK. If we replace the Knowledge Soundness with the following weaker property, *(adaptive) soundness* we obtain what we call a SNARG, a succinct non-interactive argument:

- **Adaptive Soundness.** For every PPT adversarial prover $\mathcal{A}$ there is a negligible function $\varepsilon(\lambda)$ such that for every time bound $T \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c|c} \mathsf{Ver}(\mathsf{vrs}, u, \pi) = 1 & (\mathsf{crs}, \mathsf{vrs}) \leftarrow \mathsf{Gen}(1^\lambda, T) \\ \wedge\ u \notin \mathcal{L}_{\mathcal{R}} & (u, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \varepsilon(\lambda)$$

The adaptive soundness is a weaker security notion than the adaptive knowledge soundness.

Publicly verifiable vs. Designated Verifier. In the same line of past works [DFGK14, ABLZ17, Fuc18], we will assume for simplicity that crs can be extracted from the verification key vrs.

If security (adaptive KS) holds against adversaries that also have access to the verification state vrs, then the SNARK is called *publicly verifiable*; otherwise it is *designated verifier*.

SNARK Constructions. The framework for constructing SNARKs starts with finding a "good" characterization of the complexity class NP and take advantage of its specific properties for applying some compression techniques on top.

Indeed, by choosing a suitable NP-complete problem representation as in Section 3.1, we are able to construct generic SNARK schemes for NP-complete languages. For example, many SNARKs have as a departure point the circuit satisfiability (Circ-SAT) problem. Circ-SAT problem is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output true. Circ-SAT has been proven to be NP-complete. Another very useful characterisation of the NP-complete class are the Probabilistically Checkable Proofs (PCP). Using this characterisation, we can give a framework for constructing SNARKs that was exploited by many works in the field [Mic94, CL08, GLR11, BCCT12, DFH12, BSBHR18].

Since our contributions follow a different approach for constructing SNARKs, we will not give all the technical details of this line of works. For the completeness of this short survey, we will give an informal overview of some preliminary constructions and then we will focus on the state of art, the SNARKs for circuits.

3.3 SNARK: Construction from PCP

In this section, we provide some high-level intuition for some notable SNARK construction in the literature, introduced by the work "The hunting of the SNARK" [BCC+14].

This methodology to construct SNARKs is based on PCP characterization of NP, and it is first achieved in the random oracle model (ROM), which gave only heuristical security. The idea is to apply the random-oracle-based Fiat-Shamir transform to Kilian's succinct PCP-based proof system [Kil92], achieving logarithmic proof size and verification time.

Later, the construction is improved by removing the use of the random oracles and replacing them with extractable collision-resistant hash functions (ECRH).

We first informally introduce Probabilistically Checkable Proofs (PCP), a characterisation of the NP-class.

3.3.1 Probabilistically Checkable Proofs

The original version of the PCP Theorem [ALM+98] states that proofs for any NP language can be encoded in such a way that their validity can be verified by only reading a constant number of bits, and with an error probability that is upper bounded by a constant. The class PCP is a generalization of the proof verifying system used to define NP, with the following changes:

Probabilistic Verifier. The verifier is probabilistic instead of deterministic. Hence, the verifier can have different outputs for the same inputs x.

Random Access to the Proof. The verifier has random access to the proof string π. This means each bit in the proof string can be independently queried by the verifier via a special address tape: If the verifier desires say the i-th bit in the proof of the string, it writes i in base-2 on the address tape and then receives the bit π_i.

Constant Number of Queries. We are interested in probabilistic verification procedures that access only a few locations in the proof [ALM+98], and yet are able to make a meaningful probabilistic verdict regarding the validity of the alleged proof. Specifically, the verification procedure should accept any valid proof (with probability 1) but rejects with probability at least 1/2 any alleged proof for a false assertion.

Adaptiveness. Verifiers can be adaptive or non-adaptive. A non-adaptive verifier selects its queries based only on its inputs and random tape, whereas an adaptive verifier can, in addition, rely upon bits it has already queried in π to select its next queries.

The fact that one can (meaningfully) evaluate the correctness of proofs by examining few locations in them is indeed surprizing and somewhat counter-intuitive. Needless to say, such proofs must be written in a somewhat non-standard format, because standard proofs cannot be verified without reading them in full (since a flaw may be due to a single improper inference). In contrast, proofs for a PCP system tend to be very redundant; they consist of superfluously many pieces of information (about the claimed assertion), but their correctness can be (meaningfully) evaluated by checking the consistency of a randomly chosen collection of few related pieces.

NP-proofs can be efficiently transformed into a (redundant) form that offers a trade-off between the number of locations (randomly) examined in the resulting proof and the confidence in its validity.

A more formal definition follows:

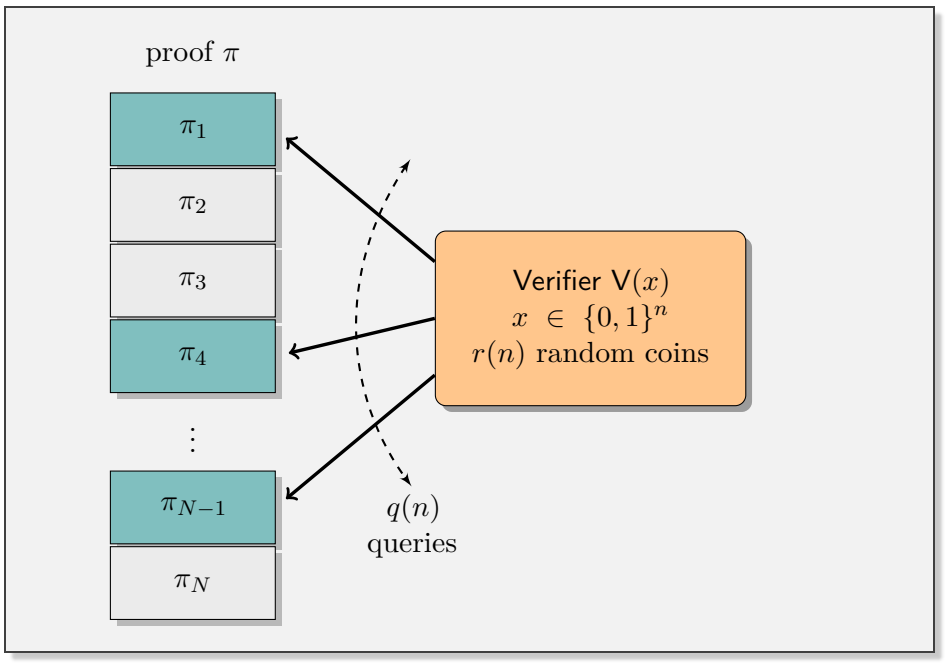


Figure 3.4: A PCP verifier V for a language $\mathcal{L}$ with input x and random access to a string π.

Definition 3.3.1 (Probabilistically Checkable Proofs)**.** *Let $\mathcal{L}$ be a language and $q, r : \mathbb{N} \to \mathbb{N}$. A probabilistically checkable proof system $\mathsf{PCP}(r(n), q(n))$ for $\mathcal{L}$ is a probabilistic polynomial-time oracle machine, called verifier and denoted V, that satisfies the following conditions:*

Efficiency. On input a string $x \in \{0,1\}^n$, and given a random access to a string π called the proof, V uses at most $r(n)$ random coins and makes at most $q(n)$ queries to locations of π (see Figure 3.4). Then it outputs 1 (for "accept") or 0 (for "reject").

Completeness. For every $x \in \mathcal{L}$ there exists a proof string π such that, on input x and access to oracle π, machine V always accepts x.

Soundness. For every $x \notin \mathcal{L}$ and every proof string π, on input x and access to oracle π, machine V rejects x with probability at least $1/2$.

The error probability (in the soundness condition) of PCP systems can be reduced by successive applications of the proof system. In particular, repeating the process for k times, reduces the probability that the verifier is fooled by a false assertion to 2^{-k}, whereas all complexities increase by at most a factor of k. Thus, PCP systems of non-trivial query-complexity provide a trade-off between the number of locations examined in the proof and the confidence in the validity of the assertion.

We say that a language $\mathcal{L}$ is in $\mathsf{PCP}(r(n), q(n))$ if $\mathcal{L}$ has a $(cr(n), dq(n))$-verifier V for some constants c, d.

Theorem 3.3.2 (PCP Theorem [ALM+98])**.**

$$\mathsf{NP} = \mathsf{PCP}(\log n, 1)$$

Kilian Interactive Argument of Knowledge from PCP. When PCP theorem [ALM+98] came out, it revolutionized the notion of "proof" – making the verification possible in time polylogarithmic in the size of a classical proof. Kilian adapted this PCP characterization of NP to the cryptographic setting, showing that one can use PCPs to construct interactive arguments (i.e., computationally sound proof systems [BCC88]) for NP that are succinct – i.e., polylogarithmic also in their communication complexity. In his work [Kil92], Kilian presents a succinct zero-knowledge argument for NP where the prover P uses a Merkle tree (see definition in Section 2.4.1) in order to provide to the verifier V "virtual access" to a PCP proof π. An example of such a Merkle tree is illustrated in Figure 3.5.

More precisely, to prove a statement $x \in \mathcal{L}$ we need the following interactions:

1. The verifier starts by sending the prover a collision resistant hash function (CRHF) H (in the sense of Section 2.4.1).
   The prover, on private input a witness w, constructs a PCP-proof π.
   In order to yield efficient verifiability, P cannot send to V the witness w, nor π.
   Instead, it builds a Merkle tree with the proof π as the leaf values (using the collision-resistant hash function H from the verifier) producing a root value.
2. The prover sends this root to V as a commitment to π.
3. V tosses a fixed polynomial number of random coins and sends them to P.
   Both the prover P and the verifier V compute the PCP queries by internally running the PCP verifier on input x and root.
4. The prover P sends back answers to those queries, together with "proofs"– called authentication paths – that each answer is consistent with the root of the Merkle tree.
   Finally, the verifier accepts if all the answers are consistent with the root value, and convinces the PCP.

Kilian's protocol is succinct, because the verifier V, invoking the PCP verifier, makes only a fixed polynomial number of queries and each query is answered with an authentication path of some fixed polynomial length, all independent of the length of the witness.

At a very high-level, the soundness follows from the fact that the Merkle tree provides the verifier "virtual access" to the PCP proof, in the sense that given the root value of the Merkle tree, for every query q, it is infeasible for a cheating prover to answer q differently depending on the queries. Therefore, interacting with the prover is "equivalent" to having access to a PCP proof oracle. Then it follows from the soundness of the PCP system that Kilian's protocol is sound.

Micali CS proofs. Micali [Mic94] showed how to make interactive arguments non-interactive by applying the Fiat-Shamir heuristic (see Section 3.1.3) to Kilian's construction. The idea was to apply a hash function, modeled as a random oracle, to its PCP string both as a form of commitment and to non-interactively generate the verifier's PCP queries.

Private Information Retrieval (PIR). The work of [CL08] proposed the PCP+MT+PIR approach to "squash" Kilian's four-message protocol into a two-message protocol. To understand their techniques, we briefly define Private Information Retrieval (PIR) schemes.

A (single-server) polylogarithmic private information retrieval (PIR) scheme ([CMS99, Lip05, GR05, BV11a]) allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved.

One trivial, but very inefficient way to achieve PIR is for the server to send an entire copy of the database to the user. There are two ways to address this problem: one is to

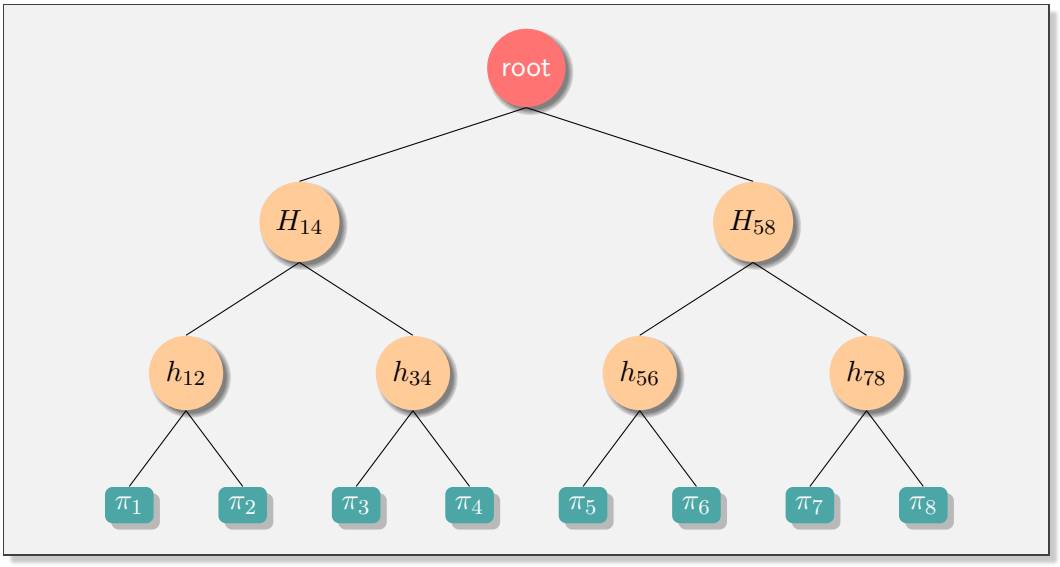


Figure 3.5: A Merkle tree for commiting to the string π. Each inner node of the tree is the hash value of the concatenation of its two children: $h_{ij} = h(\pi_i \| \pi_j), H_{ik} = h(h_{ij} \| h_{j+1k}), \mathsf{root} = h(H_{14} \| H_{58})$, for $i, j, k \in [8]$.

make the server computationally bounded, and the other is to assume that there are multiple non-cooperating servers, each having a copy of the database. We will consider the first one that assumes bounded running times and succinctness of the server answers. More formally:

Definition 3.3.3 (Private Information Retrieval)**.** *A (single-server) polylogarithmic* private information retrieval *(PIR) scheme consists of a triple of algorithms* $(\mathsf{PEnc}, \mathsf{PEval}, \mathsf{PDec})$ *that work as follow:*

$\mathsf{PEnc}(1^\lambda, i, r)$**:** outputs an encryption C_i of query i to a database DB using randomness r,
$\mathsf{PEval}(DB, C_i)$**:** outputs a succinct blob e_i "containing" the answer $DB[i]$,
$\mathsf{PDec}(e_i)$**:** decrypts the blob e_i to an answer $DB[i]$.

The three proprieties a PIR scheme should satisfy are corectness, succinctness (the running time of both $\mathsf{PEnc}, \mathsf{PEval}$ should be bounded) and semantic security, in the sense that the encryptions of indexes i with the PEnc algorithm should not reveal information about their value.

The PCP+MT+PIR Approach. We have seen that in Kilian's protocol, the verifier obtains from the prover a Merkle hash to a PCP oracle and only then asks the prover to locally open the queries requested by the PCP verifier. In [CL08]'s construction, the verifier also sends in the first message, a PIR-encrypted version of the PCP queries (the first message of a PIR scheme can be viewed as an encryption to the queries); the prover then prepares the required PCP oracle, computes and sends a Merkle hash of it, and answers the verifier's queries by replying to the PIR queries according to a database that contains the answer (as well as the authentication path with respect to the Merkle hash) to every possible verifier's query. Di Crescenzo and Lipmaa [CL08] proved the soundness of the above scheme based on the assumption that any convincing prover P must essentially behave as an honest prover:

Namely, if a proof is accepting, then the prover must have in mind a full PCP oracle, which maps under the Merkle hash procedure to the claimed root, and such a proof π can be obtained by an efficient extractor $\mathcal{E}$.

They then showed that, if this is the case, the extracted string π must be consistent with the answers the prover provides to the PCP queries, for otherwise the extractor can be used to obtain collisions of the hash function underlying the Merkle tree. Therefore, the extracted string π also passes the PCP test, where the queries are encrypted under PIR. Then, it follows from the privacy of the PIR scheme that, the string π is "computationally independent" of the query. Hence from the soundness of PCP, they conclude that the statement must be true.

3.3.2 SNARKs from PCP

We have mentioned two methodologies that can be applied to obtain SNARGs from PCPs, one is the Fiat-Shamir heuristic in the random oracle model, the other is the PCP+MT+PIR Approach. In both cases, we do not obtain knowledge soundness, but only plain adaptive soundness. Recent works [GLR11, BCCT12, DFH12, BCC+14] have improved Micali's construction by adding knowledge soundness and removing the random oracles, replacing them with "extractable collision-resistant hash functions" (ECRHs), a non-falsifiable extractability assumption.

Extractable Collision-Resistant Hash. We start by defining a natural strengthening of collision-resistant hash functions introduced in Section 2.4.1: the extractable collision-resistant hash functions (ECRH). An ECRH function family satisfies the two following properties:

- it is collision-resistant in the standard sense 2.4.1,
- it is extractable in the sense that for any efficient adversary that is able to produce a valid evaluation of the function there is an extractor that is able to produce a corresponding preimage.

The ECRH+PIR Approach. The [BCC+14] construction obtains the stronger notion of knowledge soundness arguments, SNARKs, and also a pre-processed protocol rather than one-round of communication, based on the more restrictive assumption that ECRHs exist. At a very high-level, their construction modifies the PCP+MT+PIR approach by replacing the CRHF underlying the Merkle tree with an ECRH. The additional features of this modified construction are:

(a) The verifier's message can be generated offline independently of the theorem being proved and thus we refer to this message as a verifier-generated reference string (VGRS);

(b) The input can be chosen adaptively by the prover based on previous information, including the VGRS;

(c) The construction is an (adaptive) argument of knowledge;

(d) The running time of the verifier and the proof length are "universally succinct"; in particular, they do not depend on the specific NP-relation at hand.

On the other hand, the scheme is only designated-verifier.

The main challenges in [BCC+14] construction and the required modifications they make to [CL08] are briefly mentioned in the following.

Extracting a witness. To obtain knowledge soundness, they first instantiate the underlying PCP system with PCPs of knowledge, which allow for extracting a witness from any sufficiently-satisfying proof oracle.

Adaptivity. In their setting, the prover is allowed to choose the claimed theorem after seeing the verifier's first message (or, rather, the verifier-generated reference string). In order to enable the (honest) verifier to do this, they PIR-encrypt the PCP verifier's coins rather than its actual queries (as the former are independent of the instance), and require the prover to prepare an appropriate database (containing all the possible answers for each setting of the coins, rather than a proof oracle).

From local to global extraction. Unlike [CL08], which directly assumed the "global extraction" guarantees from a Merkle tree, Bitansky et al. show that the "local extraction" guarantee can be lifted using ECRH functions instead of simple CRHF, to the "global extraction" guarantee on the entire Merkle tree. The main technical challenge in their construction is establishing this "global" knowledge feature, more precisely, to obtain an extracted PCP proof π that will be sufficiently satisfying for extracting a witness from a very "local" one (namely, the fact that it is infeasible to produce images of the ECRH without actually knowing a preimage).

To achieve this, they start from the root of the Merkle tree and "work back towards the leaves"; that is, extract a candidate proof π by recursively applying the ECRH-extractor to extract the entire Merkle tree, where the leaves should correspond to π. However, recursively composing ECRH-extractors already encounters a difficulty: each level of extraction incurs a polynomial blowup in computation size. Hence, without making a very strong assumption on the amount of "blowup" incurred by the extractor, one can only apply extraction a constant number of times. This problem is solved by replacing the binary Merkle tree with a *squashed* Merkle tree, where the fan-in of each node is polynomial rather than binary as is usually the case.

Knowledge Soundness. Given the previous discussion, knowledge soundness of the entire scheme is shown in two steps:

Local Consistency. They show that whenever the verifier is convinced, the recursively extracted string contains valid answers to the verifier's PCP queries specified in its PIR queries. Otherwise, it is possible to find collisions within the ECRH as follows. A collision finder could simulate the PIR-encryption on its own, invoke both the extraction procedure and the prover, and obtain two paths that map to the same root but must differ somewhere (as one is satisfying and the other is not) and therefore obtain a collision.

From Local to Global Consistency. Next, using the privacy guarantees of the PIR scheme, they show that, whenever one extracts a set of leaves that are satisfying with respect to the PIR-encrypted queries, the same set of leaves must also be satisfying for almost all other possible PCP queries and is thus sufficient for witness-extraction. Indeed, if this was not the case then one would be able to use the polynomial-size extraction circuit to break the semantic security of the PIR.

Furthermore, the [BCC+14] construction achieves a communication complexity and a verifier's time complexity bounded by a polynomial in the security parameter, the size of the instance, and the logarithm of the time it takes to verify a valid witness for the instance, obtaining a fully-succinct SNARK.

3.4 SNARK: Construction from QAP

We will present here the methodology for building SNARKs common to a family of constructions, some of which represent the state of the art in the field.

Most constructions and implementations of SNARKs [PHGR13, Lip13, DFGK14, Gro16, GMNO18] have as a central starting point the framework based on quadratic programs introduced by Gennaro et al. in [GGPR13]. This common framework allows to build SNARKs for programs instantiated as boolean or arithmetic circuits.

This approach has led to fast progress towards practical verifiable computations. For instance, using span programs for arithmetic circuits (QAPs), Pinocchio [PHGR13] provides evidence that verified remote computation can be faster than local computation. At the same time, their construction is zero-knowledge, enabling the server to keep intermediate and additional values used in the computation private.

Optimized versions of SNARKs based on QAP approach are used in various practical applications, including cryptocurrencies such as Zcash [BCG+14], to guarantee anonymity via the ZK property while preventing double-spending.

Circuits and Circ-SAT Problem. A SNARK scheme for a circuit has to enable verification of proofs for (Arithmetic or Boolean) Circ-SAT problem, i.e., a prover, given a circuit has to convince the verifier that it knows an assignment of its inputs that makes the output true. In the following definitions, we may see a circuit $\mathtt{C}$ as a logical specification of a satisfiability problem.

Arithmetic Circuits. Informally, an arithmetic circuit consists of wires that carry values from a field $\mathbb{F}$ and connect to addition and multiplication gates. See Figure 3.6 for an example.

Boolean Circuits. A boolean circuit consists of logical *gates* and of a set of *wires* between the gates. The wires carry values over $\{0,1\}$. See Figure 3.7 for an example.

Associated to any circuit, we define a satisfaction problem as follows:

Definition 3.4.1 (Circuit Satisfaction Circ-SAT)**.** *The circuit satisfaction problem of a circuit* $\mathcal{C}: I_u \times I_w \to \{0,1\}$ *is defined by the relation* $\mathcal{R}_{\mathcal{C}} = \{(\mathbf{u},\mathbf{w}) \in I_u \times I_w : \mathcal{C}(\mathbf{u},\mathbf{w}) = 1\}$ *and its language is* $\mathcal{L}_{\mathcal{C}} = \{\mathbf{u} \in I_u : \exists\ \mathbf{w} \in I_w, \mathcal{C}(\mathbf{u},\mathbf{w}) = 1\}$.

Standard results show that polynomially sized circuits are equivalent (up to a logarithmic factor) to Turing machines that run in polynomial time, though of course the actual efficiency of computing via circuits versus on native hardware depends heavily on the application; for example, an arithmetic circuit for matrix multiplication adds essentially no overhead, whereas a boolean circuit for integer multiplication is far less efficient.

3.4.1 From Circuits to Quadratic/Square Programs

Back in 2013, Gennaro, Gentry, Parno and Raykova [GGPR13] proposed a new, influential characterization of the complexity class NP using *Quadratic Span Programs* (QSPs), a natural extension of span programs defined by Karchmer and Wigderson [KW93].

Some variants and improvements of QSPs followed. In [Lip13], Lipmaa gave a class of more efficient quadratic span programs by combining the existing techniques with linear error-correcting codes.

Parno et al. [PHGR13] defined QAP, a similar notion for arithmetic circuits, namelly *Quadratic Arithmetic Programs.*

More recently, an improved version for boolean circuits, the *Square Span Programs* (SSP) was presented by [DFGK14]. Naturally, this led to a simplified version for arithmetic circuits in the same spirit, *Square Arithmetic Programs* (SAP), proposed in [GM17].

These are methods to compactly encode computations, so as to obtain efficient zero-knowledge SNARKs.

The main idea is to represent each gate inputs and outputs as a variable. Then we may rewrite each gate as an equation in some variables representing the gate's input and output wires. These equations are satisfied only by the values of the wires that meet the gate's logic or arithmetic specification. By composing such constraints for all the gates in the circuit, a satisfying assignment for any circuit can be specified first as a set of quadratic equations, then as a constraint on the span of a set of polynomials, defining the corresponding *Quadratic/Square Program* for the circuit. As a consequence, the prover needs to convince the verifier that all the quadratic equations are satisfiable by finding a solution of the equivalent polynomial problem.

For the sake of completness, we will define here the two most popular languages for SNARKs: Quadratic Arithmetic Programs, that are central to the most efficient SNARK implementations known today and Square Span Programs, that are easy to understand, efficient languages for boolean circuits.

Quadratic Arithmetic Programs (QAPs). Before formally defining QAPs, we walk through the steps for encoding the toy example circuit in Figure 3.6 into an equivalent QAP.

First, we select two arbitrary values from some field $\mathbb{F}$ of order p: $r_5, r_6 \in \mathbb{F}$ to represent the two multiplication gates (the addition gates will be compressed into their contributions to the multiplication gates).

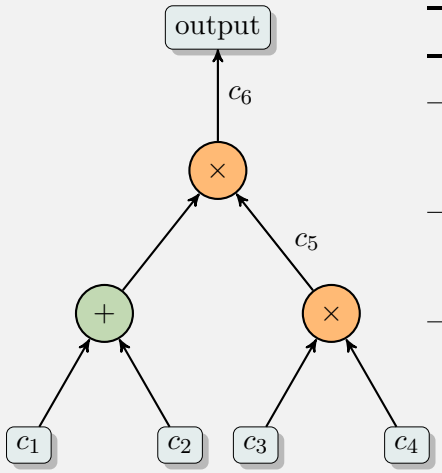
We define three sets of polynomials $\mathcal{V} = \{v_i(x)\},\ \mathcal{W} = \{w_i(x)\}$ and $\mathcal{Y} = \{y_i(x)\},\ i \in [6]$ by letting the polynomials in $\mathcal{V}$ encode the left input into each multiplication gate, the $\mathcal{W}$ encode the right input into each gate, and the $\mathcal{Y}$ encode the outputs. Thus, for the circuit in Figure 3.6, we define six polynomials for each set $\mathcal{V},\ \mathcal{W}$ and $\mathcal{Y}$, four for the input wires, and two for the outputs from the multiplication gates.

We define these polynomials based on each wire's contributions to the multiplication gates. Specifically all of the $v_i(r_5) = 0$, except $v_3(r_5) = 1$, since the third input wire contributes to the left input of c_5's multiplication gate. Similarly, $v_i(r_6) = 0$, except for $v_1(r_6) = v_2(r_6) = 1$, since the first two inputs both contribute to the left input of c_6's gate. For $\mathcal{W}$, we look at right inputs. Finally, $\mathcal{Y}$ represents outputs; none of the input wires is an output, so $y_i(r_5) = y_i(r_6) = 0$ for $i \in [4]$ and $y_5(r_5) = y_6(r_6) = 1$. We can use this encoding of the circuit to efficiently check that it was evaluated correctly.

More generally, we define a QAP, an encoding of an arithmetic function, as follows.

Definition 3.4.2 (QAP)**.** *A Quadratic Arithmetic Program Q over the field $\mathbb{F}$ contains three sets of $m+1$ polynomials $\mathcal{V} = \{v_i(x)\},\ \mathcal{W} = \{w_i(x)\}$ and $\mathcal{Y} = \{y_i(x)\},\ i \in \{0, 1 \dots m\}$ and a target polynomial $t(x)$. Suppose F is an arithmetic function that takes as input n elements of $\mathbb{F}$ and outputs n' elements, for a total of $N = n + n'$ I/O elements. Then, $(c_1, \dots, c_N) \in \mathbb{F}^N$ is a valid assignment of F's inputs and outputs, if and only if there exist coefficients $(c_{N+1}, ..., c_m)$ such that $t(x)$ divides $p(x)$, where:*

$$p(x) := \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right) \cdot \left(w_0(x) + \sum_{i=1}^{m} c_i w_i(x)\right) - \left(y_0(x) + \sum_{i=1}^{m} c_i y_i(x)\right). \tag{3.1}$$



Roots	**Polynomials in QAP** $(\mathcal{V}, \mathcal{W}, \mathcal{Y}, t(x))$		
Gates	Left inputs	Right inputs	Outputs
r_5	$v_3(r_5) = 1$ $v_i(r_5) = 0$, $i \neq 3$	$w_4(r_5) = 1$ $w_i(r_5) = 0$, $i \neq 4$	$y_5(r_5) = 1$ $y_i(r_5) = 0$, $i \neq 5$
r_6	$v_1(r_6) = v_2(r_6) = 1$ $v_i(r_6) = 0$, $i \neq 1, 2$	$w_5(r_6) = 1$ $w_i(r_6) = 0$, $i \neq 5$	$y_6(r_6) = 1$ $y_i(r_6) = 0$, $i \neq 6$
$t(x) = (x - r_5)(x - r_6)$			

Figure 3.6: Arithmetic Circuit and Equivalent QAP. The polynomials $\mathcal{V} = \{v_i(x)\}, \mathcal{W} = \{w_i(x)\}, \mathcal{Y} = \{y_i(x)\}$ and the target polynomial $t(x)$ in the QAP are defined in terms of their evaluations at the two roots, one for each multiplicaive gate, $r_5, r_6 \in \mathbb{F}$.

In other words, there must exist some polynomial $h(x)$ such that $h(x)t(x) = p(x)$. We say that the QAP Q computes F. The size of Q is m, and the degree d is the degree of $t(x)$.

In [PHGR13], the authors show that for any arithmetic circuit with d multiplication gates and N I/O elements, one can construct an equivalent QAP with degree (the number of roots) d and size (number of polynomials in each set) $m = d + N$. Note that addition gates and multiplication-by-constant gates do not contribute to the size or degree of the QAP. Thus, these gates are essentially "free" in QAP-based SNARKs.

Building a QAP Q for a general arithmetic circuit $\mathtt{C}$ is fairly straightforward:

We pick an arbitrary root $r_g \in \mathbb{F}$ for each multiplication gate g in $\mathtt{C}$ and define the target polynomial to be $t(x) = \prod_g (x - r_g)$.

We associate an index $i \in [m]$ to each input of the circuit and to each output from a multiplication gate.

Finally, we define the polynomials in $\mathcal{V}, \mathcal{W}$ and $\mathcal{Y}$ by letting the polynomials in $\mathcal{V}, \mathcal{W}$ encode the left/right input into each gate, and $\mathcal{Y}$ encode the outputs of the gates: $v_i(r_g) = 1$ if the i-th wire is a left input to gate g, and $v_i(r_g) = 0$ otherwise. Similarly, we define the values of polynomials $w_i(r_g)$ and $y_i(r_g)$.

Thus, if we consider a particular gate g and its root r_g, Equation (3.1) and the constraint $p(r_g) = t(r_g)h(r_g) = 0$ just says that the output value of the gate is equal to the product of its inputs, the very definition of a multiplication gate.

For example, in the QAP for the circuit in Figure 3.6, if we evaluate $p(x)$ at r_5, we get $c_3 c_4 = c_5$, which directly encodes the first multiplication gate, and similarly, at r_6, $p(x)$ simplifies to $(c_1 + c_2)c_5 = c_6$, that is, an encoding of the second multiplication gate.

In short, the divisibility check that $t(x)$ divides $p(x)$ decomposes into $d = \deg(t(x))$ separate checks, one for each gate g and root r_g of $t(x)$, that $p(r_g) = 0$.

Span Square Programs (SSPs). Danezis et al. [DFGK14] found a way to *linearize* all logic gates with fan-in 2 in a boolean circuit. This starts from the observation that any 2-input binary gate $g(a, b) = c$ with input wires a, b and output c can be specified using an affine combination $L = \alpha a + \beta b + \gamma c + \delta$ of the gate's input and output wires that take exactly two values, $L = 0$ or $L = 2$, when the wires meet the gate's logical specification. This leads to an equivalent single "square" constraint $(L - 1)^2 = 1$. We refer to Figure 3.7 for the truth table and simple linearization of some gates in a toy example.

Composing such constraints, a satisfying assignment for any binary circuit can be specified first as a set of affine map constraints, then as a constraint on the span of a set of polynomials, defining the square span program for this circuit.

Due to their conceptual simplicity, SSPs offer several advantages over previous constructions for binary circuits. Their reduced number of constraints lead to smaller programs, and to lower sizes and degrees for the polynomials required to represent them, which in turn reduce the computation complexity required in proving or verifying SNARKs.

Let $\mathtt{C}$ be a boolean circuit with m wires and n fan-in 2 gates. We formally define SSPs (See [DFGK14]):

Definition 3.4.3 (SSP)**.** *A Square Span Program (SSP) S over the field $\mathbb{F}$ is a tuple consisting of $m+1$ polynomials $v_0(x), \ldots, v_m(x) \in \mathbb{F}[x]$ and a target polynomial $t(x)$ such that $\deg(v_i(x)) \leqslant \deg(t(x))$ for all $i = 0, \ldots, m$. We say that the square span program SSP has size m and degree $d = \deg(t(x))$. We say that SSP accepts an input $c_1, \ldots, c_N \in \{0, 1\}$ if and only if there exist $c_{N+1}, \ldots, c_m \in \{0, 1\}$ such that $t(x)$ divides $p(x)$, where:*

$$p(x) \coloneqq \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right)^2 - 1.$$

We say that SSP S verifies a boolean circuit $\mathtt{C} : \{0, 1\}^N \to \{0, 1\}$ if it accepts exactly those inputs $(c_1, \ldots, c_N) \in \{0, 1\}^N$, satisfying $\mathtt{C}(c_1, \ldots c_N) = 1$.

Theorem 3.4.4 ([DFGK14, Theorem 2])**.** *For any boolean circuit $\mathtt{C}$ of m wires and n fan-in 2 gates and for any prime $p \geq \max(n, 8)$, there exist polynomials $v_0(x), \ldots, v_m(x)$ such that, for any distinct roots $r_1, \ldots, r_d \in \mathbb{F}$, $\mathtt{C}$ is satisfiable if and only if:*

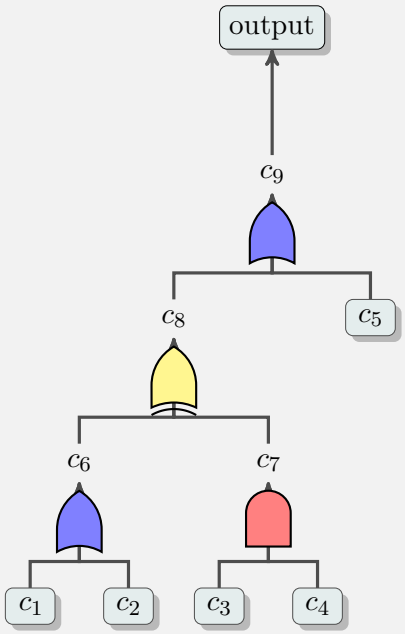
$$\prod_{i=1}^{d}(x - r_i) \text{ divides } p(x) \coloneqq \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right)^2 - 1,$$

where $c_1, \ldots, c_m \in \{0, 1\}$ correspond to the values on the wires in a satisfying assignment for the circuit.

Define $t(x) \coloneqq \prod_{i=1}^{d}(x - r_i)$, then for any circuit $\mathtt{C}$ of m wires and n gates, there exists a degree $d = m + n$ square span program $S = (v_0(x), \ldots, v_m(x), t(x))$ over a field $\mathbb{F}$ of order p that verifies $\mathtt{C}$.

Building a SSP S for a general boolean circuit $\mathtt{C} : \{0, 1\}^N \to \{0, 1\}$ with m wires and n fan-in 2 gates follows some simple steps (See Figure 3.7 for a toy example).

First, we represent an assignment to the wires of $\mathtt{C}$ as a vector $\mathbf{c} \in \{0, 1\}^m$. The assignment is a satisfying witness for the circuit if and only if the inputs belong to $\{0, 1\}$, the inputs respect all gates, and the output wire is 1. It is easy to impose the condition $c_i \in \{0, 1\}$, $\forall i \in [m]$ by requiring $2c_i \in \{0, 2\}$. Scaling some of the gate equations from Figure 3.7 by a factor 2,



Linearization of logic gates

OR ($c_1 \vee c_2 = c_6$)			AND ($c_3 \vee c_4 = c_7$)		
c_1	c_2	c_6	c_3	c_4	c_7
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	1
$-c_1 - c_2 + 2c_6 \in \{0,1\}$			$c_3 + c_4 - 2c_7 \in \{0,1\}$		

XOR gate, input and output bits

XOR ($c_6 \oplus c_7 = c_8$)			IN ($\{c_i\}_{i=1}^8$)	OUT (c_9)
c_6	c_7	c_8	c_i	c_9
0	0	0	$\in \{0,1\}$	1
0	1	1		
1	0	1		
1	1	0		
$c_6 + c_7 + c_8 \in \{0,2\}$			$2c_i \in \{0,2\}$	$3 - 3c_9 \in \{0,2\}$

Figure 3.7: Boolean circuit and the linearization of its logic gates.

we can write all gate equations in the form $L = \alpha c_i + \beta c_j + \gamma c_k + \delta \in \{0,2\}$. We want the circuit output wire c_{out} to have value 1. We do that by adding the condition $3 - 3c_{\mathsf{out}}$ to the linearization of the output gate.

We further define a matrix $\mathbf{V} \in \mathbb{Z}^{m \times d}$ and $\mathbf{b} \in \mathbb{Z}^d$ such that $\mathbf{cV} + \mathbf{b} \in \{0,2\}^d$ corresponds to the linearization of the gates and of inputs/outputs as described above. The existence of $\mathbf{c}$ such that $\mathbf{cV} + \mathbf{b} \in \{0,2\}^d$ is equivalent to a satisfying assignment to the wires in the circuit. We can rewrite this condition as

$$(\mathbf{cV} + \mathbf{b}) \circ (\mathbf{cV} + \mathbf{b} - \mathbf{2}) = 0 \iff (\mathbf{cV} + \mathbf{b} - \mathbf{1}) \circ (\mathbf{cV} + \mathbf{b} - \mathbf{1}) = \mathbf{1}, \tag{3.2}$$

where $\circ$ denotes the Hadamard product (entry-wise multiplication).

Next step consists in defining the polynomials $\{v_i(x)\}_{i=0}^m$. Let $r_1, \ldots r_d$ be d distinct elements of a field $\mathbb{F}$ of order p for a prime $p \geq \max(d, 8)$. Define $v_0(x), v_1(x), \ldots v_m(x)$ as the degree $d-1$ polynomials satisfying $v_0(r_j) = b_j - 1$ and $v_i(r_j) = V_{i,j}$.

We can now reformulate condition 3.2 again: The circuit $\mathtt{C}$ is satisfiable if and only if there exists $\mathbf{c} \in \mathbb{F}^m$ such that for all r_j : $\left(v_0(r_j) + \sum_{i=1}^m c_i v_i(r_j)\right)^2 = 1$.

Since the evaluations in $r_1, \ldots r_d$ uniquely determine the polynomial $v_{\mathbf{c}}(x) = v_0(x) + \sum_{i=1}^m c_i v_i(x)$ we can rewrite the condition 3.2:

$$\prod_{i=1}^{d-1}(x - r_i) \text{ divides } \left(v_0(x) + \sum_{i=1}^m c_i v_i(x)\right)^2 - 1.$$

Proving on Top of Quadratic and Square Programs Once we have stated the corresponding quadratic/square span program associated to the (boolean or arithmetic) circuit, the steps in building a proof protocol from this polynomial problem are the following:

Prover. The prover has to solve a SSP (or a QAP) that consists of a set of polynomials $\{v_i(x)\}$ (or respectively $\{v_i(x)\}, \{w_i(x)\}, \{y_i(x)\}$). In both cases, the task is to find a linear combination $\{c_i\}$ of its input polynomials – $v_{\mathbf{c}}(x) = v_0(x) + \sum_i c_i v_i(x)$ (and $w_{\mathbf{c}}(x), y_{\mathbf{c}}(x)$ for QAP) – in such a way that the polynomial $p(x)$ defined by the program is a multiple of another given polynomial $t(x)$.

For a given input, the worker evaluates the circuit `C` directly to obtain the output and the values of the internal circuit wires. These values correspond to the coefficients $\{c_i\}_{i=1}^m$ of the quadratic/square program.

Verifier. From the other side, the verification task consists of checking whether one polynomial divides another polynomial. This can be facilitated by the prover if it sends the quotient polynomial $h(x)$ such that $t(x)h(x) = p(x)$, which turns the task of the verifier into checking a polynomial identity $t(x)h(x) = p(x)$. Put differently, verification consists into checking that $t(x)h(x) - p(x) = 0$, i.e., checking that a certain polynomial is the zero polynomial.

Efficiency. Since the size of these polynomials is very large, the verifier will need a more efficient way to check the validity of the proof, than to multiply such big polynomials. Also, from the point of view of succinctness, sending the polynomials $h(x), v_{\mathbf{c}}(x)$ (and $w_{\mathbf{c}}(x), y_{\mathbf{c}}(x)$ for QAP), each of degrees proportional with the number of gates in the original circuit, is not optimal for our purposes.

Evaluation in a Random Point. So, instead of actually computing polynomial products, the verifier chooses a secret random point s and ask the prover to send the evaluations $h(s), v_{\mathbf{c}}(s)$ (and $w_{\mathbf{c}}(s), y_{\mathbf{c}}(s)$ for QAP) instead of the full polynomials and only checks that $t(s)h(s) = p(s)$. So the polynomial operations are simplified to field multiplications and additions independent of the degree of those polynomials.

Soundness. Checking a polynomial identity only at a single point instead of at all points reduces the security, but according to Schwartz–Zippel lemma any two distinct polynomials of degree d over a field $\mathbb{F}$ can agree on at most a $d/|\mathbb{F}|$ fraction of the points in $\mathbb{F}$. So, if we choose the field $\mathbb{F}$ carefully, $s \leftarrow\!\!\$\, \mathbb{F}$ is assumed to be picked at random and since $t(x)h(x), p(x)$ are non-zero polynomials, the possibility of a false proof to verify is bounded by a negligible fraction (where the evaluations $h(s), p(s)$ are part of, or can be computed from the proof elements). Of course, the point s should be not known in advance by the prover when it generates its polynomials. This is essential to avoid cheating strategies that lead to proofs of false statements.

Encoding the Random Point. We have concluded that the key factor for the soundness to hold is the secrecy of the evaluation point s. The prover should not know in advance this value when computing the solution to SSP $v_{\mathbf{c}}(x), h(s)$ (respectevelly $v_{\mathbf{c}}(x), w_{\mathbf{c}}(x), y_{\mathbf{c}}(x), h(s)$ for QAP). Nevertheless, the prover should be allowed to compute the evaluation of its polynomials in s. Finding a method of hiding s that, at the same time, allows the prover to perform linear operations over this hidden value and the verifier to check the proof, is the key trick in order to build a SNARK.

3.4.1.1 Encoding Schemes

The main ingredient for an efficient preprocessing SNARK is an encoding scheme Enc over a field $\mathbb{F}$ that hides the evaluation point s and has important properties that allow proving and verifying on top of encoded values.

A formalisation of these encoding schemes for SNARKs was initially introduced in [GGPR13]:

Definition 3.4.5 (Encoding Scheme)**.** *An encoding scheme* Enc *over a field* $\mathbb{F}$ *is composed of the following algorithms:*

$\mathsf{K}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ a key generation algorithm that takes as input some security parameter and outputs some secret state sk together with some public information pk.

$\mathsf{Enc}(s) \to z$ an encoding algorithm mapping a field element s to some encoding value. Depending on the encoding algorithm, Enc will require either the public information pk generated from K, or the secret state sk. To ease notation, we will omit this additional argument.

The above algorithms must satisfy the following properties:

- ***additively homomorphic:*** Intuitively, we want the encoding scheme to behave well when applying linear operations $\mathsf{Enc}\,(x + y) = \mathsf{Enc}(x) + \mathsf{Enc}(y)$.
- ***quadratic root detection:*** There exists an efficient algorithm that, given $\mathsf{Enc}(a_0), \ldots,$ $\mathsf{Enc}(a_t)$, and the quadratic polynomial $\mathsf{pp} \in \mathbb{F}[x_0, \ldots, x_t]$, can distinguish if $\mathsf{pp}(a_1, \ldots, a_t) = 0$. We will use an informal notation for this check
$$\mathsf{pp}(\mathsf{Enc}(a_0), \ldots, \mathsf{Enc}(a_t)) \stackrel{?}{=} 0.$$
- ***image verification:*** There exists an efficiently computable algorithm ImVer that can distinguish if an element c is a correct encoding of a field element ($\mathsf{ImVer}(c) \to 0/1$).

Publicly vs Designated-Verifier Encoding. In some instantiations, the encoding algorithm will need a secret state sk to perform the *quadratic root detection.*

If such a secret state is not needed, we will consider $\mathsf{sk} = \perp$ and call it "one-way" or *publicly-verifiable* encoding.

At present, the only candidates for such a "one-way" encoding scheme that we know of are based on bilinear groups, where the the bilinear maps support efficient testing of quadratic degrees without any additional secret information.

Example 3.4.6 (Pairing-based encoding scheme)**.** *Consider a symetric bilinear group of prime order* q *described by* $\mathsf{gk} := (q, \mathbb{G}, \mathbb{G}_T, \mathsf{e})$*, where* $\mathsf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is a biliniar map as defined in Chapter 2. Let* g *be a generator of* $\mathbb{G}$*. We can implement an encoding scheme with the previous properties as:*
$$\mathsf{Enc}(a) = g^a.$$

- *additively homomorphic:* To compute en encoding of a sum $g^{(a_1+a_2)}$, we just multiply the respective group elements $g^{a_1} g^{a_2} = g^{a_1+a_2}$.
For a polynomial $h(x)$, this property can be used to compute an encoding of an evaluation $h(s) = \sum_{i=0}^{d} h_i s^i$ in some point s, given $\{g^{s^i}\}_{i=1}^{d}$ as a linear combination
$$\prod_{i=0}^{d} g^{h_i} g^{s^i} = g^{\sum_{i=0}^{d} h_i s^i} = g^{h(s)}$$

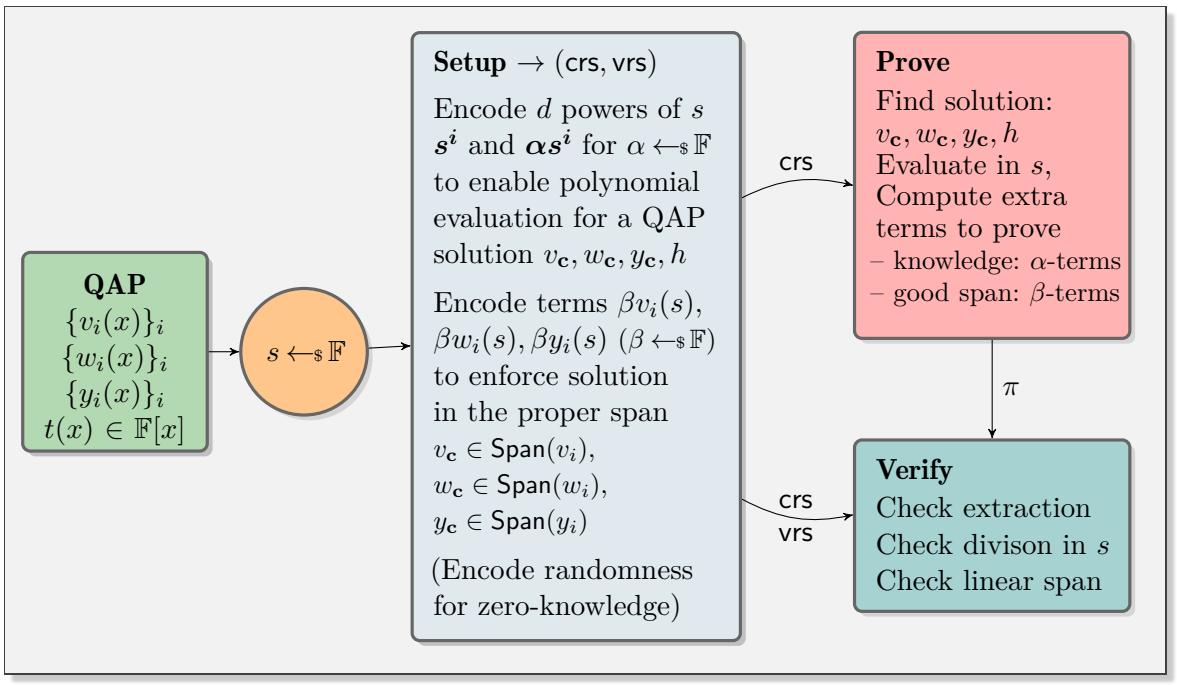


Figure 3.8: Simplified roadmap from QAP to a SNARK protocol.

- *quadratic root detection:* Given for example the following quadratic polynomial $\mathsf{pp}_0 = x_1x_2 + x_3^2$ and some encodings $(g^{a_1}, g^{a_2}, g^{a_3})$ use the bilinear map to check the equality:

$$\mathsf{e}(g,g)^{\mathsf{pp}_0(a_1,a_2,a_3)} = \mathsf{e}(g,g)^{a_1a_2+a_3^2} = \mathsf{e}(g^{a_1}, g^{a_2}) \cdot \mathsf{e}(g^{a_3}, g^{a_3}) \stackrel{?}{=} \mathsf{e}(g,g)^0.$$

- *image verification:* Typically, it is straightforward to determine whether an element is in the group $\mathbb{G}$, and all elements of $\mathbb{G}$ are valid encodings.

Remark 3.4.7. *Remark that none of the three properties requires any secret state, this leads to a publicly-verifiable SNARK, to perform the checks; the verification algorithm does not need anything else than the pairing function* e *that is public. Note also that this encoding scheme is deterministic.*

For instance, the family of elliptic curves $\mathbb{G} \coloneqq E(\mathbb{F}_q)$. described in [BF01] satisfies the above description.

3.4.2 SNARKs from QAP

In what follows, we will present the celebrated SNARK construction of Parno et al. [PHGR13].

Equipped with the encoding tool we have defined above, we are ready to construct a QAP-based SNARK scheme. A very high-level overview of the SNARK from QAP construction is provided in Figure 3.8. This diagram gives some intuition, but hides a lot of important details of the scheme.

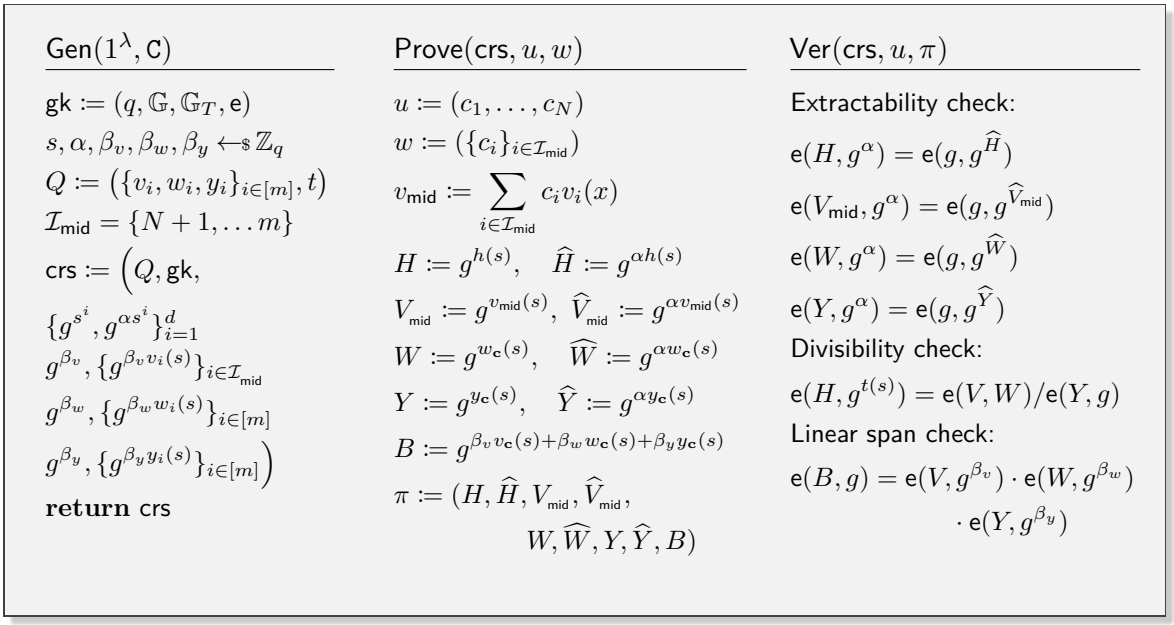


Figure 3.9: SNARK from QAP. A solution to QAP Q of size m and degree d is $v_{\mathbf{c}}(x) = \sum_i c_i v_i(x)$, $w_{\mathbf{c}}(x) = \sum_i c_i w_i(x)$, $y_{\mathbf{c}}(x) = \sum_i c_i y_i(x)$, $h(x) \coloneqq \frac{v_{\mathbf{c}}(x) w_{\mathbf{c}}(x) - y_{\mathbf{c}}(x)}{t(x)}$.

For the sake of the presentation, the description of the protocol is given for a general encoding scheme Enc. In Figure 3.9 there is a SNARK construction for an instantiation based on bilinear group encodings (see Example 3.4.6).

Generation Algorithm $\mathsf{Gen}(1^\lambda, \mathtt{C}) \to (\mathsf{crs}, \mathsf{vrs})$

The setup algorithm Gen takes as input 1^λ and the circuit $\mathtt{C}$ with N input/output values. It generates a QAP Q of size m and degree d over a field $\mathbb{F}$, that verifies $\mathtt{C}$. It defines $\mathcal{I}_{\mathsf{mid}} = \{N+1, \dots m\}$. Then, it runs the setup for an encoding scheme Enc (with secret state sk, or without, $\mathsf{sk} = \perp$). Finally, it samples $\alpha, \beta_v, \beta_w, \beta_y, s \leftarrow \mathbb{F}$ such that $t(s) \neq 0$, and returns $(\mathsf{vrs} = \mathsf{sk}, \mathsf{crs})$ where crs is:

$$\begin{aligned}\mathsf{crs} \coloneqq \Big(Q, &\mathsf{Enc}(1), \mathsf{Enc}(s), \dots, \mathsf{Enc}(s^d),\\ &\mathsf{Enc}(\alpha), \mathsf{Enc}(\alpha s), \dots, \mathsf{Enc}(\alpha s^d),\\ &\left(\mathsf{Enc}(\alpha v_i(s))\right)_{i\in\mathcal{I}_{\mathsf{mid}}}, \left(\mathsf{Enc}(\alpha w_i(s))\right)_{i\in[m]}, \left(\mathsf{Enc}(\alpha y_i(s))\right)_{i\in[m]}\\ &\left(\mathsf{Enc}(\beta_v v_i(s))\right)_{i\in\mathcal{I}_{\mathsf{mid}}}, \left(\mathsf{Enc}(\beta_w w_i(s))\right)_{i\in[m]}, \left(\mathsf{Enc}(\beta_y y_i(s))\right)_{i\in[m]}\Big)\end{aligned} \tag{3.3}$$

Prover $\mathsf{Prove}(\mathsf{crs}, u, w) \to \pi$

The prover algorithm Prove, on input some statement $u \coloneqq (c_1, \dots, c_N)$, computes a witness $w \coloneqq (c_{N+1} \dots c_m)$ and $v_{\mathsf{mid}}(x) = \sum_{i\in I_{\mathsf{mid}}} c_i v_i(x), v_{\mathbf{c}}(x) = \sum_{i\in[m]} c_i v_i(x)$, $w_{\mathbf{c}}(x) = \sum_{i\in[m]} c_i w_i(x), y_{\mathbf{c}}(x) = \sum_{i\in[m]} c_i y_i(x)$ such that :

$$t(x) \text{ divides } p(x) = v_{\mathbf{c}}(x) w_{\mathbf{c}}(x) - y_{\mathbf{c}}(x).$$

Then, it computes the quotient polynomial $h(x)$:

$$h(x) \coloneqq \frac{p(x)}{t(x)}. \tag{3.4}$$

By using the additively homomorphic property of the encoding scheme Enc and the values in the crs, the prover computes encodings of the following polynomial evaluations in s:

$$\begin{aligned} H &\coloneqq \mathsf{Enc}(h(s)), & \widehat{H} &\coloneqq \mathsf{Enc}(\alpha h(s)), \\ V_{\mathsf{mid}} &\coloneqq \mathsf{Enc}(v_{\mathsf{mid}}(s)), & \widehat{V}_{\mathsf{mid}} &\coloneqq \mathsf{Enc}\left(\alpha v_{\mathsf{mid}}(s)\right), \\ W &\coloneqq \mathsf{Enc}(w_{\mathbf{c}}(s)), & \widehat{W} &\coloneqq \mathsf{Enc}\left(\alpha w_{\mathbf{c}}(s)\right), \\ Y &\coloneqq \mathsf{Enc}(y_{\mathbf{c}}(s)), & \widehat{Y} &\coloneqq \mathsf{Enc}\left(\alpha y_{\mathbf{c}}(s)\right), \\ B &\coloneqq \mathsf{Enc}\left(\beta_v v_{\mathbf{c}}(s) + \beta_w w_{\mathbf{c}}(s) + \beta_y y_{\mathbf{c}}(s)\right). \end{aligned} \tag{3.5}$$

Where the polynomial $v_{\mathsf{mid}}(x) \coloneqq \sum_{i \in \mathcal{I}_{\mathsf{mid}}} c_i v_i(x)$. Since the values of c_i, where $i \in [N]$ correspond to the input u (which is also known to the verifier), the verifier can compute the missing part of the full linear combination $v_{\mathbf{c}}(x)$ for $\{v_i(x)\}$ and encode it by itself

$$V \coloneqq \mathsf{Enc}(v_{\mathbf{c}}(s)).$$

The proof π consists of elements $(H, \widehat{H}, V_{\mathsf{mid}}, \widehat{V}_{\mathsf{mid}}, W, \widehat{W}, Y, \widehat{Y}, B)$.

Verifier $\mathsf{Ver}(\mathsf{vrs}, u, \pi) \to 0/1$

Upon receiving a proof π and a statement u, the verifier, uses the quadratic root detection algorithm of the encoding scheme Enc to verify that the proof satisfies:

Extractability terms. $\widehat{H} \stackrel{?}{=} \alpha H, \qquad \widehat{V}_{\mathsf{mid}} \stackrel{?}{=} \alpha V_{\mathsf{mid}}, \qquad \widehat{W} \stackrel{?}{=} \alpha W, \qquad \widehat{Y} \stackrel{?}{=} \alpha Y.$
The above terms are engineered to allow extractability using a knowledge assumption.

Divisibility check. $H \cdot T \stackrel{?}{=} V \cdot W - Y$ where $T = \mathsf{Enc}(t(s))$, $V \coloneqq \mathsf{Enc}(v_{\mathbf{c}}(s))$ and can be computed using crs. This corresponds to the polynomial division constraint.

Linear span check. $B \stackrel{?}{=} \beta_v V + \beta_w W + \beta_y Y$. This check makes sure that the polynomials $v_{\mathbf{c}}(x), w_{\mathbf{c}}(x)$ and $y_{\mathbf{c}}(x)$ are indeed linear combinations of the initial set of polynomials $\{v_i\}_i, \{w_i\}_i, \{y_i\}_i$.

3.4.2.1 Adding Zero-Knowledge

The construction we just described is not zero-knowledge, since the proof terms are not uniformly distributed and may reveal information about the witness, i.e., about the polynomials $v_{\mathbf{c}}(x) = \sum_i c_i v_i(x),\ w_{\mathbf{c}}(x),\ y_{\mathbf{c}}(x),\ h(x)$. To make this proof statistically zero-knowledge, we will randomize the polynomials $v_{\mathbf{c}}(x), w_{\mathbf{c}}(x), y_{\mathbf{c}}(x), h(x)$ by adding a uniformly sampled value, while keeping the divisibility relation between them. The idea is that the prover just uses some random values $\delta_v, \delta_w, \delta_y \in \mathbb{F}$ and performs the following replacements in the proof to randomize its original polynomials from above:

- $v'_{\mathsf{mid}}(x) \coloneqq v_{\mathsf{mid}}(x) + \delta_v t(x)$,

- $w'_{\mathbf{c}}(x) \coloneqq w_{\mathbf{c}}(x) + \delta_w t(x)$,
- $y'_{\mathbf{c}}(x) \coloneqq y_{\mathbf{c}}(x) + \delta_y t(x)$,
- $h'(x) = h(x) + \delta_v w_{\mathbf{c}}(x) + \delta_w v_{\mathbf{c}}(x) + \delta_v \delta_w t^2(x) - \delta_y t(x)$.

By these replacements, the values V_{mid}, W and Y, which contain an encoding of the witness factors, basically become indistinguishable from randomness and thus intuitively they are zero-knowledge. For this modification to be possible, additional terms containing the randomness $\delta_v, \delta_w, \delta_y$ should be added to the crs to enable the prover to mask its proof and the verifier to check it. For a formal proof and other details, we refer the reader to the original work [PHGR13].

Knowledge Soundness. The intuition is that it is hard for the prover, who knows the CRS but not α, to output any pair $(H, \widehat{H})$ where H encodes some value h and $\widehat{H} = \mathsf{Enc}(\alpha h)$ unless the prover knows a representation $h = \sum_{i \in [d]} h_i s^i$ and applies the same linear combination to obtain $\widehat{H}$. Knowledge of exponent assumptions (PKE) introduced in Section 2.3.4 formalize this intuition; it says that for any algorithm that outputs a pair of encoded elements with ratio α, there is an extractor that "watches" the algorithm's computation and outputs the corresponding representation (the coefficients of a linear combination).

We will give an overview of the proof in tree steps, one for each of the three checks in the verification algorithm:

Extractability terms. From the pairs of encodings $(H, \widehat{H}), (V_{\mathsf{mid}}, \widehat{V}_{\mathsf{mid}}), (W, \widehat{W}), (Y, \widehat{Y})$, based on the q-PKE assumption we can extract out coefficients for polynomials $v_{\mathsf{mid}}(x), w_{\mathbf{c}}(x), y_{\mathbf{c}}(x)$, $h(x)$.

Divisibility check. If the check $H \cdot T \stackrel{?}{=} V \cdot W - Y$ where $T = \mathsf{Enc}(t(s))$ verifies, then $h(s)t(s) = v_{\mathbf{c}}(s)w_{\mathbf{c}}(s) - y_{\mathbf{c}}(s)$. If indeed $h(x)t(x) = v_{\mathbf{c}}(x)w_{\mathbf{d}}(x) - y_{\mathbf{c}}(x)$ as polynomials, the soundness of our QAP implies that we have extracted a true proof. Otherwise, $h(x)t(x) - v_{\mathbf{c}}(x)w_{\mathbf{d}}(x) - y_{\mathbf{c}}(x)$ is a nonzero polynomial having s as a root, which allows us to solve a q-type assumption instance (see Section 2.3.2).

Linear span check. In the scheme, the α-terms $\widehat{V}_{\mathsf{mid}} = \mathsf{Enc}(\alpha v_{\mathsf{mid}}), \widehat{W} = \mathsf{Enc}(\alpha w_{\mathbf{c}})$ and $\widehat{Y} = \mathsf{Enc}(\alpha y_{\mathbf{c}})$ are used only to extract representations of the encoded terms with respect to the power basis, and not as a linear combination in the set of polynomials $\{v_i, w_i, y_i\}_{i \in [m]}$. This extraction does not guarantee that the polynomials $v_{\mathsf{mid}}(x), w_{\mathbf{c}}(x), y_{\mathbf{c}}(x)$ lie in the appropriate spans. Therefore, the final proof term B is needed to enforce this. B can only be computed by the prover from the crs by representing $v_{\mathbf{c}}, w_{\mathbf{c}}, y_{\mathbf{c}}$ as a linear combination of corresponding $\{v_i, w_i, y_i\}_{i \in [m]}$. This is then checked in the verification algorithm $B \stackrel{?}{=} \beta_v V + \beta_w W + \beta_y Y$. If this final check passes, but polynomials $v_{\mathbf{c}}, w_{\mathbf{c}}, y_{\mathbf{c}}$ lie outside their proper span, then the one can solve d-power Diffie-Hellman problem (see Section 2.3.2).

3.4.2.2 SNARKs from SSP.

Another notable SNARK that achieves reduced computation complexity for proving evaluation of boolean circuits represented as SSPs is the construction of Danezis et al. [DFGK14]. In this short survey, we will restrict ourself to a sketched description of their scheme prioritizing intuition to rigurosity. We refer the reader to Chapter 4 for a more technical presentation of SNARKs from a similar, but improved framework for SSP.

The key element of Danezis' et al. SNARK is the use of SSP language. The square span program require only a single polynomial $v_{\mathbf{c}}(x)$ to be evaluated for verification (instead of two

for earlier QSPs, and three for QAPs) leading to a simpler and more compact setup, smaller keys, and fewer operations required for proof and verification. The resulting, SSP-based SNARK may be the most compact construction to date. The proof consists of just 4 group elements; they can be verified in just 6 pairings, plus one multiplication for each (non-zero) bit of input, irrespective of the size of the circuit $\mathtt{C} : \{0,1\}^N \to \{0,1\}$.

$\mathsf{Gen}(1^\lambda, \mathtt{C})$

$\mathsf{gk} := (q, \mathbb{G}, \mathbb{G}_T, \mathsf{e})$
$S := (v_0, \ldots, v_m, t)$
$\mathcal{I}_{\mathsf{mid}} := \{N+1, \ldots m\}$
$g \leftarrow\!\$\, \mathbb{G}, \quad s, \alpha, \beta \leftarrow\!\$\, \mathbb{Z}_q$
$\mathsf{crs} := \big(S,\ \mathsf{gk},$
$\{g^{s^i}, g^{\alpha s^i}\}_{i=0}^{d},$
$g^{\beta}, \{g^{\beta v_i(s)}\}_{i \in \mathcal{I}_{\mathsf{mid}}}, g^{\beta t(s)}\big)$
return crs

$\mathsf{Prove}(\mathsf{crs}, u, w)$

$u := (c_1, \ldots, c_N) \in \{0,1\}^N$
$w := (\{c_i\}_{i \in \mathcal{I}_{\mathsf{mid}}})$
$v_{\mathsf{mid}} := \sum_{i \in \mathcal{I}_{\mathsf{mid}}} c_i v_i(x)$
$H := g^{h(s)}, \quad V_{\mathsf{mid}} := g^{v_{\mathsf{mid}}(s)}$
$\widehat{V} := g^{\alpha v_{\mathbf{c}}(s)}, \ B_{\mathsf{mid}} := g^{\beta v_{\mathsf{mid}}(s)}$
$\pi := (H, V_{\mathsf{mid}}, \widehat{V}, B_{\mathsf{mid}})$

$\mathsf{Ver}(\mathsf{crs}, u, \pi)$

$V := g^{\sum_{i \in [N]} c_i v_i(s)} V_{\mathsf{mid}}$
Extractability check:
$\mathsf{e}(V, g^\alpha) = \mathsf{e}(g, \widehat{V})$
Divisibility check:
$\mathsf{e}(H, g^{t(s)}) = \mathsf{e}(V, \widehat{V})/\mathsf{e}(g,g)^{-1}$
Linear span check:
$\mathsf{e}(B_{\mathsf{mid}}, g) = \mathsf{e}(V_{\mathsf{mid}}, g^\beta)$

Figure 3.10: SNARK from SSP. A solution to $S : v_{\mathbf{c}}(x) = \sum_i c_i v_i(x),\ h(x) := \frac{v_{\mathbf{c}}(x)^2 - 1}{t(x)}$.

3.5 SNARK: Construction from LIP

The QAP approach was generalized in [BCI+13] under the concept of *Linear Interactive Proof* (LIP), a form of interactive ZK proofs where security holds under the assumption that the prover is restricted to compute only linear combinations of its inputs.

These proofs can then be turned into (designated-verifier) SNARKs by using an *extractable linear-only* encryption scheme.

3.5.1 Linear-Only Encoding Schemes

An *extractable linear-only* encoding scheme is an encoding scheme where any adversary can output a valid new encoding only if this is a linear combination of some previous encodings that the adversary had as input (intuitively this "limited malleability" of the scheme, will force the prover into the above restriction). At high-level, a linear-only encoding scheme does not allow any other form of homomorphism than linear operations.

Definition 3.5.1 (Extractable Linear-Only, [BCI+13])**.** *An encoding scheme* Enc *satisfies* extractable linear-only *property if for all* PPT *adversaries* $\mathcal{A}$ *there exists an efficient extractor* $\mathcal{E}_{\mathcal{A}}$ *such that, for any sufficiently large* $\lambda \in \mathbb{N}$*, any "benign" auxiliarly input* z *and any plaintext generation algorithm* M *the advantage*

$$\mathsf{Adv}^{\text{ext-lo}}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathcal{E}_{\mathcal{A}}} := \Pr[\mathsf{EXT} - \mathsf{LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathcal{E}_{\mathcal{A}}} = \mathbf{true}\ ] = \mathsf{negl}.$$

where $\mathsf{EXT} - \mathsf{LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathcal{E}_{\mathcal{A}}}$ *is defined as in Figure 3.11.*

$\mathsf{EXT} - \mathsf{LO}_{\mathsf{Enc},\mathsf{M},\mathcal{A},\mathcal{E}_{\mathcal{A}}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$

$(m_1, \ldots, m_d) \leftarrow \mathsf{M}(1^\lambda)$

$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_d))$

$(\mathsf{ct}; a_1, \ldots, a_d, b) \leftarrow (\mathcal{A} \| \mathcal{E}_{\mathcal{A}})(\sigma; z)$

where $\mathsf{ct} = (\mathsf{Enc}(m))$

return $\mathsf{ct} \notin \left\{ \mathsf{Enc}(\sum_{i=1}^{d} a_i m_i + b) \right\}$

Figure 3.11: Game for Extractable Linear-Only.

In order for this definition to be non-trivial, the extractor $\mathcal{E}_{\mathcal{A}}$ has to be *efficient*. Otherwise a naive way of finding such a linear combination $(a_1, \ldots, a_d)$ could be just to run the adversary $\mathcal{A}$, obtain $\mathcal{A}$'s outputs, decrypt them, and then output a zero linear function and hard-code the correct values in the constant term.

A stronger notion of linear-only encoding schemes is *linear-only encryption schemes* that additionally satisfy semantic security in the sense of Definition 2.14. As examples of linear-only encryption schemes, [BCI+13] propose variants of Paillier encryption [Pai99] (as also considered in [GGPR13]) and of ElGamal encryption [ElG85] (in those cases where the plaintext is guaranteed to belong to a polynomial-size set, so that decryption can be done efficiently). These variants are "sparsified" versions of their standard counterparts; concretely, a ciphertext includes not only $\mathsf{Enc}(x)$, but also $\mathsf{Enc}(\alpha x)$, for a secret element α in the message space. (This "sparsification" follows a pattern found in many constructions conjectured to satisfy "knowledge-of-exponent" assumptions).

Non-Adaptive SNARK. In [BCI+13], they start from the notion of *linear-targeted malleability*, weaker than linear-only property, that is closer to the definition template of Boneh et al. [BSW12]. In such a notion, the extractor is replaced by an efficient simulator. Relying on this weaker variant, they are only able to prove the security of their preprocessing SNARKs against non-adaptive choices of statements (and still prove soundness, though not knowledge soundness, if the simulator is allowed to be inefficient, i.e., obtain a SNARG instead of a SNARK). Concretely, the linear-only property rules out any encryption scheme where ciphertexts can be sampled obliviously; instead, the weaker notion does not, and thus allows for shorter ciphertexts.

Definition 3.5.2 (Linear-Targeted Malleability, [BCI+13])**.** *An encoding scheme* Enc *satisfies* linear-targeted malleability *property if for all* PPT *adversaries* $\mathcal{A}$ *and plaintext generation algorithm* M *there exists a* PPT *simulator* Sim *such that, for any sufficiently large* $\lambda \in \mathbb{N}$*, any "benign" auxiliarly input* z *the following two distributions* $\mathcal{D}_0(\lambda), \mathcal{D}_1(\lambda)$ *in Figure 3.12 are computationally inistinguishable.*

Linear Interactive Proof. A linear interactive proof (LIP) is defined similarly to a standard interactive proof [GMR85], except that each message sent by a prover (either an honest or a malicious one) must be a linear function of the previous messages sent by the verifier. The

$(\mathsf{pk}, \mathsf{st}, \{m_i\}, \{\mathsf{Dec}(\mathsf{ct}_j)\}) \leftarrow \mathcal{D}_0(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$

$(\mathsf{st}, m_1, \ldots, m_d) \leftarrow \mathsf{M}(1^\lambda)$

$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_d))$

$\{\mathsf{ct}_j\}_{j=1}^n \leftarrow \mathcal{A}(\sigma; z)$

where $\mathsf{Dec}(\mathsf{ct}_j) \neq \perp$

$(\mathsf{pk}, \mathsf{st}, \{m_i\}, \{d_j\}) \leftarrow \mathcal{D}_1(\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$

$(\mathsf{st}, m_1, \ldots, m_d) \leftarrow \mathsf{M}(1^\lambda)$

$(\vec{a}_1, \cdots \vec{a}_n, \vec{b}) \leftarrow \mathsf{Sim}(\mathsf{pk}; z)$

$d_j \coloneqq \sum_{i=1}^d a_{ji} m_i + b_i, \ \forall\ j \in [n]$

Figure 3.12: Distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ in Linear Targeted Malleability.

SNARK designed by [BCI+13] only makes use of two-message LIPs in which the verifier's message is independent of its input. LIP can be obtained from a Linear PCP, a PCP, for which it is possible to do the verification in such a way that it is sufficient for an honest prover to respond with a certain linear function of the verifier's queries.

Bitansky et al. show first a transformation from any Linear PCP into a two-message LIP with similar parameters. Unlike in the Linear PCP model, if the verifier simply forwards to the LIP prover the queries generated by the Linear PCP verifier, there is no guarantee that the LIP prover will apply the same linear function to each of these queries. Thus, the transformation looses a constant factor in the knowledge error.

SNARK from LIP. Bitansky et al. [BCI+13] construct a publicly-verifiable preprocessing SNARK from LIPs with low-degree verifiers. Note that, if we aim for public verifiability, we cannot use semantically-secure encryption to encode the message of the LIP verifier, because we need to "publicly test" (without decryption) certain properties of the plaintext underlying the prover's response. The idea, implicit in previous publicly-verifiable preprocessing SNARK constructions, is to use linear-only encodings (rather than encryption) that do allow such public tests, while still providing certain one-wayness properties. When using such encodings with a LIP, however, it must be the case that the public tests support evaluating the decision algorithm of the LIP and, moreover, the LIP remains secure despite some "leakage" on the queries. They show that LIPs with low-degree verifiers (which they call algebraic LIPs), combined with appropriate one-way encodings, suffice for this purpose. More concretely, they consider candidate encodings in bilinear groups (Example 3.4.6) under similar knowledge-of-exponent and computational Diffie-Hellman assumptions. These LIP constructions imply new constructions of publicly-verifiable preprocessing SNARKs, one of which can be seen as a simplification of the construction of [Gro10] and the other as a reinterpretation (and slight simplification) of the construction of [GGPR13].

Chapter 4

Post-Quantum SNARK

Lattice-based SNARKs. We have introduced zero-knowledge SNARKs proof systems and shown some constructions that rely on classical number-theoretic assumption. To this day, all known SNARK implementations rely on pre-quantum assumptions and, for this reason, are not expected to withstand cryptanalytic efforts over the next few decades. In this chapter, we introduce a new zk-SNARK that can be instantiated from lattice-based assumptions, and which is thus believed to be post-quantum secure. We weaken the computational assumptions for the SNARK of Danezis et al. [DFGK14] at the cost of one extra group element in the proof, and we furthermore provide a generalization in the spirit of Gennaro et al. [GGPR13] to this scheme. This leads to a generic framework for SNARKs from Square Span Programs (SSPs). We focus on designated-verifier proofs and propose a protocol in which a proof consists of just 5 LWE encodings. We provide a concrete choice of parameters, showing that our construction is practically instantiable.

Contents

Tale two: A Quantum Encounter

Alice had a (quantum) dream!
The previous chapter ended with a beautiful solution for Alice. She found out about the existence of SNARKs and their wonderful functionalities that enable proving the correctness of results for very complicated computations. However, last night, Alice made an incredible encounter, she met a curious cat, who introduced himself as Schrödinger's cat. The cat seemed to know everything about quantum physics, and he worried about the unavoidable advent of the quantum computer in our lives.
The cat explained to Alice that he finds himself in a terrible condition, he is both dead and alive at the same time, being condemned to live in a superposition. In the beginning, Alice could not understand anything from all the occult language the cat used, all his speech seemed such nonsense. She asked for more clarifications.
The cat told her the story of his life, how this guy Schrödinger who was a physicist made him famous throughout a quantum experience imagined in order to explain to the world the behavior of quantum particles at a larger scale.
Schrödinger intended his thought experiment as a discussion with Einstein about his EPR article in 1935. The EPR article [EPR35] highlighted the bizarre nature of quantum superpositions, in which a quantum particle remains in superposition, a combination of multiple states, until it is observed by the external world. When this happens, the superposition collapses into one or another of the possible definite states. Schrödinger described how one could, in principle, create a superposition in a large-scale system by making it dependent on a quantum particle that was in a superposition. He proposed a scenario with his cat in a locked steel chamber, wherein the cat's life or death depended on the state of a radioactive atom.
An area of related study called "observation theory" dictates that when a quantum particle is being watched it can act like a wave. Basically, the universe acts one way when we're looking, and another way when we aren't!
Alice was fascinated by all this story, this seemed so mysterious to her, but she still did not see the point of all that. The cat then mentioned an incredible application of this theory: the quantum computer!
Quantum computers are devices capable of performing computations using quantum bits, or qubits. On a classical computer, the processor uses transistors to perform calculations. Each transistor can be on or off, which correspond to a binary value, either one or zero.
Quantum computers instead use qubits that can actually be both at the same time because of this strange phenomenon called superposition. Exactly like the cat, dead and alive at the same time!
It is possible to create algorithms that run a lot faster on a quantum computer than on a classical computer, due to the unique property of qubits. Those will break many popular cryptographic systems used today. Some number theory assumptions based on factorization are vulnerable to quantum computational power.
Right now, quantum computers aren't worth the trouble and money they take to build, but in the near future, they will be real and functional.
The cat explained to Alice that he has a mission, to raise awareness of the people about the quantum power, the imminence of quantum computer emergence and its consequences.
He asked Alice to promise him that she will take that into consideration and use quantum-secure cryptographic tools in the future. Alice, moved by his strong arguments, reassured the cat that she will be cautious and try to use safe protocols.
This means that she has to start over and search for a post-quantum SNARK. That sounds like a long journey, let us see where it will lead.

4.1 Introduction

So far all known (zk)-SNARKs rely on "classical" pre-quantum assumptions[1] which are expected not to withstand cryptanalytic efforts over the course of the next 10 years. It is an interesting research question, as well as our duty as cryptographers, to provide protocols that can guarantee people's privacy over the next decade. We attempt to make a step forward in this direction by building a designated-verifier zk-SNARK from lattice-based (knowledge) assumptions. Our scheme uses as a main building block encodings that rely on the Learning With Errors (LWE) assumption, initially proposed by Regev in 2005 [Reg05b], and one of the most widespread post-quantum cryptosystem supported by a theoretical proof of security.

SNARGs Based on Lattices. Recently, in two companion papers [BISW17, BISW18], Boneh et al. provided the first designated-verifier SNARGs construction based on lattice assumptions.

The first paper has two main results: an improvement on the LPCP construction in [BCI+13] and a construction of linear-only encryption based on LWE. The second paper presents a different approach where the information-theoretic LPCP is replaced by a LPCP with multiple provers, which is then compiled into a SNARG again via linear-only encryption. The main advantage of this approach is that it reduces the overhead on the prover, achieving what they call *quasi-optimality*[2]. Quasi-optimal succinctness refers to schemes where the argument size is quasilinear in the security parameter. The authors also claim that the stronger notion of knowledge soundness (which leads to SNARKs) can be achieved by replacing the linear-only property with a stronger (extractable) assumption [BCI+13].

Our Contributions. In this chapter, we frame the construction of Danezis et al. [DFGK14] for Square Span Programs in the framework of "encodings" introduced by Gennaro et al. [GGPR13]. We slightly modify the definition of encoding to accommodate for the noisy nature of LWE schemes. This allows us to have a more fine-grained control over the error growth, while keeping previous example encodings still valid instantiations. Furthermore, SSPs are similar to but simpler than Quadratic Span Programs (QSPs) or Quadratic Arithmetic Programs (QAPs) since they use a single series of polynomials, rather than 2 or 3 and they do not have the significant overhead of previous gate and wire checkers.

We use SSPs to build simpler and more efficient designated-verifier zero-knowledge SNARKs for boolean circuit satisfiability (Circ-SAT).

We think our work is complementary to [BISW17, BISW18]. However, there are several reasons why we believe that our approach is preferable:

- **Zero-Knowledge.** The LPCP-based protocols in [BISW17, BISW18] do not investigate the possibility of achieving zero-knowledge. This leaves open the question of whether zk-SNARKs can be effectively instantiated. Considering the LPCP constructed for a QSP satisfiability problem, there is a general transformation to obtain ZK property

[1] We note that the original protocol of Kilian [Kil92] is a zk-SNARK which can be instantiated with a post-quantum assumption since it requires only a collision-resistant hash function – however (even in the best optimized version recently proposed in [BSBHR18]) the protocol does not seem to scale well for even moderately complex computations.

[2] This is the first scheme where the prover does not have to compute a cryptographic group operation for each wire of the circuit, which is instead true e.g., in known pairing-based protocols.

security level	λ	n	$\log \alpha$	$\log q$	$\lvert\pi\rvert$	$\lvert\mathsf{crs}\rvert$	ZK
medium	168	1270	−150	608	0.46 MB	7.13 MB	
	162	1470	−180	736	0.64 MB	8.63 MB	✓
high	244	1400	−150	672	0.56 MB	7.88 MB	
	247	1700	−180	800	0.81 MB	9.37 MB	✓
paranoid	357	1450	−150	800	0.69 MB	9.37 MB	
	347	1900	−180	864	0.98 MB	10.1 MB	✓

Table 4.1: Security estimates for different choices of LWE parameters (circuit size fixed to $d = 2^{15}$), together with the corresponding sizes of the proof π and of the CRS (when using a seeded PRG for its generation).

[BCI+13]. However, in the case of "noisy" encodings, due to possible information leakages in the error term, this transformation cannot be directly applied. Our SNARK construction, being SSP-based, can be made ZK at essentially no cost for either the prover or the verifier (see intuition given in Section 3.4.2.1). Our transformation is different, exploiting special features of SSPs, and yields a zk-SNARK with almost no overhead. Our construction constitutes the first (designated-verifier) zk-SNARK on lattices.

- **Weaker Assumptions.** The linear-only property on encodings introduced in [BCI+13] implies all the security assumptions needed by a SSP-suitable encoding, but the reverse is not known to hold. Our proof of security, therefore, relies on weaker assumptions and, by doing so, "distills" the minimal known assumptions needed to prove security for SSP, and instantiates them with lattices. We study the relations between our knowledge assumption and the (extractable) linear-only assumption in Section 4.5.

- **Simplicity and Efficiency.** While the result in [BISW18] seems asymptotically more efficient than SSP-based approach, we believe that, for many applications, the simplicity and efficiency of the SSP construction will still provide a concrete advantage in practice. We implemented and tested our scheme: we provide some possible concrete parameters for the instantiation of our zk-SNARKs in Table 4.1, whereas more details on the implementation, along with benchmark results can be found in our article [GMNO18].

Technical Challenges. Although conceptually similar to the original proof of security for QSP-based SNARKs, our construction must incorporate some additional modifications in order to overcome the noise growth of the LWE-based homomorphic operations. These challenges do not arise in the line of work of Boneh et al. [BISW17, BISW18] due to the more general (and stronger) assumption of linear-only encoding (see Section 4.5 for details). Additionally, our construction benefits from the optimizations of SSP-based SNARKs [DFGK14].

Instantiating our encoding scheme with a lattice-based scheme like Regev encryption, differs from [GGPR13] and introduces some technicalities, first in the verification step of the protocol, and secondly in the proof of security. Our encoding scheme is additively

homomorphic; however, correctness of the encoding is guaranteed only for a limited number of linear operations because of the error growth in lattice-based encoding schemes. More specifically, to compute a linear combination of N encodings, we need to scale some parameters for correctness to hold. Throughout this chapter, we will consider only encodings where a bounded number of homomorphic "linear" operations is allowed, and make sure that this bound is sufficient to perform verification and to guarantee the existence of a security reduction.

4.2 New Framework for SNARK from SSP

4.2.1 Boolean Circuit Satisfaction Problems

In Section 3.2, we have defined SNARKs for the universal relation. In this section, it will be more convenient to consider SNARKs for boolean circuit satisfaction problems rather than for the universal relation. We will briefly sketch the relevant definitions and differences. We begin by introducing boolean circuit satisfaction problems more formally than already recalled in Section 3.4.

Definition 4.2.1 (Boolean Circ-SAT)**.** *The Boolean Circuit Satisfaction Problem of a boolean circuit* $\mathcal{C}\colon \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$ *is the relation*

$$\mathcal{R}_{\mathcal{C}} = \{(x,w) \in \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} : \mathcal{C}(x,w) = 1\}.$$

Universal Circuit. Universal circuits allow using a single program for all n' gate circuits at the cost of $n = n' \cdot 19 \log n'$ (see [Val76]) and enables constructions of adaptively-sound SNARKs.

Remark that in the previous definition, we have ℓ_u public inputs and ℓ_w private inputs for a circuit $\mathtt{C}$. This makes $\mathtt{C}$ compatible with universal circuits $C_{\mathcal{U}} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$, that take as input an ℓ_u-bit description of a freely chosen circuit and its public input $(\mathtt{C}, u)$ and an ℓ_w-bit value w, and return 1 if and only if $\mathtt{C}(u,w) = 1$.

Along the lines of [DFGK14], we consider the "public" inputs from the point of view of the prover. For an outsourced computation, they might include both the inputs sent by the clients and the outputs returned by the server performing the computation. For Circ-SAT, they may provide a partial instantiation of the problem or parts of its solution.

4.2.2 Square Span Programs

We characterize NP as Square Span Programs (SSPs) over some field $\mathbb{F}$ of order p (See Section 3.4.1). For completeness, we state here the formal definition with notations adapted for this chapter:

Definition 4.2.2 (SSP)**.** *A Square Span Program (SSP) over the field* $\mathbb{F}$ *is a tuple consisting of* $m+1$ *polynomials* $v_0(x), \ldots, v_m(x) \in \mathbb{F}[x]$ *and a target polynomial* $t(x)$ *such that* $\deg(v_i(x)) \leqslant \deg(t(x))\ \forall i$.

A square span program ssp *accepts an input* $a_1, \ldots, a_{\ell_u} \in \{0,1\}$ *if and only if there exist* $a_{\ell_u+1}, \ldots, a_m \in \{0,1\}$ *satisfying:*

$$t(x) \text{ divides } \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x)\right)^2 - 1.$$

We say that ssp verifies a boolean circuit $\mathtt{C} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$ if it accepts exactly those inputs $(a_1, \ldots, a_{\ell_u}) \in \{0,1\}^{\ell_u}$, satisfying $\mathtt{C}(a_1, \ldots a_{\ell_u}, w) = 1$.

A square span program ssp has size m and degree $d = \deg(t(x))$.

SSP Generation. We consider the uniform probabilistic algorithm SSP that, on input a boolean circuit $\mathtt{C}$ of m wires and n gates, chooses a field $\mathbb{F}$, with $|\mathbb{F}| \geq \max(n, 8)$, and samples $d = m+n$ random elements $r_1, \ldots, r_d \in \mathbb{F}$ to define the target polynomial $t(x) = \prod_{i=1}^{d}(x - r_i)$, together with the set of polynomials $\{v_0(x), \ldots, v_m(x)\}$ composing the SSP corresponding to $\mathtt{C}$: $(v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathtt{C})$.

4.2.3 Encoding Schemes

Encoding schemes for SNARKs were initially introduced in [GGPR13]. Here, we present a variant of their definition (see Section 3.4.1.1). Our new definition accommodates for encodings with noise.

Definition 4.2.3 (Encoding Scheme)**.** *An encoding scheme* Enc *over a field* $\mathbb{F}$ *is composed of the following algorithms:*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$ a key generation algorithm that takes as input some security parameter in unary 1^λ and outputs some secret state sk together with some public information pk. To ease notation, we are going to assume the message space is always part of the public information and that pk can be derived from sk.

$S \leftarrow \mathsf{Enc}(a)$ a randomized encoding algorithm mapping a field element a to some encoding space S, such that $\{\{\mathsf{Enc}(a)\} : a \in \mathbb{F}\}$ partitions S, where $\{\mathsf{Enc}(a)\}$ denotes the set of the possible evaluations of the algorithm Enc on a.

Depending on the encoding algorithm, Enc will require either the public information pk generated from K or the secret state sk. For our application, it will be the case of sk. To ease notation, we will omit this additional argument.

The above algorithms must satisfy the following properties:

- ***d*-linearly homomorphic:** there exists a poly algorithm Eval that, given as input the public parameters pk, a vector of encodings $(\mathsf{Enc}(a_1), \ldots, \mathsf{Enc}(a_d))$, and coefficients $\mathbf{c} = (c_1, \ldots, c_d) \in \mathbb{F}^d$, outputs a valid encoding of $\mathbf{a} \cdot \mathbf{c}$ where $\mathbf{a} = (a_1, \ldots a_d)$ with probability overwhelming in λ.

- ***quadratic root detection:*** there exists an efficient algorithm that, given some parameter δ (either pk or sk), $\mathsf{Enc}(a_0), \ldots, \mathsf{Enc}(a_t)$, and the quadratic polynomial $\mathsf{pp} \in \mathbb{F}[x_0, \ldots, x_t]$, can distinguish if $\mathsf{pp}(a_1, \ldots, a_t) = 0$. With a slight abuse of notation, we will adopt the writing $\mathsf{pp}(\mathsf{ct}_0, \ldots, \mathsf{ct}_t) = 0$ to denote the quadratic root detection algorithm with inputs δ, $\mathsf{ct}_0, \ldots, \mathsf{ct}_t$, and pp.

- ***image verification:*** there exists an efficiently computable algorithm $\in$ that, given as input some parameter δ (again, either pk or sk), can distinguish if an element c is a correct encoding of a field element.

Our specific instantiation of the encoding scheme presents some slight differences with [GGPR13]. In fact, we can allow only for a limited number of homomorphic operations

because of the error growth in lattice-based encoding schemes. We note that this modification does not invalidate previous constructions. Sometimes, in order to ease notation, we will employ the writing $\mathsf{ct} \coloneqq \mathsf{Eval}\left(\mathsf{Enc}\left(a_i\right)_i, \mathbf{c}\right) = \mathsf{Enc}\left(t\right)$, meaning that ct is a valid encoding of $t = \sum a_i c_i$, that is $\mathsf{ct} \in \{\mathsf{Enc}\left(t\right)\}$. It will be clear from the context (and the use of the symbol for assignment instead of that for sampling) that the randomized encoding algorithm is not actually invoked.

Decoding Algorithm. When using a homomorphic encryption scheme in order to instantiate an encoding scheme, we simply define the *decoding algorithm* Dec as the decryption procedure of the scheme. More specifically, since we study encoding schemes derived from encryption functions, quadratic root detection and image verification for designated-verifiers are trivially obtained by using the decryption procedure Dec.

Remark that in the bilinear group instantiation, the encodings are one-way, and no such a decoding function can be defined without breaking the discrete logarithm.

4.2.4 Improved SNARK Framework from SSP

We will use the formal definition introduced in Chapter 3, Definition 3.2.2 for Universal SNARKs, with the adaptation for Circ-SAT relations (i.e., the generation algorithm takes as input a circuit $\mathtt{C}$).

Our framework uses the SSP characterisation of NP and an encoding scheme Enc as described above.

Due to their conceptual simplicity, SSPs offer several advantages over previous constructions for binary circuits. Their reduced number of constraints lead to smaller programs, and to lower sizes and degrees for the polynomials required to represent them, which in turn reduce the computation complexity required in SNARK schemes. Notably, their simpler "square" form requires only a single polynomial to be evaluated for verification (instead of two for earlier QSPs, and three for QAP) leading to a simpler and more compact setup, smaller crs, and fewer operations required for proof and verification.

Our SNARK framework is also different from the SSP-based construction of Danezis et al. [DFGK14] in the sense that the soundness of the resulting SNARK scheme relies on only one hardness assumption (excepting the extractability one) instead of two assumptions in [DFGK14]. We make a tradeoff between the number of terms in the proof π and the assumptions we need in the security reduction, obtaining a SNARK consisting of five encodings instead of four in [DFGK14].

Our proof of security, therefore, relies on fewer assumptions and, by doing so, "distills" the minimal known assumptions needed to prove security of SNARKs from SSP, and allows to latter instantiate them with lattices.

We describe our framework for general encodings in a very simplified way in Figure 4.1. We remark the difference from the SNARK of [DFGK14] (see Figure 3.10) in the extra proof term $\widehat{H}$ and the additional extractability check in the verification corresponding to this term $\widehat{H} = \alpha H$.

In Section 4.4 we will add zero-knowledge and some other technicalities related to our concrete instantiation based on lattice encodings. The generic framework described here can either accommodate publicly verifiable SNARKs or designated verifiable SNARKs depending on the choice of encoding.

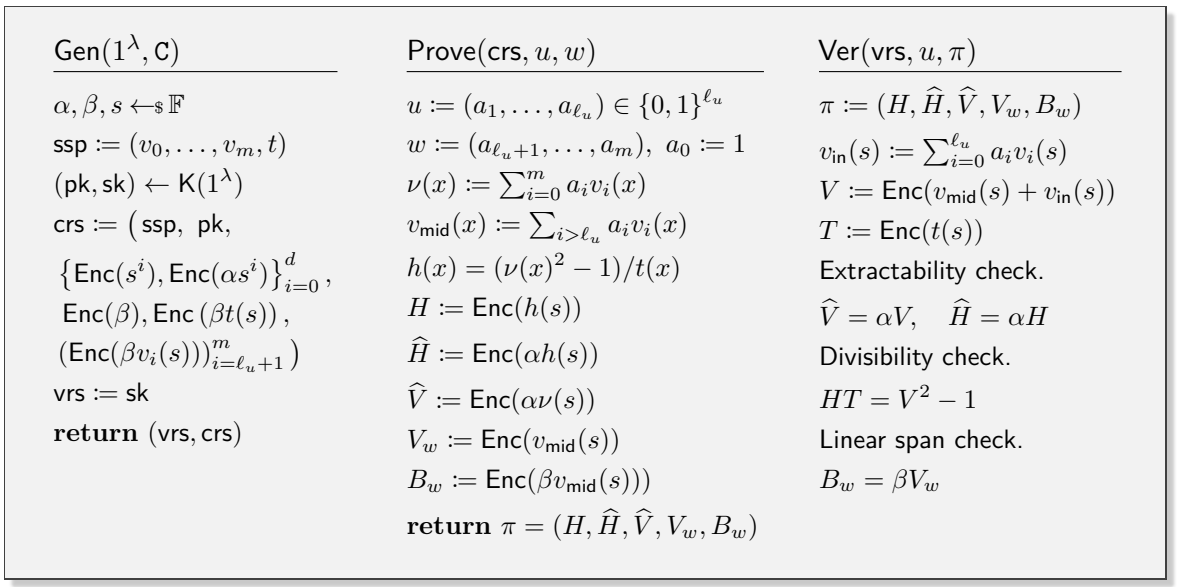


Figure 4.1: Framework for SNARK from SSP.

4.3 An Encoding Scheme Based on LWE

In this section, we describe a possible encoding scheme based on learning with errors (LWE) (see Assumption 2.3.10) that will be used as a building block for our SNARK post-quantum scheme.

Lattice-Based Encoding Scheme. We propose an encoding scheme Enc that consists of three algorithms as depicted in **??**. This is a slight variation of the classical LWE cryptosystem initially presented by Regev [Reg05b] (see Construction 2.4.4) and later extended in [BV11a]. The encoding scheme Enc is described by parameters $\Gamma := (p, q, n, \alpha)$, with $q, n, p \in \mathbb{N}$ such that $\gcd(p, q) = 1$, and $0 < \alpha < 1$. We consider the associated error distribution χ_σ over $\mathbb{Z}$, a discrete Gaussian of width $\sigma = q\alpha$. Our construction is an extension of the one presented in [BV11a].

We assume the existence of a deterministic algorithm Pgen that, given as input the security parameter in unary 1^λ, outputs an LWE encoding description Γ. The choice of using a *deterministic* parameter generation Pgen was already argued by Bellare et al. [BFS16]. The main advantage of this choice is that every entity can (re)compute the description for a given security parameter and that no single party needs to be trusted with generating the encoding parameters. Moreover, it is often the case that real-world encodings have fixed parameters for some well-known values of λ. For the sake of simplicity, we define our encoding scheme with a LWE encoding description Γ and assume that the security parameter λ can be derived from Γ.

Roughly speaking, the public information is constituted by the LWE parameters Γ and an encoding of m is simply an LWE encryption of m. The LWE secret key constitutes the secret state of the encoding scheme.

Basic Properties. We say that the encoding scheme is (statistically) *correct* if all valid

$\mathsf{K}(1^\lambda)$	$\mathsf{Enc}(\mathbf{s}, m)$	$\mathsf{Dec}(\mathbf{s}, (\mathbf{c}_0, c_1))$
$\Gamma := (p, q, n, \alpha) := \mathsf{Pgen}(1^\lambda)$	$\Gamma := (p, q, n, \alpha) := \mathsf{Pgen}(1^\lambda)$	$\Gamma := (p, q, n, \alpha) := \mathsf{Pgen}(1^\lambda)$
$\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$	$\mathbf{a} \leftarrow\!\!\$\, \mathbb{Z}_q^n$	**return** $(\mathbf{c}_0 \cdot \mathbf{s} + c_1) \pmod p$
return $(\Gamma,\ \mathbf{s})$	$\sigma := q\alpha;\quad e \leftarrow \chi_\sigma$	
	return $(-\mathbf{a},\ \mathbf{a} \cdot \mathbf{s} + pe + m)$	

Figure 4.2: An encoding scheme based on LWE.

encodings are decoded successfully (with overwhelming probability).

Definition 4.3.1 (Corectness)**.** *An encoding scheme* Enc *is* correct *if, for any* $\mathbf{s} \leftarrow \mathsf{K}(1^\lambda)$ *and* $m \in \mathbb{Z}_p$*,* $\Pr[\mathsf{Dec}(\mathbf{s}, \mathsf{Enc}(\mathbf{s}, m)) \neq m] = \mathsf{negl}$.

It is easy to see that the encoding Enc defined in **??** satisfies correctness and all the properties of Definition 4.2.3:

correctness. Let $\mathsf{ct} = (-\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + pe + m)$ be an encoding. Then ct is a valid encoding of a message $m \in \mathbb{Z}_p$ if $e < \frac{q}{2p}$.

d-linearly homomorphicity. Given a vector of d encodings $\vec{\mathsf{ct}} \in \mathbb{Z}_q^{d \times (n+1)}$ and a vector of coefficients $\mathbf{c} \in \mathbb{Z}_p^d$, the homomorphic evaluation algorithm is defined as follows: $\mathsf{Eval}\,(\vec{\mathsf{ct}}, \mathbf{c}) := \mathbf{c} \cdot \vec{\mathsf{ct}}$.

quadratic root detection. The algorithm for quadratic root detection is straightforward using Dec: decrypt the message and evaluate the polynomial, testing if it is equal to 0.

image verification. Using the decryption algorithm Dec, and provided with the secret key (i.e., $\delta := \mathbf{s}$), we can implement image verification (algorithm $\in$).

Fresh Encodings. We say that an encoding is *fresh* if it is generated through the Enc algorithm. Otherwise, we consider the encoding is *stale*.

4.3.1 Technical Challenges

Noise growth. During the homomorphic evaluation, the noise grows as a result of the operations which are performed on the encodings. Consequently, in order to ensure that the output of Eval is a valid encoding of the expected result, we need to start with a sufficiently small noise in each of the initial encodings.

In order to bound the size of the noise, we first need a basic theorem on the tail bound of discrete Gaussian distributions due to Banaszczyk [Ban95]:

Lemma 4.3.2 ([Ban95, Lemma 2.4])**.** *For any* $\sigma, T \in \mathbb{R}^+$ *and* $\mathbf{a} \in \mathbb{R}^n$*:*

$$\Pr[\mathbf{x} \leftarrow \chi_\sigma^n \ :\ |\mathbf{x} \cdot \mathbf{a}| \geq T\sigma \,\|\mathbf{a}\|] < 2\,\exp(-\pi T^2). \tag{4.1}$$

At this point, this corollary follows:

Corollary 4.3.3. *Let* $\mathbf{s} \leftarrow_{\$} \mathbb{Z}_q^n$ *be a secret key and* $\mathbf{m} = (m_0, \ldots, m_{d-1}) \in \mathbb{Z}_p^d$ *be a vector of messages. Let* $\vec{\mathsf{ct}}$ *be a vector of* d *fresh encodings so that* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathbf{s}, m_i)$*, and* $\mathbf{c} \in \mathbb{Z}_p^d$ *be a vector of coefficients. If* $q > 2p^2\sigma\sqrt{\frac{\kappa d}{\pi}}$*, then* $\mathsf{Eval}(\mathbf{c}, \vec{\mathsf{ct}})$ *outputs a valid encoding of* $\mathbf{m} \cdot \mathbf{c}$ *under the secret key* $\mathbf{s}$ *with probability overwhelming in* κ*.*

Proof. The fact that the message part is $\mathbf{m} \cdot \mathbf{c}$ is trivially true by simple homomorphic linear operations on the encodings. Then the final encoding is valid if the error does not grow too much during these operations. Let $\mathbf{e} \in \mathbb{Z}_p^d$ be the vector of all the error terms in the d encodings, and let $T = \sqrt{\kappa/\pi}$. Then by Lemma 4.3.2 we have:

$$\Pr\left[\mathbf{e} \leftarrow \chi_\sigma^d \ : \ |\mathbf{e} \cdot \mathbf{c}| \geq \sqrt{\frac{\kappa}{\pi}}\sigma \, \|\mathbf{c}\,\|\right] < 2\,\exp(-\kappa).$$

For correctness, we need the absolute value of the final noise to be less than $q/2p$ (cf. Section 4.3). Since it holds that $\forall \mathbf{c} \in \mathbb{Z}_p^d$, $\|\mathbf{c}\| \leq p\sqrt{d}$, we can state that correctness holds if:

$$\sqrt{\frac{\kappa}{\pi}}\sigma p\sqrt{d} < \frac{q}{2p}$$

which gives $q > 2p^2\sigma\sqrt{\frac{\kappa d}{\pi}}$. □

Smudging. When computing a linear combination of encodings, the distribution of the error term in the final encoding does not result in a correctly distributed fresh encoding. The resulting error distribution depends on the coefficients used for the linear combination, and despite correctness of the decryption still holds, the error could reveal more than just the plaintext. We combine homomorphic evaluation with a technique called *smudging* [AJL+12], which "smudges out" any difference in the distribution that is due to the coefficients of the linear combination, thus hiding any potential information leak. This technique has also been called "noise flooding" in the past [BPR12].

Lemma 4.3.4 (Noise Smudging, [Gen09])**.** *Let* $B_1 = B_1(\kappa)$ *and* $B_2 = B_2(\kappa)$ *be positive integers. Let* $x \in [-B_1, B_1]$ *be a fixed integer and* $y \leftarrow_{\$} [-B_2, B_2]$*. Then the distribution of* y *is statistically indistinguishable from that of* $y + x$*, as long as* $B_1/B_2 = \mathsf{negl}[\kappa]$*.*

Proof. Let Δ denote the statistical distance between the two distributions. By its definition:

$$\Delta = \frac{1}{2}\sum_{v=-(B_1+B_2)}^{B_1+B_2} |\Pr[y = v] - \Pr[y = v - x]| = \frac{1}{2}\left(\sum_{v=-(B_1+B_2)}^{-B_2}\frac{1}{B_2} + \sum_{v=B_2}^{B_1+B_2}\frac{1}{B_2}\right) = \frac{B_1}{B_2}.$$

The result follows immediately. □

In order to preserve the correctness of the encoding scheme while allowing linear evaluations, we need once again q to be large enough to accommodate for the flooding noise. In particular, q will have to be at least superpolynomial in the statistical security parameter κ.

Corollary 4.3.5. *Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be a secret key and* $\mathbf{m} = (m_1, \ldots, m_d) \in \mathbb{Z}_p^d$ *be a vector of messages. Let* $\vec{\mathsf{ct}}$ *be a vector of* d *encodings so that* ct_i *is a valid encoding of* m_i*, and* $\mathbf{c} \in \mathbb{Z}_p^d$ *be a vector of coefficients. Let* e_{Eval} *be the noise in the encoding output by* $\mathsf{Eval}(\vec{\mathsf{ct}}, \mathbf{c})$ *and* B_{Eval} *a*

bound on its absolute value. Finally, let $B_{sm} = 2^{\kappa} B_{\mathsf{Eval}}$*, and* $e_{sm} \leftarrow\!\!\$\ [-B_{sm}, B_{sm}]$*. Then the statistical distance between the distribution of* e_{sm} *and that of* $e_{sm} + e_{\mathsf{Eval}}$ *is* $2^{-\kappa}$*. Moreover, if* $q > 2p\, B_{\mathsf{Eval}}\,(2^{\kappa}+1)$ *then the result of* $\mathsf{Eval}\,(\vec{\mathsf{ct}}, \mathbf{c}\,) + \left(\vec{0}, e_{sm}\right)$ *is a valid encoding of* $\mathbf{m}\cdot\mathbf{c}$ *under the secret key* $\mathbf{s}$*.*

Proof. The claim on the statistical distance follows immediately from Lemma 4.3.4 and the fact that the message part is $\mathbf{m}\cdot\mathbf{c}$ is true by simple homomorphic linear operations on the encodings. In order to ensure that the final result is a valid encoding, we need to make sure that the error in this output encoding remains smaller than $q/2p$. The final error is upper bounded by $B_{\mathsf{Eval}} + B_{sm}$, so we have

$$B_{\mathsf{Eval}} + B_{sm} < \frac{q}{2p} \implies B_{\mathsf{Eval}} + 2^{\kappa} B_{\mathsf{Eval}} < \frac{q}{2p} \implies q > 2p\, B_{\mathsf{Eval}}\,(2^{\kappa}+1)\,.$$

□

Error Testing. By making non-blackbox use of our LWE encoding scheme, it is possible to define an implementation of the function Test-error in order to guarantee the existence of a security reduction from adversarially-generated proofs. In fact, it is not sufficient to show that a series of homomorphic operations over a forged proof can break one of the assumptions. We must also guarantee that these manipulations do not alter the correctness of the encoded value. In the specific case of LWE encodings, it is sufficient to use the secret key, recover the error, and enforce an upper bound on its norm. A possible implementation of Test-error is displayed in Figure 4.3.

Other Requirements for Security Reduction. The following lemma will be needed later during the security proof. It essentially defines the conditions under which we can take an encoding, add a smudging term to its noise, sum it with the output of an execution of Eval and finally multiply the result by an element in $\mathbb{Z}_p$.

Lemma 4.3.6 (For reduction)**.** *Let* $\mathbf{s}$*,* $\vec{\mathsf{ct}}$*,* $\mathbf{c}$*,* e_{Eval}*,* B_{Eval} *be as in Corollary 4.3.5, and let* $\mathsf{ct}' = (-\mathbf{a}', \mathbf{s}\cdot\mathbf{a}' + pe' + m')$ *be a valid encoding of a message* $m' \in \mathbb{Z}_p$ *with noise* e' *bounded by* B_e*. Let* $B_{sm} = 2^{\kappa} B_e$ *and* $e_{sm} \leftarrow\!\!\$\ [-B_{sm}, B_{sm}]$ *be a "smudging noise". Then, if* $q > 2p^2\left(\left(2^{\kappa}+1\right) B_e + B_{\mathsf{Eval}}\right)$*, it is possible to add the smudging term* e_{sm} *to* ct'*, sum the result with the output of* $\mathsf{Eval}\,(\vec{\mathsf{ct}}, \mathbf{c}\,)$*, multiply the outcome by a coefficient* k *bounded by* p*, and obtain a valid encoding of* $k\,(\mathbf{m}\cdot\mathbf{c} + m')$*.*

Proof. The correctness of the message part comes immediately from performing homomorphic linear operations on encodings, and the final output is valid if the noise remains below a certain threshold. After adding the smudging term and performing the sum, the noise term is at most $B_e + B_{sm} + B_{\mathsf{Eval}} = \left(2^{\kappa}+1\right) B_e + B_{\mathsf{Eval}}$. After the multiplication by a coefficient bounded by p, it is at most $p\left(\left(2^{\kappa}+1\right) B_e + B_{\mathsf{Eval}}\right)$. Thus, the encoding is valid if:

$$p\left(\left(2^{\kappa}+1\right) B_e + B_{\mathsf{Eval}}\right) < \frac{q}{2p}, \tag{4.2}$$

which immediately gives the result. □

Conditions on the Modulus q**.** Corollaries 4.3.3 and 4.3.5 and Lemma 4.3.6 give the conditions that the modulus q has to respect in order to allow for all the necessary computations. In

$\mathsf{Test\text{-}error}(\mathbf{s}, (\mathbf{c}_0, c_1))$

$\Gamma := (p, q, n, \alpha) := \mathsf{Pgen}(1^\lambda)$
$e' := (\mathbf{c}_0 \cdot \mathbf{s} + c_1) \ / \ p$
Check if *Equation* (4.3) holds.

Figure 4.3: The error testing procedure.

particular, Corollary 4.3.3 gives the condition to be able to homomorphically evaluate a linear combination of fresh encodings through the algorithm Eval; Corollary 4.3.5 gives the condition to be able to add a smudging noise to the result of such an evaluation; Lemma 4.3.6 gives a condition that will have to be satisfied in the security reduction. They are ordered from the least stringent to the most stringent, so the condition that must be satisfied in the end is the one given by Lemma 4.3.6:

$$q > 2p^2 \left((2^\kappa + 1) B_e + B_{\mathsf{Eval}} \right) \tag{4.3}$$

Practical Considerations. A single encoded value has size $(n+1)\log q = \widetilde{O}(\lambda)$. Therefore, as long as the prover sends a constant number of encodings, the proof is guaranteed to be (quasi) succinct. As a matter of fact, we can generate the random vector $\mathbf{a}$ that composes the first term of the encoding by extending the output of a seeded PRG. This has been proven secure in the random oracle model [Gal13].

For a more extensive analysis of the asymptotic complexity, as well as the concrete efficiency estimates, we direct the reader towards Section 4.7.

Leftover Hash Lemma (LHL). We now recall the definition of min-entropy, and the famous "leftover hash lemma" introduced by Håstad et al. [HILL99].

Definition 4.3.7 (Min-entropy)**.** *The min-entropy of a random variable X is defined as*

$$H_\infty(X) = -\log\left(\max_x \Pr[X = x]\right)$$

Lemma 4.3.8 (Leftover hash lemma)**.** *Assume a family of functions $\left\{\mathcal{H}_x : \{0,1\}^n \to \{0,1\}^\ell\right\}_{x \in X}$ is* universal*, i.e., $\forall a \neq b \in \{0,1\}^n$,*

$$\Pr_{x \in X}[\mathcal{H}_x(a) = \mathcal{H}_x(b)] = 2^{-\ell}.$$

Then, for any random variable Y:

$$\Delta\left((X, \mathcal{H}_X(Y)), (X, U_\ell)\right) \leq \frac{1}{2}\sqrt{2^{-H_\infty(Y)} \cdot 2^\ell},$$

where $U_\ell \leftarrow\!\!\$\ \{0,1\}^\ell$.

Zero-Knowledge. We now present a version of the LHL that will be useful later in this chapter, when proving the zero-knowledge property of our construction. In a nutshell, it says that a random linear combination of the columns of a matrix is statistically close to a uniformly random vector, for some particular choice of coefficients.

Lemma 4.3.9 ("Specialized" leftover hash lemma)**.** *Let n, p, q, d be non-negative integers. Let $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times d}$, and $\vec{r} \leftarrow_{\$} \mathbb{Z}_p^d$. Then we have*

$$\Delta\left((\mathbf{A}, \mathbf{A}\vec{r}\,), (\mathbf{A}, \vec{u}\,)\right) \leq \frac{1}{2}\sqrt{p^{-d} \cdot q^n},$$

where $\mathbf{A}\vec{r}$ is computed modulo q, and $\vec{u} \leftarrow_{\$} \mathbb{Z}_q^n$.

Proof. For the vector $\vec{r}$, we have that $H_\infty(\vec{r}\,) = d \log p$. Then the proof is immediate from Lemma 4.3.8:

$$\Delta\left((\mathbf{A}, \mathbf{A}\vec{r}\,), (\mathbf{A}, \vec{u})\right) \leq \frac{1}{2}\sqrt{2^{-d\log p} \cdot q^n} = \frac{1}{2}\sqrt{p^{-d} \cdot q^n}.$$

□

4.4 Post-Quantum Designated-Verifier zk-SNARK

Let Enc be an encoding scheme (Definition 4.2.3). We can, for example, consider the concrete instantiation presented in Figure 4.2. Let $\mathtt{C}$ be some circuit taking as input a ℓ_u-bit string and outputting 0 or 1. Let $\ell := \ell_u + \ell_w$, where ℓ_u is the length of the "public" input, and ℓ_w the length of the private input. The value m corresponds to the number of wires in $\mathtt{C}$ and n to the number of fan-in 2 gates. Let $d := m + n$. We construct a zk-SNARK scheme for any relation $\mathcal{R}_{\mathtt{C}}$ on pairs $(u, w) \in \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w}$ that can be computed by a polynomial size circuit $\mathtt{C}$ with m wires and n gates.

Designated Verifiability. As previously mentioned in Section 3.2, we distinguish two types of arguments of knowledge: *publicly verifiable* ones, where the verification algorithm takes as input only common reference string crs, and designated-verifier ones, where the verifier Ver takes as input together with the crs some additional private verification key vrs.

In the case of designated-verifier proofs, the proof can be verified only by the verifier Ver knowing the secret information vrs. We remark that due to the instantiation of our encodings with an encryption scheme, the resulting scheme inherits the need of the secret state sk for the proof verification, and consequently is limited to designated verifiers.

Our protocol is formally depicted in Figure 4.4. Differences from publicly-verifiable pairing-based construction are highlighted.

CRS Generation. The setup algorithm G takes as input some security parameter λ in unary form and the circuit $\mathtt{C} : \{0,1\}^{\ell_u} \times \{0,1\}^{\ell_w} \to \{0,1\}$. It generates a square span program of degree $d = m + n$ over a field $\mathbb{F}$, of size $|\mathbb{F}| \geq d$ that verifies $\mathtt{C}$ by running:

$$\mathsf{ssp} := (v_0(x), \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathtt{C})$$

Then, it runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$ for an encoding scheme Enc. Finally, it samples $\alpha, \beta, s \leftarrow \mathbb{F}$ such that $t(s) \neq 0$, and returns the CRS:

$$\begin{aligned} \mathsf{crs} := \Big(\mathsf{ssp},\ \mathsf{pk},\ & \mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^d), \\ & \mathsf{Enc}(\alpha), \mathsf{Enc}(\alpha s), \ldots, \mathsf{Enc}(\alpha s^d), \\ & \mathsf{Enc}(\beta t(s)), (\mathsf{Enc}(\beta v_i(s)))_{i=\ell_u+1}^{m} \Big) \end{aligned} \tag{4.4}$$

The error for each of these encodings has to be chosen carefully. In a nutshell, we need to intentionally increase the magnitude of the noise in some encodings, in order to mimic

$\mathsf{Gen}(1^\lambda, \mathsf{C})$

$\alpha, \beta, s \leftarrow_\$ \mathbb{F}$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$(v_0, \ldots, v_m, t) \leftarrow \mathsf{SSP}(\mathsf{C})$
// Compute crs as per Eq. 4.4
$\mathsf{vrs} := (\mathsf{sk}, s, \alpha, \beta)$
return $(\mathsf{vrs}, \mathsf{crs})$

$\mathsf{Ver}(\mathsf{vrs}, u, \pi)$

$u := (a_1, a_2, \ldots a_{\ell_u})$, $a_0 := 1$
$w_s := \mathsf{Dec}(V_w)$; $b_s := \mathsf{Dec}(B_w)$
$h_s := \mathsf{Dec}(H)$; $\widehat{h}_s := \mathsf{Dec}(\widehat{H})$
$\widehat{v}_s := \mathsf{Dec}(\widehat{V})$; $t_s := t(s)$
$v_s := \sum_{i=0}^{\ell_u} a_i v_i(s) + w_s$
Make the checks:
$eq - pke$, $eq - div$, $eq - lin$
Test-error(sk, B_w)

$\mathsf{Prove}(\mathsf{crs}, u, w)$

$\mathsf{ssp} := (v_0(x), \ldots, v_m(x), t(x))$
$u := (a_1, \ldots, a_{\ell_u}) \in \{0,1\}^{\ell_u}$, $a_0 := 1$
$w := (a_{\ell_u+1}, \ldots, a_m)$, $\gamma \leftarrow_\$ \mathbb{F}$
$\nu(x) := v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$
$v_{\mathsf{mid}}(x) := \sum_{i>\ell_u}^{m} a_i v_i(x) + \gamma t(x)$
$h(x) = (\nu(x)^2 - 1)/t(x)$
// Compute the proof terms as per Equation (4.6)
$H := \mathsf{Eval}((\mathsf{Enc}(s^i))_i^d,\ (h_i)_i^d) = \mathsf{Enc}(h(s))$
$\widehat{H} := \mathsf{Eval}((\mathsf{Enc}(\alpha s^i))_i^d,\ (h_i)_i^d) = \mathsf{Enc}(\alpha h(s))$
$\widehat{V} := \mathsf{Eval}((\mathsf{Enc}(\alpha s^i))_i^d,\ (\nu_i)_i^d) = \mathsf{Enc}(\alpha \nu(s))$
$B_w := \mathsf{Eval}((\mathsf{Enc}(\beta v_i(s)))\|(\mathsf{Enc}(\beta t(s))),\ (a_i)\|(\gamma))$
$V_w := \mathsf{Eval}((\mathsf{Enc}(s^i))_i^d,\ (v_{\mathsf{mid}\,i})_i^d) = \mathsf{Enc}(v_{\mathsf{mid}}(s))$
Apply smudging on $H, \widehat{H}, \widehat{V}, B_w, V_w$
return $\pi = (H, \widehat{H}, \widehat{V}, V_w, B_w)$

Figure 4.4: Our zk-SNARK protocol Π. New specific steps proper to our encodings are highlighted: designated-verifier in red and **lattice-based** instantiation in green.

its distribution in the simulated CRS provided to the adversary in the security reduction. Failing to do so results in the adversary's ability to distinguish between a CRS generated by K algorithm and a simulated one. We defer further analysis on this point to Section 4.7. The verification string vrs consists of the secret key sk of the encoding scheme, and the secrets s, α, β (we implicitly include the crs in vrs).

Prover. The prover algorithm, on input some statement $u := (a_1, \ldots, a_{\ell_u})$ and a witness $w := (a_{\ell_u+1}, \ldots, a_m)$ such that $(u\|w) = (a_1, \ldots, a_m)$ is a satisfying assignment for the circuit C is able to compute the solution of the ssp. The $(a_i)_i$ are such that $t(x)$ divides $(v_0(x) + \sum_{i=1}^{m} a_i v_i(x))^2 - 1$, as per Theorem 3.4.4.

Then, it samples $\gamma \leftarrow_\$ \mathbb{F}$ and sets $\nu(x) := v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \gamma t(x)$. Let:

$$h(x) := \frac{\nu(x)^2 - 1}{t(x)} \tag{4.5}$$

be a randomized solution of the ssp. By linear evaluation it is possible to compute

$$\begin{aligned} H &:= \mathsf{Enc}(h(s)), \qquad \widehat{H} := \mathsf{Enc}(\alpha h(s)), \qquad \widehat{V} := \mathsf{Enc}\left(\alpha \nu(s)\right), \\ V_w &:= \mathsf{Enc}\left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right), \\ B_w &:= \mathsf{Enc}\left(\beta\left(\sum_{i=\ell_u+1}^{m} a_i v_i(s) + \gamma t(s)\right)\right). \end{aligned} \tag{4.6}$$

In fact, H (respectively, $\widehat{H}$) can be computed from the encodings of $1, s, \ldots, s^d$ (respectively, $\alpha, \alpha s, \ldots, \alpha s^d$) and the coefficients of Equation (4.5). The element $\widehat{V}$ can be computed from the encodings of $\alpha s, \ldots, \alpha s^d$. Finally, V_w (respectively, B_w) can be computed from the encodings of $s, \ldots, s^d$ (respectively, $\beta t(s), \beta v_{\ell_u+1}(s), \ldots, \beta v_m(s)$). All these linear evaluations involve at most $d+1$ terms, and the coefficients are bounded by p. By using the above elements, the prover returns a proof $\pi \coloneqq (H, \widehat{H}, \widehat{V}, V_w, B_w)$.

Verifier. Upon receiving a proof π and a statement $u = (a_1, \ldots, a_{\ell_u})$, the verifier, in possession of the verification key vrs (that implicitly contains the crs), proceeds with the following verifications. First, it uses the quadratic root detection algorithm of the encoding scheme Enc to verify that the proof satisfies:

$$\widehat{h}_s - \alpha h_s = 0 \text{ and } \widehat{v}_s - \alpha v_s = 0, \tag{eq-pke}$$

$$(v_s^2 - 1) - h_s t_s = 0, \tag{eq-div}$$

$$b_s - \beta w_s = 0. \tag{eq-lin}$$

where $(h_s, \widehat{h}_s, \widehat{v}_s, w_s, b_s)$ are the values encoded in $(H, \widehat{H}, \widehat{V}, V_w, B_w) \coloneqq \pi$ and t_s, v_s are computed as $t_s \coloneqq t(s)$ and $v_s \coloneqq v_0 + \sum_{i=1}^{\ell_u} a_i v_i(s) + w_s$.

Then, the verifier checks whether it is still possible to perform some homomorphic operations, using the $\mathsf{Test\text{-}error}$ procedure, implemented in Figure 4.3 for the specific case of lattice encodings. More precisely, the verifier tests whether it is still possible to add another encoding and multiply the result by an element bounded by p, without compromising the correctness of the encoded element. This will guarantee the existence of a reduction in the knowledge soundness proof of Section 4.6.2. If all above checks hold, return "true". Otherwise, return "false".

Remark 4.4.1. *Instantiating our encoding scheme on top of a "noisy" encryption scheme like Regev's introduces multiple technicalities that affect the protocol, the security proof, and the parameters' choice. For instance, in order to compute a linear combination of d encodings via* Eval *we need to scale down the error parameter and consequently increase the parameters q and n in order to maintain correctness and security. Similarly, the distributions of the error terms and the random vectors are affected by the homomorphic evaluation, and we must guarantee that the resulting terms are still simulatable. All these issues will be formally addressed in Section 4.6, and then analyzed more pragmatically in Section 4.7.*

4.5 Assumptions

Throughout this chapter, we rely on various computational assumptions. All of them are long-standing assumptions in the frame of DLog-hard groups, and they have been stated in Section 2.3.2 as standard q-type assumptions over bilinear groups. They have already been generalized in the scope of "encoding schemes" in [GGPR13]. We recall them here in term of our encoding scheme (Definition 4.2.3) with the necessary adaptations.

Assumption 4.5.1 (q-PKE)**.** *The q-Power Knowledge of Exponent (q-PKE) assumption holds relative to an encoding scheme* Enc *and for the class $\mathcal{Z}$ of non-uniform* PPT *auxiliary input generators if, for every non-uniform* PPT *adversary $\mathcal{A}$, there exists a non-uniform* PPT *extractor $\mathcal{E}_{\mathcal{A}}$ such that:*

$$\mathsf{Adv}^{\mathsf{q\text{-}pdh}}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_{\mathcal{A}}} \coloneqq \Pr\left[\mathsf{q\text{-}PKE}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_{\mathcal{A}}} = \mathbf{true}\right] = \mathsf{negl},$$

where $\mathsf{q\text{-}PKE}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}}$ *is the game depicted in Figure 4.5.*

The q-PDH assumption has been a long-standing, standard q-type assumption [Gro10, BBG05]. It basically states that given $(\mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q), \mathsf{Enc}(s^{q+2}), \ldots, \mathsf{Enc}(s^{2q}))$, it is hard to compute an encoding of the missing power $\mathsf{Enc}(s^{q+1})$.

Assumption 4.5.2 (q-PDH)**.** *The q-Power Diffie-Hellman (q-PDH) assumption holds for encoding* Enc *if for all* PPT *adversaries $\mathcal{A}$ we have:*

$$\mathsf{Adv}^{\mathsf{q\text{-}pdh}}_{\mathsf{Enc},\mathcal{A}} := \Pr[\, q\text{-}\mathsf{PDH}_{\mathsf{Enc},\mathcal{A}} = \mathbf{true} \,] = \mathsf{negl},$$

where q-$\mathsf{PDH}_{\mathsf{Enc},\mathcal{A}}$ *is defined as in Figure 4.5.*

$\mathsf{q\text{-}PKE}_{\mathsf{Enc},\mathcal{Z},\mathcal{A},\mathcal{E}_\mathcal{A}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$\alpha, s \leftarrow\!\!\$\, \mathbb{F}^*$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc}(1),$
$\mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q), \mathsf{Enc}(\alpha),$
$\mathsf{Enc}(\alpha s), \ldots, \mathsf{Enc}(\alpha s^q))$
$z \leftarrow \mathcal{Z}(\mathsf{pk}, \sigma)$
$(\mathsf{ct}, \widehat{\mathsf{ct}}; \{a_i\}_i^q) \leftarrow (\mathcal{A} \| \mathcal{E}_\mathcal{A})(\sigma, z)$
return $(\widehat{\mathsf{ct}} - \alpha\mathsf{ct} = 0) \ \wedge$
$\mathsf{ct} \notin \{\mathsf{Enc}(\sum_i^q a_i s^i)\}$

$\mathsf{q\text{-}PKEQ}_{\mathsf{Enc},\mathcal{A},\mathcal{E}_\mathcal{A}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$s \leftarrow\!\!\$\, \mathbb{F}$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc}(1),$
$\mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q),$
$\mathsf{Enc}(s^{q+2}), \ldots, \mathsf{Enc}(s^{2q}))$
$(\mathsf{Enc}(c), e; b) \leftarrow (\mathcal{A} \| \mathcal{E}_\mathcal{A})(\sigma)$
if $b = 0$ **return** $e \in \{\mathsf{Enc}(c)\}$
else return $e \notin \{\mathsf{Enc}(c)\}$

$\mathsf{q\text{-}PDH}_{\mathsf{Enc},\mathcal{A}}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$s \leftarrow\!\!\$\, \mathbb{F}$
$\sigma \leftarrow (\mathsf{pk}, \mathsf{Enc}(1),$
$\mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q),$
$\mathsf{Enc}(s^{q+2}), \ldots, \mathsf{Enc}(s^{2q}))$
$y \leftarrow \mathcal{A}(\sigma)$
return $y \in \{\mathsf{Enc}(s^{q+1})\}$

Figure 4.5: Games for q-PKE, q-PKEQ, q-PDH assumptions.

Finally, we need another assumption to be able to "compare" adversarially-generated messages. The q-PKEQ assumption states that for any adversary $\mathcal{A}$ that outputs two ciphertexts, there exists an extractor $\mathcal{E}_\mathcal{A}$ that can tell whether they encode the same value.

Assumption 4.5.3 (q-PKEQ)**.** *The q-Power Knowledge of Equality (q-PKEQ) assumption holds for the encoding scheme* Enc *if, for every* PPT *adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}_\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{q\text{-}pkeq}}_{\mathsf{Enc},\mathcal{A},\mathcal{E}_\mathcal{A}} := \Pr[\, q\text{-}\mathsf{PKEQ}_{\mathsf{Enc},\mathcal{A},\mathcal{E}_\mathcal{A}} = \mathbf{true} \,] = \mathsf{negl},$$

where q-$\mathsf{PKEQ}_{\mathsf{Enc},\mathcal{A},\mathcal{E}_\mathcal{A}}$ *is the game depicted in Figure 4.5.*

This last assumption is needed solely in the case where the attacker has access to a verification oracle (see Section 4.6). Since the encoding could be non-deterministic, the simulator in the security reduction of Section 4.6 needs to rely on q-PKEQ to simulate the verification oracle. Pragmatically, this assumption allows us to test for equality of two adversarially-produced encodings without having access to the secret key.

4.5.1 Assumptions Comparison

We study the relations between our knowledge assumption and the (extractable) linear-only assumption.

At a first glimpse, it might seem unjustified to have brought assumptions often used in the dLog setting into the lattice domain, where they are highly non-standard. Despite this fact, in this section, we argue
(i) that the q-PKE and q-PDH assumptions are weaker than the extractable linear-only malleability for wich the works [BCI+13, BISW17] claim to get a post-quantum SNARK scheme, and
(ii) what consequence an attack on those assumptions would have.

Throughout the last years, a long line of research in lattice-based cryptography has been trying to develop fully-homomorphic encryption schemes and bilinear pairing maps. So far, no bilinear map is known in the context of lattices, and some have argued that its existence would lead to efficient cryptographic primitives such as multilinear maps and indistinguishability obfuscation (iO).

However, to prevent any undesired malleability, the LWE encodings used in our construction are carrefully tailored. These encodings have some pre-added noise, so that their malleability is limited to a small number of linear operations that are needed for the proof evaluation. Therefore, breaking our scheme in terms of homomorphism is equivalent to solving the problem of efficiently removing noise in such LWE encoding schemes (which is known as an open problem). We see this as a win-win situation.

Moreover, our assumptions are *weaker* than previously employed assumptions for lattice-based SNARGs. Indeed, the linear-only assumption of [BCI+13, BISW17] (see Figure 3.11 in Section 3.5.1) informally states that an adversary can only perform affine operations over the encodings provided as input.

It is not immediately clear to see what the consequences of this assumption are in the case of LWE encodings (like the one we presented in section Section 4.3) or the one in [LP11], used in [BISW17]. Consider for example a set of parameters Γ allowing for $d-1$ homomorphic operations modulo p and the adversary $\mathcal{A}$ that, upon receiving as input d ciphertexts, computes d homomorphic linear operations on them. With non-negligible probability the error would wrap around the modular representation, leading to a "decryptable" encoding (any element of $\mathbb{Z}_q^{n+1}$ is a valid encoding) but for which the adversary does not know an affine map. The authors of [BISW17] suggest to use *double-encryption* in these cases, i.e., present two different encodings of each value, and ask the adversary to homomorphically evaluate these terms twice. If the two ciphertexts do not encode the same element, the game is lost. Obviously, this comes at the cost of doubling the size of each encoding and the computation time for the prover and the verifier.

We will try in this section to argue that the assumptions we rely on for our SNARK construction are weaker than the ones in

We say that an encoding scheme is IND-CPA in the same sense as for encryption schemes (see Figure 2.14) if any PPT adversary has negligible probability of distinguishing the encoding of any two chosen messages.

Theorem 4.5.4. *If* Enc *is an* IND-CPA *extractable linear-only encoding scheme as defined in Figure 3.11, it satisfies* q-PDH *(see Figure 4.5).*

Proof. Let us consider an adversary $\mathcal{A}_{\mathsf{PDH}}$ for the q-PDH assumption. We show that there

exists an adversary $\mathcal{A}$ able to break IND-CPA.

Consider the PPT machine $\mathcal{A}$ that samples uniformly at random two field elements, s_0 and s_1, then submits the two distinct chosen plaintexts s_0^q, s_1^q to the IND-CPA challenger. $\mathcal{A}$ queries the IND-CPA encoding oracle with s_0^k, s_1^k for $k = 0, \dots, q-1, q+2, \dots, 2q$. The oracle gives back some encodings $\sigma_0 := \left(\mathsf{Enc}(1), \mathsf{Enc}(s_0), \dots, \mathsf{Enc}(s_0^{q-1}), \mathsf{Enc}(s_0^{q+2}), \dots, \mathsf{Enc}(s_0^{2q})\right)$ and $\sigma_1 := \left(\mathsf{Enc}(1), \mathsf{Enc}(s_1), \dots, \mathsf{Enc}(s_1^{q-1}), \mathsf{Enc}(s_1^{q+2}), \dots, \mathsf{Enc}(s_1^{2q})\right)$. When it receives the chalenged ciphertext $\mathsf{Enc}(s_b^q)$ in the IND-CPA experiment, $\mathcal{A}$ runs the q-PDH adversary $\mathcal{A}_{\mathsf{PDH}}$ on $\{\sigma_0, \mathsf{Enc}(s_b^q)\}$ and on $\{\sigma_1, \mathsf{Enc}(s_b^q)\}$, thus in one of the cases obtaining (with non-negligible probability) some encoding $\mathsf{ct}_b \in \left\{\mathsf{Enc}(s_b^{q+1})\right\}$ where $b \in \{0,1\}$. By EXT-LO assumption, as stated in Figure 3.11, for any such adversary $\mathcal{A}_{\mathsf{PDH}}$ able to produce a new ciphertext, there exists an extractor $\mathcal{E}_{\mathsf{LO}}$ which, given as input σ_b, $b \in \{0,1\}$ (for the succesful case) and the same random coins of the adversary $\mathcal{A}_{\mathsf{PDH}}$, returns a polynomial p such that $p(s_b) = s_b^{q+1}$. Let $f(x) := p(x) - x^{q+1}$. By our assumption of success in q-PDH game, $f(s_b) = 0$; by Schwartz-Zippel lemma, $f(s_{1-b}) \neq 0$ with probability $1 - 2q/|\mathbb{F}| = 1 - \mathsf{negl}$. $\mathcal{A}$ returns the bit b^* such that $f(s_{b^*}) = 0$, solving the IND-CPA challenge with overwhelming probability. □

Theorem 4.5.5. *If* Enc *is an* IND-CPA *extractable linear-only encoding scheme, it satisfies* q-PKE.

Proof. We will show that Enc satisfies q-PKE, meaning there is no couple $(\mathcal{A}_{\mathsf{PKE}} \| \mathcal{E}_{\mathcal{A}})$ able to win the q-PKE game (cf. Figure 4.5).

Suppose, by contradiction, there exists an adversary $\mathcal{A}_{\mathsf{PKE}}$ that is able to produce a valid output $\mathsf{ct}, \widehat{\mathsf{ct}}$, i.e., such that $\alpha\mathsf{ct} - \widehat{\mathsf{ct}} \in \mathsf{Enc}(0)$. We show that there is an extractor $\mathcal{E}_{\mathcal{A}}$ that outputs the correct linear combination with non negligible probability.

Let M be the plaintext generation algorithm that, upon receiving the computational security parameter λ and $d = 2q + 2$ in unary form, samples $s \leftarrow\!\!\$\, \mathbb{F}$ and outputs plaintexts $1, s, \dots, s^q$. Let $\sigma \leftarrow (\mathsf{Enc}(1), \mathsf{Enc}(s), \dots, \mathsf{Enc}(s^q), \mathsf{Enc}(\alpha), \mathsf{Enc}(\alpha s), \dots, \mathsf{Enc}(\alpha s^q))$. The adversary $\mathcal{A}_{\mathsf{PKE}}$, when run on this input σ, outputs (with non-negligible probability) $\mathsf{ct}, \widehat{\mathsf{ct}}$ such that $\alpha\mathsf{ct} - \widehat{\mathsf{ct}} \in \mathsf{Enc}(0)$.

Let us define the adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ for the game EXT-LO that, upon receiving as input σ, run the same instantiation of $\mathcal{A}_{\mathsf{PKE}}$ and output ct - respectively $\widehat{\mathsf{ct}}$. By our claim of linear-only property, there exist the extractors $\mathcal{E}_0$ and $\mathcal{E}_1$ for $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively, outputting $a_0, \dots, a_q, b_0, \dots, b_q$ and $a'_0, \dots, a'_q, b'_0, \dots, b'_q$ such that

$$\mathsf{ct} \in \left\{\mathsf{Enc}\left(\textstyle\sum_i^q a_i s^i + \sum_i^q b_i \alpha s^i\right)\right\}, \quad \widehat{\mathsf{ct}} \in \left\{\mathsf{Enc}\left(\textstyle\sum_i^q a'_i s^i + \sum_i^q b'_i \alpha s^i\right)\right\}$$

with non negligible probability.

Knowing that $\alpha\mathsf{ct} - \widehat{\mathsf{ct}} \in \mathsf{Enc}(0)$ implies either that the polynomial

$$P(X,Y) = X^2 \textstyle\sum_i^q b_i Y^i + X \sum_i^q (a_i - b'_i) Y^i - \sum_i^q a'_i Y^i$$

is the zero polynomial, or that (α, s) are roots of $P(X,Y)$. The second case is ruled out by semantic security of the encoding scheme and Schwartz-Zippel lemma, by a reasoning similar to the proof of Theorem 4.5.4.

The case where $P(X,Y) = 0$ gives us $b_i = a'_i = 0, a_i = b'_i,\ \forall i = 0, \dots, q$. Therefore, we are able to define an extractor $\mathcal{E}_{\mathcal{A}}$ for q-PKE that outputs the coefficients a_i of the linear

combination with non-negligible probability, showing that any successful adversary against q-PKE able to output $\mathsf{ct}, \widehat{\mathsf{ct}}$ such that $\alpha\mathsf{ct} - \widehat{\mathsf{ct}} = 0$, has the knowledge of the coefficients a_i such that $\mathsf{ct} \in \left\{\mathsf{Enc}\left(\sum_i^d a_i s^i\right)\right\}$.

□

4.6 Proofs of Security

We aim to construct designated-verifier SNARKs, so we will make some further observations that were not introduced in Chapter 3.

Strong Knowledge Soundness. An important consideration that arises when defining knowledge soundness in the designated-verifier setting is whether the adversary should be granted access to a verification oracle. Pragmatically, allowing the adversary to query a verification oracle captures the fact that CRS can be reused poly times.

While this distinction does not need to be made in the publicly-verifiable setting, the same is not true for the designated-verifier setting. In the specific case of our construction, we formulate and prove our protocol allowing the adversary access to the $\Pi.\mathsf{Ver}(\mathsf{vrs}, \cdot, \cdot)$ oracle (which has been named *strong soundness* in the past [BISW17]), and later discuss which optimizations can take place when using the weaker notion of soundness, where the adversary cannot access a verification oracle.

In this section, we prove our main theorem:

Theorem 4.6.1. *If the* q-PKE*,* q-PKEQ *and* q-PDH *assumptions hold for the encoding scheme* Enc*, the protocol* Π *on* Enc *is a zk-SNARK with statistical completeness, statistical zero-knowledge and computational knowledge soundness.*

Proof of statistical completeness. Corollary 4.3.3 states the conditions on Γ for which the homomorphically computed encodings are valid with probability at least $1-\mathsf{negl}(\kappa)$. Lemma 4.3.6 affirms that correctly generated proofs satisfy Equation (4.2) with probability overwhelming in κ. Therefore $\mathsf{Test\text{-}error}$ returns true, and completeness follows trivially by Theorem 3.4.4. □

4.6.1 Zero-Knowledge

To obtain a zero-knowledge protocol, we do two things: we add a smudging term to the noise of the encoding, in order to make the distribution of the final noise independent of the coefficients a_i, and we randomize the target polynomial $t(x)$ to hide the witness. The random vectors constituting the first element of the ciphertext are guaranteed to be statistically indistinguishable from uniformly random vectors by leftover hash lemma (cf. Lemma 4.3.9).

Proof of zero-knowledge. The simulator $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Gen}}, \mathsf{Sim}_{\mathsf{Prove}})$ for zero-knowledge is shown in Figure 4.6. The errors are independently sampled from the same uniform distribution over the (integer) interval $[-2^\kappa T\sigma_{B_w},\ 2^\kappa T\sigma_{B_w}]$, where T is a small constant and $\sigma_{B_w} := p\sigma\sqrt{d+1}\sqrt{p^2+m-\ell_u}$. We will call this the *smudging distribution.*

Checking that the proof output by $\mathsf{Sim}_{\mathsf{Prove}}$ is indeed correct (i.e., that it verifies Equations (eq-pke) to (eq-lin)) is trivial. We are left with showing that the two proofs, the real one and the simulated one, are statistically indistinguishable.

Note that once the value of V_w in the proof has been fixed, the verification equations uniquely determine $H, \widehat{H}, \widehat{V}$, and B_w. This means that for any (u, w) such that $\mathsf{C}(u, w) = 1$,

both the real arguments and the simulated arguments are chosen uniformly at random such that the verification equations will be satisfied. One can prove that values for V_w are statistically indistinguishable when executing $\Pi.\mathsf{P}$ and $\mathsf{Sim}_{\mathsf{Prove}}$: V_w is the encoding of a uniformly random variable γ_w in Sim and the masking of a polynomial evaluation by adding $\gamma t(s)$, where γ is chosen uniformly at random (note that $t(s) \neq 0$) in $\Pi.\mathsf{P}$. What is encoded in the remaining terms is simply dictated by the verification constraints.

In both worlds, the proof is a tuple of 5 encodings $(H, \widehat{H}, \widehat{V}, V_w, B_w)$. Once the vrs is fixed, each encoding can be written as $(-\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + pe + m)$, for some $\mathbf{a} \in \mathbb{Z}_q^n$ and some $m \in \mathbb{Z}_p$ satisfying the verification equations. Due to Lemma 4.3.9, the random vectors $\mathbf{a}$ are indistinguishable from uniformly random in both worlds. The error terms are statistically indistinguishable due to Lemma 4.3.4. (See Section 4.7 for a detailed explanation of these values.)

The zero-knowledge follows from these claims, since the simulator can use re-randomization to ensure that its actual encodings (not just what is encoded) are appropriately uniform. □

The zero-knowledge property is undoubtedly interesting, but it comes at a cost: smudging the error terms requires us to scale the ciphertext modulus by κ bits. For those applications where the zero-knowledge property is not required, we can simplify the protocol by removing $\gamma t(x)$ from the computation of $h(x)$ and avoiding the smudging procedure on every proof term.

$\mathsf{Sim}_{\mathsf{Gen}}(1^\lambda, \mathsf{C})$

$\alpha, \beta, s \leftarrow\!\!\$\ \mathbb{F}$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{K}(1^\lambda)$
$(v_0, \ldots, v_m(x), t(x)) \leftarrow \mathsf{SSP}(\mathsf{C})$
Compute crs as per Eq. 4.4
$\mathsf{td} \coloneqq (\mathsf{sk}, s, \alpha, \beta)$
return $(\mathsf{crs}, \mathsf{td})$

$\mathsf{Sim}_{\mathsf{Prove}}(\mathsf{td}, u)$

$(\mathsf{sk}, s, \alpha, \beta) \coloneqq \mathsf{td}$; $(a_1, \ldots, a_{\ell_u}) \coloneqq u$
$a_0 \coloneqq 1$; $\gamma_w \leftarrow\!\!\$\ \mathbb{F}$
$h \coloneqq \left((\sum_{i=0}^{\ell_u} a_i v_i(s) + \gamma_w)^2 - 1 \right) \ / \ t(s)$
$H \leftarrow \mathsf{Enc}(h)$; $\widehat{H} \leftarrow \mathsf{Enc}(\alpha h)$
$\widehat{V} \leftarrow \mathsf{Enc}(\sum_{i=0}^{\ell_u} a_i \alpha v_i(s) + \alpha \gamma_w)$
$V_w \leftarrow \mathsf{Enc}(\gamma_w)$; $B_w \leftarrow \mathsf{Enc}(\beta \gamma_w)$
Apply smudging on $H, \widehat{H}, \widehat{V}, B_w, V_w$
return $(H, \widehat{H}, \widehat{V}, V_w, B_w)$

Figure 4.6: Simulator for Zero-Knowledge.

4.6.2 Knowledge Soundness

Before diving into the technical details of the proof of soundness, we provide some intuition in an informal sketch of the security reductions: the CRS for the scheme contains encodings of $\mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^d)$, as well as encodings of these terms multiplied by some field elements $\alpha, \beta \in \mathbb{F}$. The scheme requires the prover P to exhibit encodings computed homomorphically from such CRS.

The reason why we require the prover to duplicate its effort w.r.t. α is so that the simulator in the security proof can extract representations of $\widehat{V}$, $\widehat{H}$ as degree-d polynomials $v(x)$, $h(x)$

such that $v(s) = v_s, h(s) = h_s$, by the q-PKE assumption (for $q = d$). The assumption also guarantees that this extraction is efficient. This explains the first quadratic root detection check Equation (eq-pke) in the verification algorithm.

Suppose an adversary manages to forge a SNARK of a false statement and pass the verification test. Then, the soundness of the square span program (Theorem 3.4.4) implies that, for the extracted polynomials $v(x), h(x)$ and for the new defined polynomial $v_{\mathsf{mid}}(x) \coloneqq v(x) - v_0(x) - \sum_i^{\ell_u} a_i v_i(x)$, one of the following must be true:

i. $h(x)t(x) \neq v^2(x) - 1$, but $h(s)t(s) = v^2(s) - 1$, from Equation (eq-div);

ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m)$, but B_w is a valid encoding of $\mathsf{Enc}(\beta v_{\mathsf{mid}}(s))$, from Equation (eq-lin).

If the first case holds, then $p(x) \coloneqq (v^2(x) - 1) - h(x)t(x)$ is a nonzero polynomial of degree some $k \leq 2d$ that has s as a root, since the verification test implies $(v^2(s) - 1) - h(s)t(s) = 0$. The simulator can use $p(x)$ to solve q-PDH for $q \geq 2d - 1$ using the fact that $\mathsf{Enc}\left(s^{q+1-k}p(s)\right) \in \{\mathsf{Enc}(0)\}$ and subtracting off encodings of lower powers of s to get $\mathsf{Enc}(s^{q+1})$.

To handle the second case, i.e., to ensure that $v_{\mathsf{mid}}(x)$ is in the linear span of the $v_i(x)$'s with $\ell_u < i \leq m$ we use an extra scalar β, supplement the CRS with the terms $\{\mathsf{Enc}(\beta v_i(s))\}_{i>\ell_u}, \mathsf{Enc}(\beta t(s))$, and require the prover to present (encoded) $\beta v_{\mathsf{mid}}(s)$ in its proof. An adversary against q-PDH will choose a polynomial $\beta(x)$ convenient to solve the given instance. More specifically, it sets $\beta(x)$ with respect to the set of polynomials $\{v_i(x)\}_{i>\ell_u}$ such that the coefficient for x^{q+1} in $\beta(x)v_{\mathsf{mid}}(x)$ is zero. Then, to generate the values in the crs it uses $\beta \coloneqq \beta(s)$ (that can be computed from its input consisting of encodings of powers of s). All these allow it to run the SNARK adversary and to obtain from its output B_w an encoding of some polynomial with coefficient s^{q+1} non-zero and thus solve q-PDH. Also here, the verification algorithm guarantees that even with all the above homomorphic operations, the challenger still decrypts the correct value with $1 - \mathsf{negl}(\kappa)$ probability.

Proof of computational knowledge soundness. Let $\mathcal{A}_\Pi$ be the PPT adversary in the game for knowledge soundness (See Section 3.2, Figure 3.3) able to produce a proof π for which $\Pi.\mathsf{Ver}$ returned "true". We first claim that it is possible to extract the coefficients of the polynomial $v(x)$ corresponding to the values v_s encoded in V. The setup algorithm first generates the parameters $(\mathsf{pk}, \mathsf{sk})$ of an encoding scheme Enc and picks $\alpha, \beta, s \in \mathbb{F}$, which are used to compute $\mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^d), \mathsf{Enc}(\alpha), \mathsf{Enc}(\alpha s), \ldots, \mathsf{Enc}(\alpha s^d)$. Fix some circuit C, and let ssp be an SSP for C. Let $\mathcal{A}_{\mathsf{PKE}}$ be the d-PKE adversary, that takes as input a set of encodings:

$$\sigma \coloneqq \left(\mathsf{pk}, \mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^d), \mathsf{Enc}(\alpha), \mathsf{Enc}(\alpha s), \ldots, \mathsf{Enc}(\alpha s^d)\right).$$

The auxiliary input generator $\mathcal{Z}$ is the PPT machine that upon receiving as input σ, samples $\beta \leftarrow_{\$} \mathbb{Z}_p$, constructs the remaining terms of the CRS (as per Equation (4.4)), and outputs them in z using ssp. Thus, $\mathcal{A}_{\mathsf{PKE}}$ sets $\mathsf{crs} \coloneqq (\mathsf{ssp} \| \sigma \| z)$ and invokes $\mathcal{A}_\Pi(\mathsf{crs})$. As a result, it obtains a proof $\pi = (H, \widehat{H}, \widehat{V}, V_w, B_w)$. On this proof, it computes:

$$V \coloneqq \mathsf{Enc}\left(v_0 + \sum_{i=1}^{\ell_u} a_i v_i(s) + w_s\right) = V_w + v_0 + \sum_{i=1}^{\ell_u} a_i v_i(s). \tag{4.7}$$

where w_s is the element encoded in V_w. Finally, $\mathcal{A}_{\mathsf{PKE}}$ returns $(\widehat{V}, V)$. If the adversary $\mathcal{A}$ outputs a valid proof, then by verification equation Equation (eq-pke) it holds that the two encodings $(V, \widehat{V})$ encode values $v_s, \widehat{v}_s$ such that $\widehat{v}_s - \alpha v_s = 0$. Therefore, by q-PKE assumption, there exists an extractor $\mathcal{E}_{\mathsf{PKE}}$ that, using the same input (and random coins) of $\mathcal{A}_{\mathsf{PKE}}$, outputs a vector $(c_0, \ldots, c_d) \in \mathbb{F}^{d+1}$ such that V is an encoding of $\sum_{i=0}^{d} c_i s^i$ and $\widehat{V}$ is an encoding of $\sum_{i=0}^{d} \alpha c_i s^i$. In the same way, it is possible to recover the coefficients of the polynomial $h(x)$ used to construct $(H, \widehat{H})$, the first two elements of the proof of $\mathcal{A}_{\Pi}$ (again, by Equation (eq-pke)).

Our witness-extractor $\mathcal{E}_{\Pi}$, given crs, emulates the extractor $\mathcal{E}_{\mathsf{PKE}}$ above on the same input σ, using as auxiliary information z the rest of the CRS given as input to $\mathcal{E}_{\Pi}$. By the reasoning discussed above, $\mathcal{E}_{\Pi}$ can recover $(c_0, \ldots, c_d)$ coefficients extracted from the encodings $(V, \widehat{V})$. Consider now the polynomial $v(x) \coloneqq \sum_{i=0}^{d} c_i x^i$. If it is possible to write the polynomial as $v(x) = v_0(x) + \sum_i^m a_i v_i(x) + \delta t(x)$ such that $(a_1, \ldots, a_m) \in \{0,1\}^m$ satisfies the assignment for the circuit $\mathtt{C}$ with $u = (a_1, \ldots, a_{\ell_u})$, then the extractor returns the witness $w = (a_{\ell_u+1}, \ldots, a_m)$.

With overwhelming probability, the extracted polynomial $v(x) \coloneqq \sum_{i=0}^{d} c_i x^i$ does indeed provide a valid witness w. Otherwise, there exists a reduction to q-PDH that uses the SNARK adversary $\mathcal{A}_{\Pi}$. Define the polynomial

$$v_{\mathsf{mid}}(x) \coloneqq v(x) - v_0(x) - \sum_{i=1}^{\ell_u} a_i v_i(x)$$

We know by definition of SSP (Definition 3.4.3) and by Theorem 3.4.4 that $\mathtt{C}$ is satisfiable if and only if

$$t(x) \mid v^2(x) - 1 \ \wedge \ v_{\mathsf{mid}}(x) = \sum_{i}^{d} c_i x^i - v_0(x) - \sum_{i}^{\ell_u} a_i v_i(x) \in \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m, t)$$

Therefore, by contradiction, if the adversary $\mathcal{A}_{\Pi}$ does not know a witness $w \in \{0,1\}^{m-\ell_u}$ for u (such that $(u, w) \in \mathcal{R}_{\mathtt{C}}$), but still the two verification checks Equation (eq-div) and Equation (eq-lin) pass, we have that either one of the following two cases must hold:

i. $t(x)h(x) \neq v^2(x) - 1$, but $t(s)h(s) = v^2(s) - 1$; or

ii. $v_{\mathsf{mid}}(x) \notin \mathrm{Span}(v_{\ell_u+1}, \ldots, v_m, t)$, but B_w is an encoding of $\beta v_{\mathsf{mid}}(s)$.

Let $\mathcal{B}_{\mathsf{PDH}}$ be an adversary against the q-PDH assumption. Given a q-PDH challenge

$$\left(\mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q), \mathsf{Enc}\left(s^{q+2}\right), \ldots, \mathsf{Enc}\left(s^{2q}\right)\right), \qquad \text{for } q \in \{2d-1, d\}$$

adversary $\mathcal{B}_{\mathsf{PDH}}$ samples uniformly at random $\alpha \leftarrow\!\!\$\, \mathbb{F}$, and defines some $\beta \in \mathbb{F}$ (that we will formally construct later) and constructs a CRS as per Equation (4.4). There are some subtleties in how $\mathcal{B}_{\mathsf{PDH}}$ generates the value β. In fact, β can be generated without knowing its value explicitly, but rather knowing its representation over the power basis $\{s^i\}_{i=0, i\neq q+1}^{2q}$ – that is, knowing a polynomial $\beta(x)$ and its evaluation in s. Some particular choices of β will allow us to provide a solution for a q-PDH challenge. $\mathcal{B}_{\mathsf{PDH}}$ invokes the adversary $\mathcal{A}_{\Pi}$ as well as the extractor $\mathcal{E}_{\Pi}$ on the generated CRS, thus obtaining a proof π and the linear

combination used by the prover for the polynomials $h(x), v(x)$ and also extracts a witness for the statement being proved.

For the *strong soundness* (see Section 4.6), in order to simulate the verification oracle and to answer the verification queries of $\mathcal{A}_\Pi$, $\mathcal{B}_{\mathsf{PDH}}$ has to compare its encodings (obtained from the extracted coefficients and its input) with $\mathcal{A}$'s proof terms, accept if the terms match, and reject otherwise. Because the encoding scheme is not deterministic, adversary $\mathcal{B}_{\mathsf{PDH}}$ invokes the PKEQ extractor and simulates the verification oracle correctly with overwhelming probability.

The reduction in the two mentioned cases works as follows:

i. The extracted polynomials $h(x)$ and $v(x)$ satisfy $t(s)h(s) = v^2(s) - 1$, but $t(x)h(x) \neq v^2(x) - 1$. By q-PDH assumption this can happen only with negligible probability. We define $P(x) = v^2(x) - 1 - t(x)h(x)$, that in this case is a non-zero polynomial of degree $k \leq 2d$ having s as a root. Let p_k be the highest nonzero coefficient of $P(x)$. Write $\tilde{P}(x) = x^k - p_k^{-1} \cdot P(x)$. Since s is a root of $x^k - \tilde{P}(x)$, it is a root of $x^{q+1} - x^{q+1-k}\tilde{P}(x)$. $\mathcal{B}_{\mathsf{PDH}}$ solves q-PDH by computing $\mathsf{Enc}(s^{q+1}) = \mathsf{Enc}(s^{q+1-k}\tilde{P}(s))$ for $q = 2d - 1$. Since $\deg(\tilde{P}) \leq k - 1$, the latter is a known linear combination of encodings $\mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^q)$ which are available from the q-PDH challenge. More precisely, $\mathcal{B}_{\mathsf{PDH}}$ will compute $\mathsf{Eval}((\mathsf{Enc}(s^{i+q+1-k}))_i, (\tilde{p}_i)_i^{2d-1})$ on *fresh* encodings $\mathsf{Enc}(1), \mathsf{Enc}(s), \mathsf{Enc}(s^2), \ldots, \mathsf{Enc}(s^q)$ solving the q-PDH challenge for $q \geq 2d - 1$.

ii. In the second case, suppose that the polynomial v_{mid} extracted as previously described cannot be expressed as a linear combination of $\{v_{\ell_u+1}, \ldots, v_m, t\}$. The proof still passes the verification, so we have a consistent value for $B_w \in \{\mathsf{Enc}(\beta v_{\mathsf{mid}}(s))\}$.

$\mathcal{B}_{\mathsf{PDH}}$ generates a uniformly random polynomial $a(x)$ of degree q subject to the constraint that all of the polynomials $a(x)t(x)$ and $\{a(x)v_i(x)\}_{i=\ell_u+1}^m$ have coefficient 0 for x^{q+1}. We note that for $q = d$, there are $q - (m - \ell_u) > 0$ degrees of freedom in choosing $a(x)$.

$\mathcal{B}_{\mathsf{PDH}}$ defines β to be the evaluation of $a(x)$ in s, i.e., $\beta \coloneqq a(s)$. Remark that $\mathcal{B}_{\mathsf{PDH}}$ does not know s explicitly, but having access to the encodings of $2q - 1$ powers of s, it is able to generate valid encodings $(\mathsf{Enc}(\beta v_i(s)))_i$ and $\mathsf{Enc}(\beta t(s))$ using Eval. Note that, by the construction of β, this evaluation is of $d + 1$ elements in $\mathbb{F}$ and that the $(q + 1)$-th power of s is never used. Now, since $v_{\mathsf{mid}}(x)$ is not in the proper span, the coefficient of degree $q + 1$ of $xa(x)v_{\mathsf{mid}}(x)$ must be nonzero with overwhelming probability $1 - 1/|\mathbb{F}|$. The term B_w of the proof must encode a known polynomial in s: $\sum_{i=0}^{2q} b_i s^i \coloneqq \beta v_{\mathsf{mid}}(s) = a(s)v_{\mathsf{mid}}(s)$ where the coefficient b_{q+1} is non-trivial. $\mathcal{B}_{\mathsf{PDH}}$ can subtract off encodings of multiples of other powers of s to recover $\mathsf{Enc}(s^{q+1})$ and break q-PDH. This requires an evaluation on *fresh* encodings:

$$\mathsf{Eval}\left((\mathsf{Enc}(s^i))_{\substack{i=0 \\ i\neq q+1}}^{q+d}, (-b_i)_{\substack{i=0 \\ i\neq q+1}}^{q+d}\right). \tag{4.8}$$

Adding the above to B_w and multiplying by the inverse of the $(q + 1)$-th coefficient (using once again Eval) will provide a solution to the q-PDH problem for $q = d$.

Since the two cases above are not possible by q-PDH assumption, $\mathcal{E}_\Pi$ extracts a valid witness if the proof of $\mathcal{A}_\Pi$ is valid. □

As previously mentioned in Section 4.6, the proof of knowledge soundness allows oracle access to the verification procedure. In the context of a weaker notion of soundness where

the adversary does not have access to the $\Pi.\mathsf{Ver}(\mathsf{vrs},\cdot,\cdot)$ oracle, the proof is almost identical, except that there is no need for the $\mathcal{B}_{\mathsf{PDH}}$ adversary to answer queries and to simulate the verification, and therefore no need for the q-PKEQ assumption anymore.

4.7 Efficiency and Concrete Parameters

The prover's computations are bounded by the security parameter and the size of the circuit, i.e., $\mathsf{P} \in \widetilde{O}(\lambda d)$. As in [GGPR13, DFGK14], the verifier's computations depend solely on the security parameter, i.e., $\mathsf{Ver} \in O(\lambda)$. The proof consists of a constant number (precisely, 5) of LWE encodings, i.e., $|\pi| = 5 \cdot \widetilde{O}(\lambda)$. Finally, the complexity of the setup procedure is $\widetilde{O}(\lambda d)$.

Using the propositions from Section 4.3 and knowing the exact number of homomorphic operations that need to be performed in order to produce a proof, we can now attempt at providing some concrete parameters for our encoding scheme.

We fix the statistical security parameter $\kappa := 32$, as already done in past works on fully homomorphic encryption (e.g., [DM15, CGGI16]). We fix the circuit size $d := 2^{15}$, which is sufficient for some practical applications such as the computation of SHA-256. For some practical examples of circuits, we direct the reader towards [BCG+14, PHGR13].

For a first attempt at implementing our solution, we assume a weaker notion of soundness, i.e., that in the KS game the adversary does not have access to a verification oracle (cf. Figure 3.3). Concretely, this means that the only bound in the size of p is given by the guessing probability of the witness, and the guessing of a field element. We thus fix p to be a prime[3] of 32 bits for the size of the message space.

The CRS is composed of encodings of different nature: some of them are fresh $(\mathsf{Enc}(1), \mathsf{Enc}(s), \ldots, \mathsf{Enc}(s^d))$, some happen to be stale in the construction of $\mathcal{A}_{\mathsf{PKE}}$ and the construction of $\mathcal{B}_{\mathsf{PDH}}$ Section 4.6.2 (Item i.) $(\mathsf{Enc}(\alpha s), \ldots, \mathsf{Enc}(\alpha s^d))$, and some are stale from the construction of $\mathcal{B}_{\mathsf{PDH}}$ Section 4.6.2 (Item ii.) $(\mathsf{Enc}(\beta t(s)), (\mathsf{Enc}(\beta v_i(s)))_i)$. They are displayed in Figure 4.7.

Since, as we have seen, $\mathcal{B}_{\mathsf{PDH}}$ manipulates the q-PDH challenge via homomorphic operations, we must guarantee that the protocol adversary can perform *at least* the same number of homomorphic operations as in the real-world protocol. Therefore, in the real protocol, we must intentionally increase the magnitude of the noise in the CRS: the terms $\mathsf{Enc}(\alpha s^i)$ (with $i = 0, \ldots, d$) are generated by multiplying the respective *fresh* encoding $\mathsf{Enc}(s^i)$ by a term bounded by p; the terms $\mathsf{Enc}(\beta t(s)), \{\mathsf{Enc}(\beta v_i(s))\}_i$ instead are generated via Eval of $d+1$ elements with coefficients bounded by p. Concretely, when encoding these elements using the encoding scheme of Section 4.3, the error for $\mathsf{Enc}(\alpha s^i)$ is sampled from $p \cdot \chi_\sigma$; the error for $\mathsf{Enc}(\beta t(s)), \mathsf{Enc}(\beta v_i(s)))$ is sampled from $(p\sqrt{d+1}) \cdot \chi_\sigma$.

The proof π consists of five elements $(H, \widehat{H}, \widehat{V}, V_w, B_w)$, as per Equation (4.6). H and V_w are computed using an affine function on d encodings with coefficients modulo p; $\widehat{H}, \widehat{V}$ are computed using a linear function on $d+1$ encodings with coefficients modulo p; finally, B_w is computed using a linear combination of $m - \ell_u$ encodings with coefficients in $\{0, 1\}$, except the last one which is modulo p. Overall, the term that carries the highest load of homomorphic computations is B_w. The generation of B_w is outlined in Figure 4.7, and to it (as well as to the other proof terms) we add a smudging term for constructing a zero-knowledge proof π.

[3] In particular, we need p and q to be relatively prime for the correctness of the encoding scheme [BV11a, footnote 18].

In the construction of the adversary $\mathcal{B}_{\mathsf{PDH}}$ (Item ii.), we need to perform some further homomorphic operations on the proof element B_w in order to solve the q-PDH challenge, namely one addition (Equation (4.8)) and one multiplication by a known scalar b bounded by p.

The result of the first operation is denoted by $\mathsf{Enc}(b \cdot s^{q+1})$ in Figure 4.7; the final result is the solution to the q-PDH challenge.

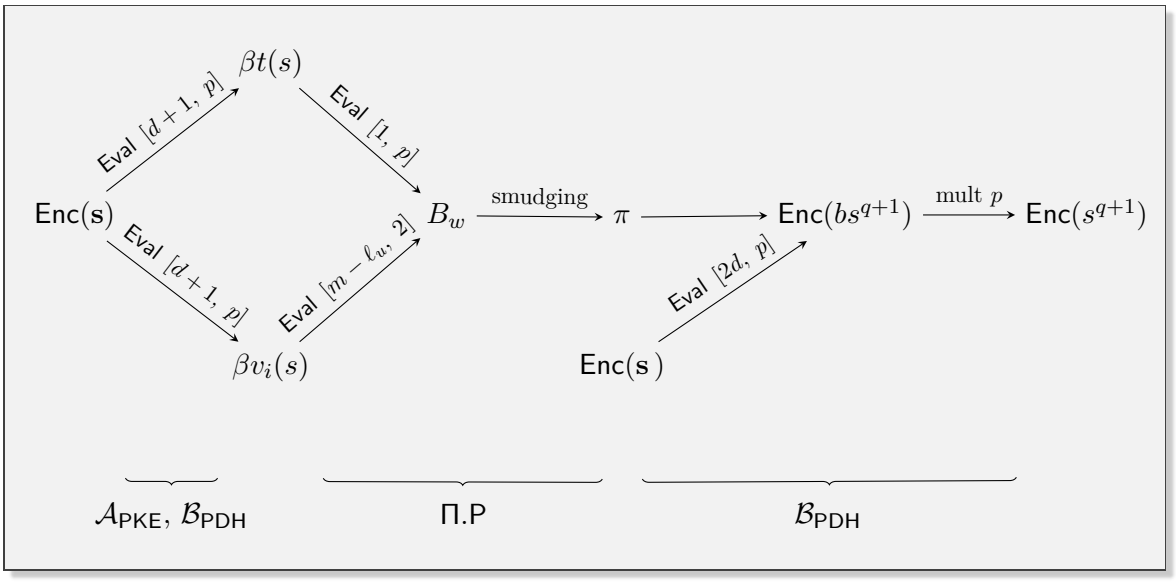


Figure 4.7: Summary of evaluations in the security proof.

We now outline the calculations that we use to choose the relevant parameters for our encoding scheme. In particular, we will focus on the term B_w since, as already stated, it is the one that is involved in the largest number of homomorphic operations.

The chain of operations that need to be supported is depicted in Figure 4.7: The leftmost part of the figure refers to the construction of adversaries for q-PKE and q-PDH; the central part refers to the protocol itself (i.e., the construction of the proof π); the rightmost part refers to the construction of the adversary for q-PDH. The syntax $\mathsf{Eval}\,[d, p]$ denotes a homomorphic evaluation on d encodings with coefficients in $\mathbb{Z}_p$. $\mathsf{Enc}(\mathbf{s})$ denotes the PDH challenge. We now analyze them one by one.

The correctness of the other terms follows directly from Corollary 4.3.3.

First of all, the terms $(\beta v_i(s))_i$ and $\beta t(s)$ are produced through the algorithm Eval executed on $d+1$ *fresh* encodings with coefficients modulo p. Let σ be the discrete Gaussian parameter of the noise terms in fresh encodings; then, by Pythagorean additivity, the Gaussian parameter of encodings output by this homomorphic evaluation is $\sigma_{\mathsf{Eval}} \coloneqq p\sigma\sqrt{d+1}$. Then the term $\beta t(s)$ is multiplied by a coefficient in $\mathbb{Z}_p$, and the result is added to a subset sum of the terms $(\beta v_i(s))_i$, i.e., a weighted sum with coefficients in $\{0, 1\}$. It is trivial to see that, for the first term, the resulting Gaussian parameter is bounded by $p\sigma_{\mathsf{Eval}}$, whereas for the second term it is bounded by $\sigma_{\mathsf{Eval}}\sqrt{m-\ell_u}$. The parameter of the sum of these two terms is then bounded by $\sigma_{B_w} \coloneqq \sigma_{\mathsf{Eval}}\sqrt{p^2+m-\ell_u}$. Let us then consider a constant factor T for "cutting the Gaussian tails", i.e., such that the probability of sampling from the distribution and

obtaining a value with magnitude larger than T times the standard deviation is as small as desired. We can then write that the absolute value of the error in B_w is bounded by $T\sigma_{B_w}$. At this point we add a *smudging* term, which amounts to multiplying the norm of the noise by $(2^\kappa + 1)$ (cf. Corollary 4.3.5). Finally, the so-obtained encoding has to be summed with the output of an Eval invoked on $2d$ fresh encodings with coefficients modulo p and multiplied by a constant in $\mathbb{Z}_p$. It is easy to calculate that the final noise is then bounded by $Tp\sigma_{B_w}(2^\kappa + 1) + Tp\sigma_{\mathsf{Eval}}$ (cf. Lemma 4.3.6). By substituting the values of σ_{Eval}, σ_{B_w}, remembering that $\sigma \coloneqq \alpha q$ and imposing the condition for having a valid encoding, we obtain

$$Tp^2\alpha q\sqrt{d+1}\left(\sqrt{p^2+m-\ell_u}\,(2^\kappa+1)+1\right) < \frac{q}{2p}.$$

The above corresponds to Equation (4.3) with bounds $Be \coloneqq T\sigma_{B_w}$ and $B_{\mathsf{Eval}} \coloneqq T\sigma_{\mathsf{Eval}}$. By simplifying q and isolating α, we get:

$$\alpha < \frac{1}{2Tp^3\sqrt{d+1}\left(\sqrt{p^2+m-\ell_u}\,(2^\kappa+1)+1\right)}.$$

With our choice of parameters and by taking $T = 8$, we can select for instance $\alpha = 2^{-180}$.

Once α and p are chosen, we select the remaining parameters q and n in order to achieve the desired level of security for the LWE encoding scheme. To do so, we take advantage of Albrecht's estimator[4] [APS15b] which, as of now, covers the following attacks: meet-in-the-middle exhaustive search, coded-BKW [GJS15], dual-lattice attack and small/sparse secret variant [Alb17], lattice reduction with enumeration [LP11], primal attack via uSVP [AFG14, BG14], Arora-Ge algorithm [AG11] using Gröbner bases [ACFP14].

Finally, based on these parameters, we can concretely compute the size of the CRS[5] and that of the proof π. The CRS is composed of $d + (d+1) + (m+1)$ encodings, corresponding to the encodings of the d powers of s, the encodings of α multiplied by the $d+1$ powers of s, the m encodings of $(\beta v_i)_i$, and the encoding of $\beta t(s)$. This amounts to $(2d + m + 2)$ LWE encodings, each of which has size $(n+1)\log q$ bits[6]. For the calculations, we bound m by d and state that the size of the CRS is that of $(3d+2)$ LWE encodings. From an implementation point of view, we can consider LWE encodings $(\vec{a}, b) \in \mathbb{Z}_q^{n+1}$ where the vector $\vec{a}$ is the output of a seeded PRG. This has been proven secure in the random oracle model [Gal13]. Therefore, the communication complexity is greatly reduced, as sending an LWE encoding just amounts to sending the seed for the PRG and the value $b \in \mathbb{Z}_q$. For security to hold, we can take the size of the seed to be λ bits, thus obtaining the final size of the CRS: $(3d+2)\log q + \lambda$ bits. The proof π is composed of 5 LWE encodings, therefore it has size $|\pi| = 5(n+1)\log q$ bits. Note that in this case, we cannot trivially use the same trick with the PRG, since the encodings are produced through homomorphic evaluations.

4.7.1 Implementation

We implemented our construction in standard C11, using the library GMP [Gt12] for handling arbitrary precision integers and the library FLINT [HJP13] for handling polynomials. We

[4] https://bitbucket.org/malb/lwe-estimator

[5] We take into account only the encodings that are contained in the CRS. The other terms have considerably smaller impact on its size or can be agreed upon offline (e.g.,, the SSP).

[6] Note that the magnitude of the noise term, i.e., whether the encoding is fresh or stale, has no impact on the size of an encoding. This size is a function only of n (the number of elements in the vector) and the modulus q.

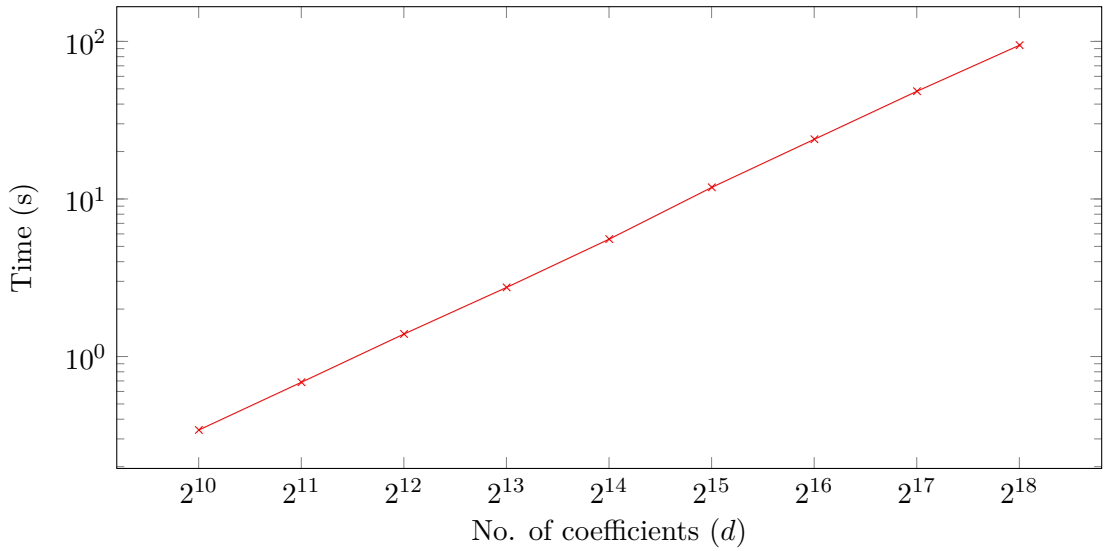


Figure 4.8: Time measurements for Eval.

chose p to be the pseudo-Mersenne prime $2^{32}-5$, and $q = 2^{736}$. This allows for fast arithmetic operations: reduction modulo q simply consists in a bitmask, modular operations by p can fit a `uint64_t` type, and multiplication of a scalar modulo p to a vector in $\mathbb{Z}_q^{n+1}$ does not require any allocation for the carry. The dimension of the lattice was chosen $n = 1470$, corresponding to the "medium" security level discussed in [GMNO18].

We performed extensive benchmarks of our protocol on a single thread of an Intel Core i7-4720HQ CPU @ 2.60GHz, running Arch Linux (kernel version 4.14.39). Our implementation is publicly available [7].

[7] See `https://www.di.ens.fr/~orru/pq-zk-snarks`.

Chapter 5

O-SNARKs

On the (In)security of SNARKs in the Presence of Oracles. In this chapter, we study the feasibility of knowledge extraction for SNARKs in a scenario that has not been analyzed before. While prior work focuses on the case of adversarial provers that may receive (statically generated) auxiliary information, here we consider the scenario where adversarial provers are given access to an oracle. For this setting, we study if and under what assumptions such provers can admit an extractor. First, we formalize the question of extraction in the presence of oracles by proposing a suitable knowledge soundness definition for this setting. We call SNARKs satisfying this definition O-SNARKs. Second, we study whether O-SNARKs exist, providing both negative and positive results. On the negative side, we show that, assuming one-way functions, there do not exist O-SNARKs in the standard model for every signing oracle family (and thus for general oracle families as well). On the positive side, we show that when considering signature schemes with appropriate restrictions on the message length O-SNARKs for the corresponding signing oracles exist, based on classical SNARKs.

Contents

Tale three: Just the place for a SNARK!

Young Alice is very concerned about her health, and she always wants to be in good shape and eat healthily. She found an app that can track her activity, that collects information about her lifestyle, and later it is able to calculate the perfect diet and a personalized exercise program for her. She wants to try it out, but of course, relying on the advice of a random service found on the Internet is not very wise. Alice is cautious and she understands the importance of having the possibility of testing the correctness of the results, especially when those have an impact on her health.
Now, that proofs of knowledge are not a secret anymore to her, she thinks of using SNARKs for securing this system. Piece of cake! She contacts the service provider and asks if they can implement such a proof as a functionality of their system.
She has an unexpected answer that makes her doubt about the universality of SNARK schemes and their applications.
As she already knew, a SNARK is a tool to prove knowledge of a process, of a computation.
The service provider tells Alice that they cannot prove knowledge of all the process, because they do not have this complete knowledge either!
They explain that in order to provide their customers with the personalized health programs they get some vital part of their advice from external specialists.
They are continuously updating with the latest medical studies, but they do not have control over how these studies have been done, and they instead receive directly the certified parameters they should apply to their computation.
Alice finds this explanation very plausible; she would not mind the service to ask for further advice to some medical center that she also trusts. However, in these conditions, there must be a way to prove that they did their work as expected using the certified parameters from the medical center.
Alice thinks about this new, unexpected situation, and she wonders if the SNARKs do not have too many limitations to be used in such complex applications.
It seems that if one is provided with external knowledge, SNARKs are not of very much help. The way a SNARK is defined does not allow to use it securely in any possible application. That is such a pity!
This raises many very intricate questions that Alice would like to answer... Can we change the way we define SNARKs? How would this new tool look like and work?
Alice has to solve a new cryptographic puzzle to be able to secure her new app.

5.1 Introduction

5.1.1 Extraction for Proving Knowledge

We have seen so far how the SNARKs schemes work and how we can build secure schemes with classical security, and even with post-quantum guarantees. The cryptographic community assigned a lot of effort in constructing efficient such schemes for the sake of various applications. Notably, their powerful property of *knowledge soundness* makes them a unique tool in more complex protocols where we need to guarantee that different parties "know" some information or some process. We will focus in this chapter in examining some use cases of SNARK schemes and see the limitations we encounter.

Recall that assuming such a soundness property is formally stated in terms of some *knowledge extraction* in the adaptive knowledge soundness definition: more precisely, for any prover algorithm, there is an efficient extractor such that, whenever the prover returns an accepting proof,the extractor outputs a valid witness.

Extraction with Auxiliary Input. Unfortunately, stated as above, knowledge soundness is insufficient for being used in many applications. The problem is that, when using SNARKs in larger cryptographic protocols, adversarial provers may get additional information which can contribute to the generation of adversarial proofs. To address this problem, a stronger, and more useful, definition of knowledge soundness requires that for any adversary $\mathcal{A}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that, for any honestly generated crs and any polynomial-size auxiliary input z, whenever $\mathcal{A}(\mathsf{crs}, \mathsf{z})$ returns an accepting proof, $\mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{z})$ outputs a valid witness.

This type of definition is certainly more useful for using SNARKs in larger cryptographic protocols, but it also introduces other subtleties. As first discussed in [HT98], extraction in the presence of arbitrary auxiliary input can be problematic, if not implausible. Formal evidence of this issue has been recently given in [BCPR14, BP15]. Bitansky et al. [BCPR14] show that, when assuming indistinguishability obfuscation, there do not exist extractable one-way functions (and thus SNARKs) with respect to arbitrary auxiliary input of unbounded polynomial length. Boyle and Pass [BP15] generalize this result showing that assuming collision-resistant hash functions and differing-input obfuscation, there is a fixed auxiliary input distribution for which extractable one-way functions do not exist.

5.1.2 Extraction in the Presence of Oracles

In this chapter, we continue the study on the feasibility of extraction by looking at a scenario that, to the best of our knowledge, has not been explicitly analyzed before. We consider the case in which adversarial provers run in interactive security experiments where they are given *access to an oracle*. For this setting, we study if and under what assumptions such provers can admit an extractor.

Before giving more detail on our results, let us discuss a motivation for analyzing this scenario. To keep the presentation simple, here we give a motivation via a hypothetical example; more concrete applications are discussed later.

A case study application. Consider an application where Oscar gets a collection of signatures generated by Bob, and he has to prove to Alice that he owns a valid signature of Bob on some message m such that $P(m) = 1$. Let us say that this application is secure if Oscar, after asking for signatures on several messages, cannot cheat letting Alice accept for a false statement (i.e., $P(m) = 0$, or $P(m) = 1$ but Oscar did not receive a signature on m). If

messages are large and one wants to optimize bandwidth, SNARKs can be a perfect candidate solution for doing such proofs,[1] i.e., Oscar can generate a knowledge soundness of (m, σ) such that "(m, σ) verifies with Bob's public key and $P(m) = 1$".

An attempt of security proof. Intuitively, the security of this protocol should follow easily from the knowledge soundness of the SNARK and the unforgeability of the signature scheme. However, somewhat surprisingly, the proof becomes quite subtle. Let us consider a cheating Oscar that always outputs a proof for a statement in the language.[2] If Oscar is still cheating, then it must be that he is using a signature on a message that he did not query – in other words a forgery. Then one would like to reduce such a cheating Oscar to a forger for the signature scheme.

To do this one would proceed as follows. For any Oscar, one defines a forger that, on input the verification key vk, generates the SNARK crs, gives $(\mathsf{crs}, \mathsf{vk})$ to Oscar, and simulates Oscar's queries using its own signing oracle. When Oscar comes with the cheating proof, the forger would need an extractor for Oscar in order to obtain the forgery from him. However, even if we see Oscar as a SNARK prover with auxiliary input vk, Oscar does not quite fit the knowledge soundness definition in which adversaries have no oracles. To handle similar cases, one typically shows that for every interactive Oscar, there is a non-interactive algorithm $\mathcal{B}$ that runs Oscar simulating his oracles (i.e., $\mathcal{B}$ signs queries by sampling the signing key) and returns the same output. The good news is that for such $\mathcal{B}$ one can claim the existence of an extractor $\mathcal{E}_{\mathcal{B}}$ as it fits the knowledge soundness definition. The issue is though that $\mathcal{E}_{\mathcal{B}}$ expects the same input of $\mathcal{B}$, which includes the secret signing key. This means that our candidate forger (which does not have the secret signing key) cannot run this extractor.

Applications that need extraction with oracles. Besides the above example, this issue can show up essentially in every application of SNARKs in which adversaries have access to oracles with a secret state, and one needs to run an extractor during an experiment (e.g., a reduction) where the secret state of the oracle is not available.

In this chapter, we address this problem by providing both negative and positive results to the feasibility of extraction in the presence of oracles. On one hand, our negative results provide an explanation of why the above proofs do not go through so easily. On the other hand, our positive results eventually provide precise guidelines to formally state and prove the security of the cryptographic constructions mentioned above (albeit with some restrictions).

5.1.3 An Overview of Our Results

As a first step, we formalize the problem of defining non-black-box extraction in the presence of oracles by proposing a notion of *SNARKs in the presence of oracles* (O-SNARKs for short). In a nutshell, an O-SNARK is like a SNARK except that adaptive knowledge soundness must hold with respect to adversaries that have access to an oracle $\mathcal{O}$ sampled from some oracle family $\mathbb{O}$.[3] Slightly more in detail, we require that for any adversary $\mathcal{A}^{\mathcal{O}}$ with access to O there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that, whenever $\mathcal{A}^{\mathcal{O}}$ outputs a valid proof, $\mathcal{E}_{\mathcal{A}}$ outputs a valid witness, by running on the same input of $\mathcal{A}$, plus the transcript of oracle queries and answers made by $\mathcal{A}$. The advantage of this new notion is that it lends itself to easy and intuitive security proofs in all those applications where one needs to execute extractors in

[1] Further motivation can also be to maintain the privacy of m by relying on zero-knowledge SNARKs.
[2] The other case of statements not in the language can be easily reduced to the soundness of the SNARK.
[3] In fact, the notion is parametrized by the family $\mathbb{O}$, i.e., we say Π is an O-SNARK for $\mathbb{O}$.

interactive security games with oracles. In particular, we show that by replacing SNARKs with O-SNARKs (for appropriate oracle families) we can formally prove the security of the cryptographic constructions mentioned in the previous section.

Besides its applications, the next (and more intriguing) question is whether O-SNARKs exist. This is what we study in the second part of this work. Our results are summarized in the following paragraphs.

O-SNARKs in the random oracle model. As a first positive result, we show that the construction of CS proofs of Micali [Mic00] yields an O-SNARK for *every* oracle family, in the random oracle model. This result follows from the work of Valiant [Val08] which shows that Micali's construction already allows for extraction. More precisely, using the power of the random oracle model, Valiant shows a *black-box* extractor. This powerful extractor can then be used to build an O-SNARK extractor that works for any oracle family.

Insecurity of O-SNARKs for every oracle family, in the standard model. Although the above result gives a candidate O-SNARK, it only works in the random oracle model, and it is tailored to one construction. It is therefore interesting to understand whether extraction with oracles is feasible in the *standard model*. And it would also be interesting to see if this is possible for classical SNARKs. Besides its theoretical interest, the latter question has also a practical motivation since there are several efficient SNARK constructions proposed in the last years that one might like to use in place of CS proofs. Our first result in this direction is that assuming existence of collision-resistance hash functions (CRHFs) there do not exist O-SNARKs for NP with respect to *every* oracle family. More in detail we show the following:

Theorem 5.1.1 (Informal)**.** *Assume the existence of CRHFs. Then for any polynomial $p(\cdot)$ there is a family of oracles $\mathbb{O}_p$ such that any candidate O-SNARK for NP, that is correct and succinct with proofs of length bounded by $p(\cdot)$, cannot satisfy adaptive knowledge soundness with respect to oracles from $\mathbb{O}_p$.*

A basic intuition behind our result is that oracles provide additional auxiliary input to adversaries and, as formerly shown in [BCPR14, BP15], this can create issues for extraction. In fact, to obtain our result, we might also have designed an oracle that simply outputs a binary string following a distribution with respect to which extraction is impossible due to [BCPR14, BP15]. However, in this case, the result should additionally assume the existence of indistinguishability (or differing-input) obfuscation. In contrast, our result shows that such impossibility holds by only assuming the existence of CRHFs, which is a much weaker assumption.

Insecurity of O-SNARKs for every family of signing oracles. Motivated by the three applications mentioned earlier,[4] we focus on studying the existence of O-SNARKs with respect to *signing oracles*. Along this direction, our first observation is that it is possible to embed the code of our counterexample oracle inside the signing algorithm of any signature scheme. This yields a signature scheme which is unforgeable, yet for the corresponding signing oracle extraction is impossible (unless one can find a hash collision). Basically, this shows that there do not exist O-SNARKs with respect to every signing oracle.

However, we further study the case of signing oracles, and show that

[4] We do believe that many more applications along the same line – proving knowledge of valid signatures – are conceivable.

Theorem 5.1.2 (Informal)**.** *Assume unforgeable signature schemes exist. Then for any polynomial $p(\cdot)$ there is an unforgeable signature scheme Σ_p such that any candidate O-SNARK, that is correct and succinct with proofs of length bounded by $p(\cdot)$, cannot satisfy adaptive knowledge soundness with respect to signing oracles corresponding to Σ_p.*

Although this counterexample is more complex than the previous one based on CRHFs, this is interesting for at least two reasons. First, it can be based on a weaker assumption (OWFs vs CRHFs). Second, the (secure) signature scheme we design makes the generic homomorphic signature construction completely insecure. Namely, when plugging the scheme Σ_p in the homomorphic signature construction, there is an adversary which can break its security. This essentially shows that the proof issue is not just a difficulty in finding a way to do the proof but that certain oracles can actually lead to insecure constructions.

Finally, let us note that the two negative results mentioned so far also imply that even SNARKs that are secure in the classical sense, cannot satisfy adaptive knowledge soundness with respect to every family of oracles.

Existence of O-SNARKs for specific families of signing oracles. We study ways to circumvent our impossibility result for signing oracles that we mentioned earlier. Indeed, the above result can be interpreted as saying that there exist (perhaps degenerate) signature schemes such that there are no O-SNARKs with respect to the corresponding signing oracle family. This is not ruling out that O-SNARKs may exist for specific signature schemes, or – even better – for specific classes of signature schemes. We provide the following positive answers to this question:

1. We show that hash-and-sign signatures, where the hash function is modeled as a random oracle, yield "safe oracles", i.e., oracles such that any classical SNARK is an O-SNARK for that oracle, in the random oracle model. The only restriction of this implication is that the classical SNARK needs to be secure w.r.t. auxiliary input consisting of random strings plus signatures on random messages. Note that this result does not guarantee that extraction is always possible, but it allows to formally reduce extraction in the presence of oracles to extraction with respect to auxiliary input from a specific distribution.

Unfortunately, this result remains of limited interest in terms of applications. The first reason is that this yields an O-SNARK in the random oracle model for which a candidate – Micali's CS proofs – is already known. The second reason is more subtle and is related with the fact that hash-and-sign signatures use a random oracle: in all those applications where the SNARK is used to prove knowledge of valid signatures, the statement to prove can no longer be defined using an NP machine as the verification algorithm invokes the random oracle. Hence, except for conjecturing that this still works when replacing the random oracle with a suitable hash function, one cannot derive a formal proof.

2. We then turn our attention to the standard model setting. We show that any classical SNARK is an O-SNARK for signing oracles if the message space of the signature scheme is properly bounded, and O-SNARK adversaries query "almost" the entire message space. More precisely, we show the following:

Theorem 5.1.3 (Informal)**.** *Let Σ be a signature scheme with message space $\mathcal{M}$ where $|\mathcal{M}| = \mathsf{poly}(\lambda)$ (resp. $|\mathcal{M}| = \lambda^{\omega(1)}$), and let $Q = |\mathcal{M}| - c$ for a constant $c \in \mathbb{N}$. Then a classical SNARK that is polynomially (resp. sub-exponentially) secure is also an O-SNARK for the family of signing oracles corresponding to Σ, for adversaries that make $|\mathcal{M}| - c$ signing queries.*

Another restriction of the above theorem is that the classical SNARK must allow for extraction w.r.t. auxiliary input consisting of a verification key and a collection of signatures on random messages. We also show a similar theorem for a more complex family of multiple signing oracles, which we use to prove the security of succinct functional signatures.

Interpretation of our results. In line with recent work [BCPR14, BP15] on the feasibility of extraction in the presence of auxiliary input, our results indicate that additional care must be taken when considering extraction *in the presence of oracles*. While for auxiliary input impossibility of extraction is known under obfuscation-related assumptions, in the case of oracles we show that extraction becomes impossible even when only assuming collision-resistance hash functions. Furthermore, our work establishes a framework that eases the analysis and the use of SNARK extractors in all those security experiments where these are given access to an oracle. Our results for signing oracles allow to relate the security of SNARKs in this new scenario to their security in the (better studied) case of extraction with auxiliary input, thus avoiding to make further (and new) conjectures on the plausibility of extractability assumptions. Also, we believe that our results for signing oracles are general enough to be re-used in other applications where SNARKs are employed to prove knowledge of valid signatures.

Finally, we remark that our counterexamples are of artificial nature and do not give evidence of extraction's impossibility in the presence of "natural" oracles. In this sense, they can be seen of theoretical interest. Yet, given the importance of provable security and considering the increasing popularity of SNARKs in more practical scenarios, we believe these results can also be useful to protocol designers.

5.2 SNARKs with Auxiliar Input

We recall here the classical definition of SNARKs and the knowledge soundness property, that is classically stated for some auxiliary input generated using a "benign" relation generator. Through this chapter, we will consider only publicly verifiable SNARKs.

Definition 5.2.1 (SNARK for NP [BCC+14])**.** *A SNARK* $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ *is defined by three algorithms:*

$\mathsf{Gen}(1^\lambda, T) \to \mathsf{crs}$**:** on input a security parameter $\lambda \in \mathbb{N}$ and a time bound $T \in \mathbb{N}$, this algorithm outputs a common reference string crs;

$\mathsf{Prove}(\mathsf{crs}, u, w) \to \pi$**:** given a prover reference string crs, a statement u and a witness w s.t. $(u, w) \in \mathcal{R}$, this algorithm produces a proof π;

$\mathsf{Ver}(\mathsf{crs}, u, \pi) \to b$**:** on input a the crs, an instance u, and a proof π, the verifier algorithm outputs $b = 0$ (reject) or $b = 1$ (accept);

satisfying *completeness*, *succinctness*, *knowledge-soundness*. We recall only the knowledge soundness property which is central in this chapter. (See Section 3.2 for the complete definition.)

Knowledge Soundness. For every PPT adversarial prover $\mathcal{A}$ of size $s(\lambda) = \mathsf{poly}(\lambda)$ there exists a non-uniform extractor $\mathcal{E}_\mathcal{A}$ of size $t(\lambda) = \mathsf{poly}(\lambda)$ and a negligible function $\varepsilon(\lambda)$ such that for every auxiliary input $\mathsf{z} \in \{0,1\}^{poly(\lambda)}$, and every time bound $T \in \mathbb{N}$,

$$\mathsf{Adv}^{\mathsf{ks}}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}} := \Pr\left[\mathsf{KS}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda) = \mathbf{true}\right] = \mathsf{negl},$$

$\underline{\mathsf{KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z})}$

$\mathsf{crs} \leftarrow \mathsf{Gen}(1^{\lambda}, T)$

$\mathsf{z} \leftarrow \mathcal{Z}(1^{\lambda})$

$(u, \pi; w) \leftarrow (\mathcal{A} \| \mathcal{E}_{\mathcal{A}})(\mathsf{crs}, \mathsf{z})$

return $\big(\mathcal{R}(u, w) = 0$

$\wedge \ \mathsf{Ver}(\mathsf{crs}, u, \pi)\big) = 1$

Figure 5.1: Game for adaptive knowledge soundness for auxiliary input

where $\mathsf{KS}_{\Pi,\mathcal{A},\mathcal{E}_{\mathcal{A}}}(\lambda)$ is defined in Figure 5.1.

Furthermore, we say that Π satisfies (s, t, ε)-adaptive proof of knowledge if the above condition holds for concrete values (s, t, ε).

Remark 5.2.2 (About extraction and auxiliary input)**.** *First, we stress that in the* KS *property the extractor* $\mathcal{E}_{\mathcal{A}}$ *takes exactly the same input of* $\mathcal{A}$*, including its random tape. Second, the* KS *definition can also be relaxed to hold with respect to auxiliary inputs from specific distributions (instead of arbitrary ones). Namely, let* $\mathcal{Z}$ *be a probabilistic algorithm (called the auxiliary input generator) that outputs a string* z*, and let compactly denote this process as* $\mathsf{z} \leftarrow \mathcal{Z}$*. Then we say that adaptive knowledge soundness holds for* $\mathcal{Z}$ *if the above definition holds for auxiliary inputs sampled according to* $\mathcal{Z}$*, where* $\mathcal{Z}$ *is also a non-uniform polynomial-size algorithm. More formally, we have the following definition.*

Definition 5.2.3 ($\mathcal{Z}$-auxiliary input SNARKs)**.** Π *is called a* $\mathcal{Z}$*-auxiliary input SNARK if* Π *is a SNARK as in Definition 5.2.1 except that adaptive knowledge soundness holds for auxiliary input* $\mathsf{z} \leftarrow \mathcal{Z}$*.*

On the Limits of Auxiliary Input. As discussed in previous work [HT98, BCPR14, BP13], dealing with auxiliary inputs is a delicate aspect of the SNARK definition (and extractability assumptions in general). Bitansky et al. [BCPR14] showed that indistinguishability obfuscation implies that there are potential auxiliary inputs to the adversary that allow it to create a valid proof in an obfuscated way such that it is impossible to extract the witness. Boyle and Pass [BP15] show later that assuming the stronger notion of public coin differing input obfuscation there is even auxiliary inputs that defeat witness extraction for all candidate SNARKs:

Theorem 5.2.4 (Informal [BP15])**.** *Assume the existence of fully homomorphic encryption with decryption in* NC^1*. Then there exist efficiently computable distributions* $\bar{\mathcal{Z}}, \bar{\mathcal{D}}$ *such that one of the following two primitives does* not *exist:*

- *SNARKs w.r.t. auxiliary input from* $\bar{\mathcal{Z}}$*.*
- *differing-inputs obfuscation for distribution* $\bar{\mathcal{D}}$ *of* NC^1 *circuits and auxiliary inputs.*

5.3 SNARKs in the Presence of Oracles

In this section we formalize the notion of extraction in the presence of oracles for SNARKs. We do this by proposing a suitable adaptive knowledge soundness definition, and we call a SNARK satisfying this definition a *SNARK in the presence of oracles* (O-SNARK, for short). As we shall see, the advantage of O-SNARKs is that this notion lends itself to easy and intuitive security proofs in all those applications where one needs to execute extractors in interactive security games with oracles (with a secret state). Below we provide the definition while the existence of O-SNARKs is discussed in Section 5.4.

5.3.1 O-SNARKs

Let $\mathbb{O} = \{\mathcal{O}\}$ be a family of oracles. We denote by $\mathcal{O} \leftarrow \mathbb{O}$ the process of sampling an oracle $\mathcal{O}$ from the family $\mathbb{O}$ according to some (possibly probabilistic) process. For example, $\mathbb{O}$ can be a random oracle family, i.e., $\mathbb{O} = \{\mathcal{O} : \{0,1\}^{\ell} \to \{0,1\}^{L}\}$ for all possible functions from ℓ-bits strings to L-bits strings, in which case $\mathcal{O} \leftarrow \mathbb{O}$ consists of choosing a function $\mathcal{O}$ uniformly at random in $\mathbb{O}$. As another example, $\mathbb{O}$ might be the signing oracle corresponding to a signature scheme, in which case the process $\mathcal{O} \leftarrow \mathbb{O}$ consists of sampling a secret key of the signature scheme according to the key generation algorithm (and possibly a random tape for signature generation in case the signing algorithm is randomized).

For any oracle family $\mathbb{O}$, we define an O-SNARK Π for $\mathbb{O}$ as follows.

Definition 5.3.1 ($\mathcal{Z}$-auxiliary input O-SNARKs for $\mathbb{O}$)**.** *We say that Π is a $\mathcal{Z}$-auxiliary input O-SNARK for the oracle family $\mathbb{O}$ if Π satisfies the properties of completeness and succinctness as in Definition 3.2.2, and the following property of adaptive knowledge soundness for $\mathbb{O}$:*

- **Adaptive Knowledge Soundness for $\mathbb{O}$.** Consider the experiment in Figure 5.2 for security parameter $\lambda \in \mathbb{N}$, time bound $T \in \mathbb{N}$, adversary $\mathcal{A}$, extractor $\mathcal{E}_{\mathcal{A}}$, auxiliary input generator $\mathcal{Z}$ and oracle family $\mathbb{O}$, where $\mathsf{qt} = \{q_i, \mathcal{O}(q_i)\}$ is the transcript of all oracle queries and answers made and received by $\mathcal{A}$ during its execution.

$\underline{\mathsf{O\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O})}$

$\mathsf{z} \leftarrow \mathcal{Z}(1^{\lambda});\ \mathcal{O} \leftarrow \mathbb{O}$

$\mathsf{crs} \leftarrow \mathsf{Gen}(1^{\lambda}, T)$

$\mathsf{qt} = (q_1, \mathcal{O}(q_1), \ldots, q_Q, \mathcal{O}(q_Q))$

$w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{z}, \mathsf{qt})$

return $(\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1)$

$\wedge\ ((y, w) \notin \mathcal{R})$

Figure 5.2: Game for adaptive knowledge soundness for $\mathbb{O}$.

Π satisfies adaptive knowledge soundness with respect to oracle family $\mathbb{O}$ and auxiliary input from $\mathcal{Z}$ if for every non-uniform *oracle* prover $\mathcal{A}^{\mathcal{O}}$ of size $s(\lambda) = \mathsf{poly}(\lambda)$ making

at most $Q(\lambda) = \mathsf{poly}(\lambda)$ queries there exists a non-uniform extractor $\mathcal{E}_{\mathcal{A}}$ of size $t(\lambda) = \mathsf{poly}(\lambda)$ and a negligible function $\varepsilon(\lambda)$ such that for every time bound T,

$$\Pr[\mathsf{O\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O}) \Rightarrow 1] \leq \varepsilon(\lambda)$$

Furthermore, we say that Π satisfies (s, t, Q, ε)-adaptive knowledge soundness with respect to oracle family $\mathbb{O}$ and auxiliary input from $\mathcal{Z}$ if the above condition holds for concrete values (s, t, Q, ε).

5.3.2 Non-Adaptive O-SNARKs

In this section we define a relaxation of O-SNARKs in which the adversary is non-adaptive in making its queries to the oracle. Namely, we consider adversaries that first declare all their oracle queries $q_1, \ldots, q_Q$ and then run on input the common reference string as well as the queries' outputs $\mathcal{O}(q_1), \ldots, \mathcal{O}(q_Q)$. More formally,

Definition 5.3.2 ($\mathcal{Z}$-auxiliary input non-adaptive O-SNARKs for $\mathbb{O}$)**.** *We say that* Π *is a* $\mathcal{Z}$*-auxiliary input* non-adaptive *O-SNARK for the oracle family* $\mathbb{O}$*, if* Π *satisfies the properties of completeness and succinctness as in Definition 3.2.2, and the following property of non-adaptive queries knowledge soundness for* $\mathbb{O}$*:*

- ***Non-Adaptive Knowledge Soundness for*** $\mathbb{O}$**.** *Consider the experiment in Figure 5.3 for security parameter* $\lambda \in \mathbb{N}$*, time bound* $T \in \mathbb{N}$*, adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, extractor* $\mathcal{E}_{\mathcal{A}}$*, auxiliary input generator* $\mathcal{Z}$ *and oracle family* $\mathbb{O}$ *where* st *is simply a state information shared between* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*.*

$\mathsf{O\text{-}nonAd\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O})$

$(q_1, \ldots, q_Q, st) \leftarrow \mathcal{A}_1(1^\lambda)$

$\mathsf{z} \leftarrow \mathcal{Z}(1^\lambda)$; $\mathcal{O} \leftarrow \mathbb{O}$

$\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, T)$

$\mathsf{qt} = (q_1, \mathcal{O}(q_1), \ldots, q_Q, \mathcal{O}(q_Q))$

$(y, \pi) \leftarrow \mathcal{A}_2(st, \mathsf{crs}, \mathsf{z}, \mathsf{qt})$

$w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{z}, \mathsf{qt})$

return $(\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1) \ \wedge \ ((y, w) \notin \mathcal{R})$

Figure 5.3: Game for non-adaptive knowledge soundness for $\mathbb{O}$.

Π *satisfies non-adaptive knowledge soundness with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if for every non-uniform prover* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *of size* $s(\lambda) = \mathsf{poly}(\lambda)$ *making at most* $Q(\lambda) = \mathsf{poly}(\lambda)$ *non-adaptive queries there exists a non-uniform extractor* $\mathcal{E}_{\mathcal{A}}$ *of size* $t(\lambda) = \mathsf{poly}(\lambda)$ *and a negligible function* $\varepsilon(\lambda)$ *such that for every time bound* T*,*

$$\Pr[\mathsf{O\text{-}nonAd\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathcal{Z}, \mathbb{O}) \Rightarrow 1] \leq \varepsilon(\lambda)$$

Furthermore, we say that Π *satisfies* (s, t, Q, ε)*-non-adaptive proof of knowledge with respect to oracle family* $\mathbb{O}$ *and auxiliary input from* $\mathcal{Z}$ *if the above condition holds for concrete values* (s, t, Q, ε)*.*

It is also possible to define a stronger variant of the above definition in which $\mathcal{A}_1$ is given (adaptive) oracle access to $\mathcal{O}$, whereas $\mathcal{A}_2$ has no access to $\mathcal{O}$, except for the query transcript obtained by $\mathcal{A}_1$. It is not hard to see that the result given in the following paragraph extends to work under this intermediate definition as well.

5.3.2.1 Existence of Non-Adaptive O-SNARKs from SNARKs.

Below we prove a simple result showing that non-adaptive O-SNARKs follow directly from classical SNARKs for which the knowledge soundness property holds for arbitrary auxiliary input distributions.

The idea of the proof is that the second stage adversary $\mathcal{A}_2$ of non-adaptive O-SNARKs is very much like a classical SNARK adversary that makes no queries and receives a certain auxiliary input which contains the set of oracle queries chosen by $\mathcal{A}_1$ with corresponding answers. The fact that the auxiliary input includes the set of queries chosen by $\mathcal{A}_1$, which is an arbitrary adversary, implies that the SNARK must support arbitrary, *not necessarily benign*, auxiliary inputs (i.e., it is not sufficient to fix an auxiliary input distribution that depends only on the oracle family $\mathbb{O}$).

Theorem 5.3.3. *Let* $\mathbb{O}$ *be any oracle family. If* Π *is a SNARK satisfying* (s, t, ε)*-adaptive* KS *(for any auxiliary input), then* Π *is a non-adaptive O-SNARK for* $\mathbb{O}$ *satisfying* (s, t, Q, ε)*-non-adaptive KS.*

Proof. The idea is that the second stage adversary $\mathcal{A}_2$ of non-adaptive O-SNARKs is very much like a classical SNARK adversary that makes no queries and receives a certain auxiliary input which contains the set of oracle queries chosen by $\mathcal{A}_1$ with corresponding answers. In addition to inputs distributed according to the family $\mathbb{O}$, the SNARK must satisfy knowledge soundness for The fact that the auxiliary input includes the set of queries chosen by $\mathcal{A}_1$, which is an arbitrary adversary, implies that the SNARK must support arbitrary, *not necessarily benign*, auxiliary inputs (i.e., it is not sufficient to fix an auxiliary input distribution that depends only on the oracle family $\mathbb{O}$).

Given the first stage adversary $\mathcal{A}_1$, and the oracle family $\mathbb{O}$ we define the following auxiliary input distribution:

$\mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}(1^\lambda)$
  $(\{q_1, \ldots, q_Q\}, st) \leftarrow \mathcal{A}_1(1^\lambda)$
  $\mathcal{O} \xleftarrow{\$} \mathbb{O}$
  return $\langle st, \{q_i, \mathcal{O}(q_i)\}_{i=1}^{Q} \rangle$

Then, for any $(\mathcal{A}_1, \mathcal{A}_2)$ we can build the following SNARK adversary $\mathcal{B}$ taking $z \leftarrow \mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}$:

$\mathcal{B}(\mathsf{crs}, z)$
  Parse $z = \langle st, q_1, y_1, \ldots, q_Q, y_Q \rangle$
  Run $\mathcal{A}_2(st, \mathsf{crs}, \mathsf{qt} = (q_1, y_1, \ldots, q_Q, y_Q)) \rightarrow (y, \pi)$
  Return the same (y, π) returned by $\mathcal{A}_2$.

Since Π is by assumption a SNARK, for $\mathcal{B}$ there exists an extractor $\mathcal{E}_{\mathcal{B}}$ such that, for any auxiliary input (and in particular for auxiliary input from $\mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}$), it holds

$$\Pr[\mathsf{O\text{-}nonAd\text{-}KS}(\lambda, T, \mathcal{B}, \mathcal{E}_{\mathcal{B}}, \mathcal{Z}_{\mathcal{A}_1,\mathbb{O}}) \Rightarrow 1] \leq \varepsilon(\lambda)$$

Finally, we simply define $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}}$. Since $\mathcal{B}$'s simulation of $\mathcal{A}_2$ is perfect, it is easy to see that for any $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ this extractor $\mathcal{E}_{\mathcal{A}}$ is such that

$$\Pr[\mathsf{O\text{-}nonAd\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}) \Rightarrow 1] \leq \varepsilon(\lambda)$$

□

5.4 On the Existence of O-SNARKs

In this section, we study whether O-SNARKs exist and under what assumptions. Namely, we consider the question of whether a SNARK Π is also an O-SNARK for *every* oracle family, or for any *specific* oracle family. In the following sections we give both positive and negative answers to this question.

We observe that in the positive cases where we prove that a SNARK Π is also an O-SNARK for a certain oracle family, we actually need to assume that Π is a $\mathcal{Z}$-auxiliary input SNARK where $\mathcal{Z}$ is an auxiliary input distribution that depends on the oracle. On one hand, such proofs establish a formal connection between the two notions (a connection that – we stress – is not always possible). On the other hand, one has then to make sure that extraction in the presence of $\mathcal{Z}$-auxiliary input is plausible.

5.4.1 O-SNARKs in the Random Oracle Model from Micali's CS Proofs

In this section we briefly discuss how the construction of CS proofs of Micali [Mic00] can be seen as an O-SNARK for *any* oracle family, albeit in the random oracle model. To see this, we rely on the result of Valiant [Val08] who shows that Micali's construction is a "CS proof of knowledge" in the random oracle model. The main observation is in fact that Valiant's proof works by showing a *black-box* extractor working for any prover.

Proposition 5.4.1. *Let $\mathbb{O}$ be any oracle family and RO be a family of random oracles. Let Π_{Mic} be the CS proof construction from [Mic00]. Then Π_{Mic} is an O-SNARK for $(\mathsf{RO}, \mathbb{O})$, in the random oracle model.*

Sketch. Let $\mathcal{E}^{\mathsf{RO}}$ be Valiant's black-box extractor[5] which takes as input the code of the prover and outputs a witness w. For any adversary $\mathcal{A}^{\mathsf{RO},\mathcal{O}}$ we can define its extractor $\mathcal{E}_{\mathcal{A}}$ as the one that, on input the query transcript qt of $\mathcal{A}$, executes $w \leftarrow \mathcal{E}^{\mathsf{RO}}(\mathcal{A})$ by simulating all the random oracle queries of $\mathcal{E}^{\mathsf{RO}}$ using qt, and finally outputs the same w. The reason why qt suffices to $\mathcal{E}_{\mathcal{A}}$ for simulating random oracle queries to $\mathcal{E}^{\mathsf{RO}}$ is that Valiant's extractor $\mathcal{E}^{\mathsf{RO}}$ makes exactly the same queries of the prover. □

[5] The CS proofs of knowledge definition used by Valiant considers adversaries that are non-adaptive in choosing the statement. However it easy to see that the construction and the proof work also for the adaptive case.

5.4.2 Impossibility of O-SNARKs in the Standard Model

In this section, we show that, *in the standard model*, there do not exist O-SNARKs with respect to *every* family of oracles. We show this under the assumption that universal one-way hash functions (and thus one-way functions [Rom90]) exist. To show the impossibility, we describe an oracle family in the presence of which any candidate O-SNARK that is correct and succinct cannot satisfy adaptive knowledge soundness with respect to that oracle family. More specifically, we do this by means of a signing oracle family. Namely, we show a secure signature scheme Σ_p such that every correct and succinct O-SNARK Π cannot satisfy adaptive knowledge soundness in the presence of the signing oracle corresponding to Σ_p. Interestingly, such a result not only shows that extraction cannot work for general families of oracles, but also for families of signing oracles, a class which is relevant to several applications.

For every signature scheme $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ we let $\mathbb{O}_\Sigma$ be the family of oracles $\mathcal{O}(m) = \mathsf{sign}(\mathsf{sk}, m)$, where every family member $\mathcal{O}$ is described by a secret key sk of the signature scheme, i.e., the process $\mathcal{O} \leftarrow \mathbb{O}_\Sigma$ corresponds to obtaining sk through a run of $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{kg}(1^\lambda)$. For the sake of simplicity, we also assume that the oracle allows for a special query, say $\mathcal{O}(\text{`}vk'),$[6] whose answer is the verification key vk.

Theorem 5.4.2. *Assume that one-way functions exist. Then for every polynomial $p(\cdot)$ there exists a* UF-CMA*-secure signature scheme Σ_p such that every candidate O-SNARK Π for* NP*, that is correct and succinct with proofs of length bounded by $p(\cdot)$, does not satisfy adaptive knowledge soundness with respect to $\mathbb{O}_{\Sigma_p}$.*

An Intuition of the Result. Before delving into the details of the proof, we provide the main intuition of this result. This intuition does not use signature schemes but includes the main ideas that will be used in the signature counterexample. Given a UOWHF function family $\mathcal{H}$, consider the NP binary relation $\tilde{R}_\mathcal{H} = \{((h, x), w) : h \in \mathcal{H}, h(w) = x\}$, let Π be a SNARK for NP and consider $p(\cdot)$ the polynomial for which Π is succinct. The idea is to show an oracle family $\tilde{\mathbb{O}}$ and an adversary $\bar{\mathcal{A}}$ for which there is no extractor unless $\mathcal{H}$ is not a universal one-way family. For every polynomial $p(\cdot)$, the oracle family contains oracles $\mathcal{O}_p$ that given a query q, interpret q as the description of a program $\mathcal{P}(\cdot, \cdot)$, samples a random member of the hash family $h \xleftarrow{\$} \mathcal{H}$, a random w, computes $x = h(w)$, and outputs (h, x) along with $\pi \leftarrow \mathcal{P}((h, x), w)$. If $\mathcal{P}(\cdot, \cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$, then the oracle is simply returning a hash image with a proof of knowledge of its (random) preimage. The adversary $\bar{\mathcal{A}}^{\mathcal{O}_p}$ is the one that on input crs, simply asks one query $q = \mathcal{P}(\cdot, \cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$, gets $((h, x), \pi) \leftarrow \mathcal{O}_p(q)$ and outputs $((h, x), \pi)$. Now, the crucial point that entails the non-existence of an extractor is that, provided that the input w is sufficiently longer than π, every valid extractor for such $\bar{\mathcal{A}}$ that outputs a valid w' immediately implies a collision (w, w') for h.[7] Finally, to prevent adversarially chosen $\mathcal{P}$ from revealing too much information, we require the oracle to check the length of π, and the latter is returned only if $|\pi| \leq p(\lambda)$.

Proof of Theorem 5.4.2. The proof consists of two main steps. First, we describe the construction of the signature scheme Σ_p based on any other UF-CMA-secure signature scheme $\widehat{\Sigma}$ with message space $\mathcal{M} = \{0, 1\}^*$ (that exists assuming OWFs [Lam79, Rom90]), and show

[6] Here vk is an arbitrary choice; any symbol not in $\mathcal{M}$ would do so. Introducing the additional query simplifies the presentation, otherwise vk should be treated as an auxiliary input from a distribution generated together with the oracle sampling.

[7] This relies on the fact that sufficiently many bits of w remain unpredictable, even given π.

that Σ_p is $\mathsf{UF\text{-}CMA}$-secure. Σ_p uses also a UOWHF family $\mathcal{H}$. Second, we show that, when considering the oracle family $\mathbb{O}_{\Sigma_p}$ corresponding to the signature scheme Σ_p, a correct Π with succinctness $p(\cdot)$ cannot be an O-SNARK for $\mathbb{O}_{\Sigma_p}$, i.e., we show an efficient O-SNARK adversary $\mathcal{A}_p^{\mathcal{O}}$ (with access to a Σ_p signing oracle $\mathcal{O}(\cdot) = \mathsf{sign}(\mathsf{sk}, \cdot)$), for which there is no extractor unless $\mathcal{H}$ is not one-way.

The Counterexample Signature Scheme Σ_p. Let $\widehat{\Sigma}$ be any $\mathsf{UF\text{-}CMA}$-secure scheme with message space $\mathcal{M} = \{0,1\}^*$. Let $\mathbb{H} = \{\mathcal{H}\}_\lambda$ be a collection of function families $\mathcal{H} = \{h : \{0,1\}^{q(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}$ where each $\mathcal{H}$ is an universal one-way hash family with $q(\lambda) \geq p(\lambda)+\ell(\lambda)+\lambda$. Let $M_{\mathcal{H}}((h,x),w)$ be the machine that on input $((h,x),w)$ accepts iff $h(w) = x$, and $\mathcal{R}_{\mathcal{H}}$ be the NP relation consisting of all pairs (y,w) such that, for $y = (M_{\mathcal{H}},(h,x),t)$, $M_{\mathcal{H}}((h,x),w)$ accepts in at most t steps.

The scheme Σ_p has message space $\mathcal{M} = \{0,1\}^*$; its algorithms work as follows:

$\mathsf{kg}(1^\lambda)$**:** Run $(\widehat{\mathsf{vk}}, \widehat{\mathsf{sk}}) \leftarrow \widehat{\Sigma}.\mathsf{kg}(1^\lambda)$, set $\mathsf{vk} = \widehat{\mathsf{vk}}$, $\mathsf{sk} = \widehat{\mathsf{sk}}$.

$\mathsf{sign}(\mathsf{sk}, m)$**:** Signing works as follows

- generate $\widehat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, m)$;
- sample $h \xleftarrow{\$} \mathcal{H}$ and $w \xleftarrow{\$} \{0,1\}^{q(\lambda)}$;
- compute $x = h(w)$, $t = \#M_{\mathcal{H}}((h,x),w)$, and set $y = (M_{\mathcal{H}},(h,x),t)$;
- interpret m as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y,w)$;
- if $|\pi| \leq p(\lambda)$, set $\pi' = \pi$, else set $\pi' = 0$;
- output $\sigma = (\widehat{\sigma}, h, x, \pi')$.

$\mathsf{vfy}(\mathsf{vk}, m, \sigma)$**:** Parse $\sigma = (\widehat{\sigma}, h, x, \pi')$ and return the output of $\widehat{\Sigma}.\mathsf{vfy}(\widehat{\mathsf{vk}}, m, \widehat{\sigma})$.

It is trivial to check that, as long as $\widehat{\Sigma}$ is a $\mathsf{UF\text{-}CMA}$-secure scheme, Σ_p is also $\mathsf{UF\text{-}CMA}$-secure. Moreover, remark that the scheme Σ_p does not depend on the specific O-SNARK construction Π but only on the universal polynomial $p(\cdot)$ bounding its succinctness.

Impossibility of O-SNARKs for $\mathbb{O}_{\Sigma_p}$. To show that Π is not an O-SNARK for $\mathbb{O}_{\Sigma_p}$ (under the assumption that $\mathcal{H}$ is universally one-way), we prove that there is an adversary $\mathcal{A}_p^{\mathcal{O}}$ such that every candidate extractor $\mathcal{E}$ fails in the adaptive knowledge soundness game.

Lemma 5.4.3. *If $\mathcal{H}$ is universally one-way then every Π for NP that is correct and succinct with proofs of length $p(\cdot)$ is not an O-SNARK for $\mathbb{O}_{\Sigma_p}$.*

Proof. Let $\mathcal{A}_p^{\mathcal{O}}$ be the following adversary: on input crs, encode the Prove algorithm of Π with hardcoded crs as a program $\mathcal{P}(\cdot,\cdot) := \mathsf{Prove}(\mathsf{crs},\cdot,\cdot)$; let q be $\mathcal{P}$'s description, and make a single query $\sigma = (\widehat{\sigma}, h, x, \pi') \leftarrow \mathcal{O}(q)$; return (y, π') where $y = (M_{\mathcal{H}},(h,x),t)$ is appropriately reconstructed. We show that for every polynomial-size extractor $\mathcal{E}$ it holds

$$\Pr[\mathsf{O\text{-}KS}(\lambda, \mathcal{A}_p, \mathcal{E}, \mathbb{O}_{\Sigma_p}) \Rightarrow 0] \leq \nu_{\mathcal{H}}(\lambda) + 2^{-\lambda}$$

where $\nu_{\mathcal{H}}(\lambda) = \mathsf{Adv}^{UOWHF}_{\mathcal{B},\mathcal{H}}(\lambda)$ is the advantage of any adversary $\mathcal{B}$ against $\mathcal{H}$'s universal one-wayness. This means that there is no extractor unless $\mathcal{H}$ is not a universal one-way family.

We proceed by contradiction assuming the existence of a polynomial-size extractor $\mathcal{E}$ such that the above probability is greater than some non-negligible ε. We show how to build an adversary $\mathcal{B}$ that breaks universal one-wayness of $\mathcal{H}$ with non-negligible probability.

$\mathcal{B}$ first chooses an hash input $w \xleftarrow{\$} \{0,1\}^{q(\lambda)}$, and then receives an instance h of $\mathcal{H}$. Next, $\mathcal{B}$ generates $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and $(\widehat{\mathsf{vk}}, \widehat{\mathsf{sk}}) \leftarrow \widehat{\Sigma}.\mathsf{kg}(1^\lambda)$, and runs $\mathcal{A}_p^{\mathcal{O}}(\mathsf{crs})$ simulating the oracle $\mathcal{O}$ on the single query $q := \mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs},\cdot,\cdot)$ asked by $\mathcal{A}_p$. In particular, to answer the query $\mathcal{B}$ uses the secret key $\widehat{\mathsf{sk}}$ to generate $\widehat{\sigma}$, and computes $x = h(w)$ using the function h received from its challenger, and the input w chosen earlier. Notice that such a simulation can be done perfectly in a straightforward way, and that $\mathcal{A}_p$'s output is the pair (y,π) created by $\mathcal{B}$. Next, $\mathcal{B}$ runs the extractor $w' \leftarrow \mathcal{E}(\mathsf{crs}, \mathsf{qt} = (\mathcal{P}(\cdot,\cdot), (\widehat{\sigma}, h, x, \pi))$, and outputs w'.

By correctness of Π it holds that the pair (y,π) returned by $\mathcal{A}_p$ satisfies $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1$. Thus, by our contradiction assumption, with probability $\geq \varepsilon(\lambda)$, $\mathcal{E}$ outputs w' such that $(y, w') \in \mathcal{R}_{\mathcal{H}}$. Namely, $h(w') = x = h(w)$. To show that this is a collision, we argue that, information-theoretically, $w' \neq w$ with probability $\geq 1 - 1/2^\lambda$. This follows from the fact that w is randomly chosen of length $q(\lambda) \geq p(\lambda) + \ell(\lambda) + \lambda$ and the only information about w which is leaked to $\mathcal{E}$ is through π and $x = h(w)$, an information of length at most $p(\lambda) + \ell(\lambda)$. Therefore there are at least λ bits of entropy in w, from which $\Pr[w' = w] \leq 2^{-\lambda}$ over the random choice of w. Hence, $\mathcal{B}$ can break the universal one-wayness of $\mathcal{H}$ with probability $\geq \varepsilon(\lambda) - 2^{-\lambda}$.

□

5.4.3 O-SNARKs for Signing Oracles from SNARKs

O-SNARKs for Signing Oracles from SNARKs in the ROM. In this section, we show that it is possible to "immunize" *any* signature scheme in such a way that any classical SNARK is also an O-SNARK for the signing oracle corresponding to the transformed scheme. The first idea is very simple and consists into applying the hash-then-sign approach using a hash function that will be modeled as a random oracle. A limitation of this result is that, since the verification algorithm uses a random oracle, in all those applications where the SNARK is used to prove knowledge of valid signatures, one would need a SNARK for $\mathsf{NP}^{\mathcal{O}}$. Hence, the best one can do is to conjecture that this still works when replacing the random oracle with a suitable hash function.

Let us now state our result formally. To this end, for any signature scheme Σ and polynomial $Q(\cdot)$ we define $\mathcal{Z}_{Q,\Sigma}$ as the distribution on tuples $\langle \mathsf{vk}, m_1, \sigma_1, \ldots, m_Q, \sigma_Q \rangle$ obtained by running the following probabilistic algorithm:

$\mathcal{Z}_{Q,\Sigma}(1^\lambda)$
  `let` $Q = Q(\lambda)$
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$
  $\tilde{\mathcal{M}} \leftarrow_{\$} \mathsf{MsgSample}(\mathcal{M}, Q)$
  `let` $\tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
  `for` $i = 1$ `to` Q `do` :
    $\sigma_i \leftarrow \mathsf{sign}(\mathsf{sk}, m_i)$
  `return` $\langle \mathsf{vk}, \{m_i, \sigma_i\}_{i=1}^{Q} \rangle$

Figure 5.4: $\mathcal{Z}_{Q,\Sigma}$ auxiliary input distribution

where $\mathsf{MsgSample}(\mathcal{M}, Q)$ is an algorithm that returns Q distinct messages, each randomly chosen from $\mathcal{M}$.

Theorem 5.4.4. *Let Σ be a $\mathsf{UF\text{-}CMA}$-secure signature scheme, and $\mathcal{H}$ be a family of hash functions modeled as a random oracle. Let $\mathcal{U}_n$ be the uniform distribution over strings of length n, and $\mathcal{Z}_{Q,\Sigma}$ be the distribution defined above, where Q is any polynomial in the security parameter. Then there exists a signature scheme $\Sigma_{\mathcal{H}}$ such that every $(\mathcal{Z}, \mathcal{U}, \mathcal{Z}_{\Sigma,Q})$-auxiliary input SNARK Π is a $\mathcal{Z}$-auxiliary input O-SNARK for $(\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})$ where $\mathbb{O}_{\mathcal{H}}$ is a random oracle.*

Proof. The proof consists of two steps. First we define the signature scheme $\Sigma_{\mathcal{H}}$ and prove its security from Σ. Next, we show that Π is an O-SNARK for $(\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})$.

The Signature Scheme $\Sigma_{\mathcal{H}}$. The signature scheme $\Sigma_{\mathcal{H}}$ is essentially an application of the hash-then-sign paradigm to scheme Σ. The only difference is that, instead of hashing only messages, we also hash a short random string of λ bits.

Let $\Sigma = (\Sigma.\mathsf{kg}, \Sigma.\mathsf{sign}, \Sigma.\mathsf{vfy})$ be a signature scheme with message space $\mathcal{M} = \{0,1\}^L$, and let $\mathcal{H}\{H : \{0,1\}^* \to \{0,1\}^L\}$ be a family of hash functions modeled as a random oracle. The signature scheme $\Sigma_{\mathcal{H}} = (\Sigma_{\mathcal{H}}.\mathsf{kg}, \Sigma_{\mathcal{H}}.\mathsf{sign}, \Sigma_{\mathcal{H}}.\mathsf{vfy})$ works as follows:

$\Sigma_{\mathcal{H}}.\mathsf{kg}(1^\lambda)$**:** Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{kg}(1^\lambda)$. Output $(\mathsf{sk}, \mathsf{vk})$.

$\Sigma_{\mathcal{H}}.\mathsf{sign}(\mathsf{sk}, m)$**:** Sample $s \xleftarrow{\$} \{0,1\}^\lambda$, compute $h \leftarrow H(s|m)$, $\widehat{\sigma} \leftarrow \Sigma.\mathsf{sign}(\mathsf{sk}, h)$, and output $\sigma = (s, \widehat{\sigma})$.

$\Sigma_{\mathcal{H}}.\mathsf{vfy}(\mathsf{vk}, m, \sigma)$**:** Parse $\sigma = (s, \widehat{\sigma})$, compute $h \leftarrow H(s|m)$, return the same output of $\Sigma.\mathsf{vfy}(\mathsf{vk}, h, \widehat{\sigma})$.

Lemma 5.4.5. *If Σ is an $\mathsf{UF\text{-}CMA}$-secure signature scheme, so is $\Sigma_{\mathcal{H}}$ in the random oracle model.*

The security proof of the lemma is straightforward and is omitted.

Π is an O-SNARK for $(\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})$. We are going to prove[8] that for every non-uniform polynomial-size oracle adversary $\mathcal{A}$ there exists a non-uniform polynomial-size extractor $\mathcal{E}_{\mathcal{A}}$ and a negligible function $\varepsilon(\lambda)$ such that for every time bound T,

$$\Pr[\mathsf{O\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, (\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})) \Rightarrow 1] \leq \varepsilon(\lambda)$$

As a first step, we show that for every non-uniform polynomial-size algorithm $\mathcal{A}$, making q queries to $\mathcal{O}_{\mathcal{H}}$ and Q queries to $\mathcal{O}_{\Sigma_{\mathcal{H}}}$ (where both $q, Q = \mathsf{poly}(\lambda)$), there is a non-uniform, polynomial-size, non-interactive algorithm $\mathcal{B}$.

$\mathcal{B}$ takes as input (crs, o, z), where $o \leftarrow \mathcal{U}_{q \cdot L + Q \cdot \lambda}$, $z \leftarrow \mathcal{Z}_{Q,\Sigma}$, and these are parsed as follows: $o = \langle r'_1, \ldots, r'_q, s_1, \ldots, s_Q \rangle$ with $r'_j \in \{0,1\}^L$ and $s_i \in \{0,1\}^\lambda$, $z = \langle \mathsf{vk}, r_1, \widehat{\sigma}_1, \ldots, r_Q, \widehat{\sigma}_Q \rangle$.

$\mathcal{B}(\mathsf{crs}, o, z)$

Run $\mathcal{A}^{\mathcal{O}_{\mathcal{H}}, \mathcal{O}_{\Sigma_{\mathcal{H}}}}(\mathsf{crs}) \to (y, \pi)$ and simulate queries to $\mathcal{O}_{\mathcal{H}}$ and $\mathcal{O}_{\Sigma_{\mathcal{H}}}$ as follows:

Query $\mathcal{O}_{\mathcal{H}}(x_j), j \in \{1, \ldots, q\}$:

[8] For simplicity, we are ignoring the additional auxiliary input distribution $\mathcal{Z}$, as it is easy to see that it essentially "carries over".

Answer with $\mathcal{O}_{\mathcal{H}}(x_j) = r'_j$ using r'_j from o
Query $\mathcal{O}_{\Sigma_{\mathcal{H}}}(m_i), i \in \{1, \ldots, Q\}$:
If $\mathcal{O}_{\mathcal{H}}(s_i|m_i)$ has been already answered before, abort.
Else, set $\mathcal{O}_{\mathcal{H}}(s_i|m_i) = r_i$, and answer $\mathcal{O}_{\Sigma_{\mathcal{H}}}(m_i) = (s_i, \widehat{\sigma}_i)$,
where $r_i, \widehat{\sigma}_i$ are taken from z and s_i from o.
Return the same (y, π) returned by $\mathcal{A}$.

The simulation provided by $\mathcal{B}$ to $\mathcal{A}$ is perfect unless $\mathcal{B}$ aborts during a signing query. This event – let us formally call it Abort – happens if there exist $i \in [Q]$ and $j \in [q]$ such that $x_j = s_i|m_i$. However, over the random choice of $s_i \xleftarrow{\$} \{0,1\}^\lambda$ in the i-th query (indeed s_i was never used before and is thus hidden to $\mathcal{A}$), we have that

$$\Pr[\mathsf{Abort}] \leq \sum_{i=1}^{Q} \frac{q}{2^\lambda} \leq \frac{q \cdot Q}{2^\lambda}$$

Hence, if $\mathcal{A}$ succeeds in producing an instance-proof pair (y, π), so does $\mathcal{B}$ with overwhelming probability.

Then, by the adaptive knowledge soundness property, we have that for every such $\mathcal{B}$ there is an extractor $\mathcal{E}_{\mathcal{B}}$ such that for every polynomial n, Q

$$\Pr[\mathsf{O\text{-}KS}(\lambda, T, \mathcal{B}, \mathcal{E}_{\mathcal{B}}, (\mathcal{U}_n, \mathcal{Z}_{Q,\Sigma})) \Rightarrow 1] \leq \varepsilon(\lambda)$$

for a negligible ε.

So far, we have that for every $\mathcal{A}$ there is an extractor $\mathcal{E}_{\mathcal{B}}$. In what follows, we use this $\mathcal{E}_{\mathcal{B}}$ to define the extractor $\mathcal{E}_{\mathcal{A}}$. The extractor $\mathcal{E}_{\mathcal{A}}$ takes as input crs and the transcript

$$\mathsf{qt} = \langle x_1, r'_1, \ldots, x_q, r'_q, r_1, \ldots, r_Q, m_1, s_1, \widehat{\sigma}_1, \ldots, m_Q, s_Q, \widehat{\sigma}_Q \rangle$$

of queries made by $\mathcal{A}$, and proceeds as follows:

$\mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{qt})$
Run $w \leftarrow \mathcal{E}_{\mathcal{B}}(\mathsf{crs}, o, z)$
Return the same w returned by $\mathcal{E}_{\mathcal{B}}$

Basically, it rearranges the data in qt to fulfill the format of (o, z) from distributions $\mathcal{U}_{qL+Q\lambda}, \mathcal{Z}_{Q,\Sigma}$. It is not hard to see that from the above construction we have

$$\begin{aligned} \Pr[\mathsf{O\text{-}KS}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, (\mathbb{O}_{\mathcal{H}}, \mathbb{O}_{\Sigma_{\mathcal{H}}})) \Rightarrow 1] &\leq \Pr[\mathsf{O\text{-}KS}(\lambda, T, \mathcal{B}, \mathcal{E}_{\mathcal{B}}, (\mathcal{U}_n, \mathcal{Z}_{Q,\Sigma})) \Rightarrow 1] + \frac{qQ}{2^\lambda} \\ &\leq \varepsilon(\lambda) + \frac{qQ}{2^\lambda} \end{aligned}$$

□

O-SNARKs for Signing Oracles with Restrictions from SNARKs. In the following we study other ways to obtain O-SNARKs for signing oracles. We give a positive result showing that any SNARK Π is an O-SNARK for the signing oracle of signature scheme Σ if:

(i) the message space of Σ is appropriately bounded (to be polynomially or at most superpolynomially large);

(ii) Π tolerates auxiliary input consisting of the public key of Σ plus a collection of signatures on randomly chosen messages;

(iii) one considers O-SNARK adversaries that query the signing oracle on almost the entire message space.

Furthermore, in case of superpolynomially large message spaces, one needs to assume sub-exponential hardness for Π.

The intuition behind this result is to simulate the O-SNARK adversary by using a (non-interactive) SNARK adversary that receives the public key and a set of signatures on (suitably chosen) messages as its auxiliary input. If these messages exactly match[9] those queried by the O-SNARK adversary, the simulation is perfect. However, since the probability of matching exactly all the $Q = \mathsf{poly}(\lambda)$ queries may decrease exponentially in Q (making the simulation meaningless), we show how to put proper bounds so that the simulation can succeed with probability depending only on the message space size.

More formally, our result is stated as follows. Let Σ be a signature scheme with message space $\mathcal{M}$, and let $Q := Q(\cdot)$ be a function of the security parameter. Let $\mathcal{Z}_{Q,\Sigma}$ be the following auxiliary input distribution

$\mathcal{Z}_{Q,\Sigma}(1^\lambda)$
  $\texttt{let } Q = Q(\lambda)$
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$
  $\tilde{\mathcal{M}} \leftarrow\!\!\$\ \mathsf{MsgSample}(\mathcal{M}, Q)$
  $\texttt{let } \tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
  $\texttt{for } i = 1 \texttt{ to } Q \texttt{ do :}$
    $\sigma_i \leftarrow \mathsf{sign}(\mathsf{sk}, m_i)$
  $\texttt{return } \langle \mathsf{vk}, \{m_i, \sigma_i\}_{i=1}^{Q} \rangle$

Figure 5.5: $\mathcal{Z}_{Q,\Sigma}$ auxiliary input distribution

where $\mathsf{MsgSample}(\mathcal{M}, Q)$ is a probabilistic algorithm that returns a subset $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of cardinality Q chosen according to some strategy that we discuss later. At this point we only assume a generic strategy such that $\delta(|\mathcal{M}|, Q) = \Pr[\mathsf{MsgSample}(\mathcal{M}, Q) = \mathcal{M}^*]$ for any $\mathcal{M}^* \subseteq \mathcal{M}$ of cardinality Q.

Theorem 5.4.6. *Let Σ be a signature scheme with message space $\mathcal{M}$, let $\mathbb{O}_\Sigma$ be the associated family of signing oracles, and let $\mathcal{Z}_{Q,\Sigma}$ be as defined above. If Π is a $\mathcal{Z}_{Q,\Sigma}$-auxiliary input SNARK satisfying (s, t, ε)-adaptive KS, then Π is an O-SNARK for $\mathbb{O}_\Sigma$ satisfying $(s', t', Q, \varepsilon')$-adaptive KS, where $\varepsilon' = \varepsilon/\delta(|\mathcal{M}|, Q)$, $s' = s - O(Q \cdot \log |\mathcal{M}|)$, and $t' = t$.*

Proof. The basic idea for the proof is to show that for any O-SNARK adversary $\mathcal{A}$ against Π there is a non-interactive SNARK adversary $\mathcal{B}_\mathcal{A}$ against Π that can perfectly simulate

[9] We note that the proof requires an exact match and it is not sufficient that the O-SNARK adversary's queries are a subset of the sampled messages. A more precise explanation of this fact is given at the end of the proof.

$\mathcal{A}$, provided that the auxiliary input from $\mathcal{Z}_{Q,\Sigma}$ "matches" the queries made by $\mathcal{A}$. More precisely, for any adversary $\mathcal{A}^{\mathcal{O}}$ making Q oracle queries we define the following adversary $\mathcal{B}_{\mathcal{A}}$ in Figure 5.6.

$\mathcal{B}_{\mathcal{A}}(\mathsf{crs}, \mathsf{z} = \langle \mathsf{vk}, \{m_1, \sigma_i\}_{i=1}^{Q} \rangle)$
  Run $(y, \pi) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{crs})$ and simulate every query $\mathcal{O}(m)$ as follows:
    `let` $\tilde{\mathcal{M}} = \{m_1, \ldots, m_Q\}$
    `if` $m = 0$ `return` vk
    `else if` $m = m_i \in \tilde{\mathcal{M}}$ `return` σ_i
    `else return` $\perp$
  `return` (y, π)

Figure 5.6: Adversary $\mathcal{B}_{\mathcal{A}}$

As one can see, the adversary $\mathcal{B}_{\mathcal{A}}$ fits the syntax of a SNARK adversary in the adaptive knowledge soundness property, by which (if Π is a SNARK) there exists an extractor $\mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$ such that for every time bound T and every Q

$$\Pr\left[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1\right] \leq \varepsilon$$

For every $\mathcal{A}$ we define the extractor $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$, and our goal is to bound

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \leq \varepsilon'$$

To prove this, we proceed by contradiction. Namely, let us assume that there is $\mathcal{A}$ (and thus corresponding $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$) such that:

$$\Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \geq \varepsilon'$$

We define $\mathbf{G}_1$ to be the following variation of experiment $\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}})$: it generates $\mathsf{z}' \leftarrow \mathcal{Z}_{Q,\Sigma}$ (using the same keypair generated when sampling oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{sign}}$), and at the end $\mathbf{G}_1$ outputs 1 if $\mathsf{qt} = \mathsf{z}'$ holds in addition to $\mathsf{Ver}(\mathsf{vst}, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}$. Let $\mathsf{QueryMatch}$ be the event that in $\mathbf{G}_1$ it occurs $\mathsf{qt} = \mathsf{z}'$. Clearly, $\Pr[\mathsf{QueryMatch}] = \delta(Q, |\mathcal{M}|)$, and thus

$$\begin{aligned} \Pr[\mathbf{G}_1 \Rightarrow 1] &= \Pr[\mathsf{QueryMatch}] \cdot \Pr[\mathsf{O\text{-}AdPoK}(\lambda, T, \mathcal{A}, \mathcal{E}_{\mathcal{A}}, \mathbb{O}_{\mathsf{sign}}) \Rightarrow 1] \\ &\geq \delta(Q, |\mathcal{M}|) \cdot \varepsilon' = \varepsilon \end{aligned}$$

To conclude the proof, and reach the contradiction, we show below that

$$\Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1] \geq \Pr[\mathbf{G}_1 \Rightarrow 1]$$

To see this, consider experiment $\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma})$ and let $\mathsf{QueryMatch}'$ be the event that, inside the execution of $\mathcal{B}_{\mathcal{A}}$, all the messages queried by $\mathcal{A}$ coincide with those in z. Since $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$ one can see that experiment $\mathbf{G}_1$ is basically identical to $\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1 \wedge \mathsf{QueryMatch}'$. Namely,

$$\begin{aligned}\Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1] &\geq \Pr[\mathsf{AdPoK}(\lambda, T, \mathcal{B}_{\mathcal{A}}, \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}, \mathcal{Z}_{Q,\Sigma}) \Rightarrow 1 \wedge \mathsf{QueryMatch}'] \\ &= \Pr[\mathbf{G}_1 \Rightarrow 1] \geq \varepsilon\end{aligned}$$

which concludes the proof.

As a final remark, the reason why we need messages in $\tilde{\mathcal{M}}$ to exactly match the queries of $\mathcal{A}$ is that we construct $\mathcal{A}$'s extractor from $\mathcal{B}_{\mathcal{A}}$'s extractor – $\mathcal{E}_{\mathcal{A}} = \mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$. Hence, in order for $\mathcal{E}_{\mathcal{A}}$ to call $\mathcal{E}_{\mathcal{B}_{\mathcal{A}}}$, it needs to know all its inputs, which is not the case if $\mathcal{B}$ has a superset of the signed messages known to $\mathcal{A}$. □

Implications of Theorem 5.4.6. The statement of Theorem 5.4.6 is parametrized by values $|\mathcal{M}|, Q$ and the function $\delta(|\mathcal{M}|, Q)$, which in turn depends on the query guessing strategy. As for the $\mathsf{MsgSample}(\mathcal{M}, Q)$ algorithm, let us consider the one that *samples a random subset* $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ of cardinality Q. For this algorithm we have $\delta(|\mathcal{M}|, Q) = \frac{1}{\binom{|\mathcal{M}|}{Q}}$.

Notice that $\delta(|\mathcal{M}|, Q)$ is governing the success probability of our reduction, and thus we would like this function not to become negligible. However, since $Q = \mathsf{poly}(\lambda)$ is a parameter under the choice of the adversary, it might indeed be the case that $\delta(|\mathcal{M}|, Q) \approx 2^{-Q} \approx 2^{-\lambda}$, which would make our reduction meaningless. To avoid this bad case, we restrict our attention to adversaries for which $Q = |\mathcal{M}| - c$ for some constant $c \geq 1$, i.e., adversaries that ask for signatures on the entire message but a constant number of messages. For this choice of Q we indeed have that $\delta(|\mathcal{M}|, Q) = \frac{1}{|\mathcal{M}|^c}$ depends only on the cardinality of $|\mathcal{M}|$. This gives us the following corollary:

Corollary 5.4.7. *Let Σ be a signature scheme with message space $\mathcal{M}$ where $|\mathcal{M}| = \mathsf{poly}(\lambda)$ (resp. $|\mathcal{M}| = \lambda^{\omega(1)}$), and let $Q = |\mathcal{M}| - c$ for constant $c \in \mathbb{N}$. If Π is a polynomially (resp. sub-exponentially) secure $\mathcal{Z}_{Q,\Sigma}$-auxiliary input SNARK, then Π is an O-SNARK for $\mathbb{O}_\Sigma$ (for adversaries making Q queries).*

5.4.4 O-SNARKs for (Pseudo)random Oracles from SNARKs

In this section, we show a positive result on the existence of O-SNARKs for (pseudo)random oracles, based on classical SNARKs that are assumed to satisfy knowledge soundness with respect to randomly-distributed auxiliary input. While we do not have a direct application of this result, we think that it can be useful and of independent interest.

Let $\mathbb{O} = \{\mathcal{O} : \{0,1\}^\ell \to \{0,1\}^L\}$ be a random oracle family, i.e., a family where sampling a member $\mathcal{O} \leftarrow \mathbb{O}$ consists of sampling a suitably long random tape (of size $2^\ell \cdot L$). Let us stress that here, when we refer to a random oracle family, we do not necessarily consider the random oracle model. We simply consider the case of an interactive game in which $\mathcal{A}$ has oracle access to a random function.

Theorem 5.4.8. *Let $\mathbb{O}$ be a random oracle family. Let $\mathcal{Z}$ be some distribution, $\mathcal{U}_n$ be the uniform distribution over strings of length n, and denote by $(\mathcal{Z},\mathcal{U})$ the distribution over pairs (z, o) such that $\mathsf{z} \leftarrow \mathcal{Z}$ and $o \leftarrow \mathcal{U}$. If Π is a $(\mathcal{Z}_q,\mathcal{U})$-auxiliary input SNARK for every $q = \mathsf{poly}(\lambda)$, then Π is a $\mathcal{Z}$-auxiliary input O-SNARK for $\mathbb{O}$.*

Proof. Completeness and succinctness follow immediately. So, we are left to prove that (adaptive) knowledge soundness implies the corresponding property in the presence of an oracle $\mathcal{O} \leftarrow \mathbb{O}$.

Formally, we have to show that for any efficient oracle prover $\mathcal{A}^{\mathcal{O}}$ there exists an efficient extractor algorithm $\mathcal{E}_{\mathcal{A}}$ such that the joint probability that $\mathcal{A}^{\mathcal{O}}$ outputs a valid proof but $\mathcal{E}_{\mathcal{A}}$ returns a wrong witness is negligible. At a high level, we do this proof by showing that $\mathcal{E}_{\mathcal{A}}$'s existence is equivalent to the existence of a SNARK extractor $\bar{\mathcal{E}}$, which is assured under the adaptive knowledge soundness whenever $\mathcal{A}^{\mathcal{O}}$ exists.

First, for any $\mathcal{A}^{\mathcal{O}}$ we construct another adversary $\bar{\mathcal{A}}$. Precisely, let $\mathcal{A}$ be a non-uniform algorithm that takes as input $(\mathsf{crs}, \mathsf{z})$ and (without loss of generality) makes exactly Q queries to $\mathcal{O}$ for some $Q = \mathsf{poly}(\lambda)$. Then, we can define an adversary $\bar{\mathcal{A}}$ that on input $(\mathsf{crs}, \bar{\mathsf{z}})$ – where $\bar{\mathsf{z}} = (\mathsf{z}, o) \leftarrow (\mathcal{Z}, \mathcal{U}_{Q \cdot L})$ – works as follows:

$\bar{\mathcal{A}}(\mathsf{crs}, \bar{\mathsf{z}})$
  Run $\mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{z}) \rightarrow (y, \pi)$ simulating $\mathcal{O}$ queries as follows:
    given the i-th query $\mathcal{O}(q_i)$, answer with the substring
    $\mathcal{O}(q_i) = a_i = o_{L\cdot(i-1)+1} | \cdots | o_{L \cdot i}$
  Return the same (y, π) returned by $\mathcal{A}$

The simulation of $\mathcal{O}$ provided by $\bar{\mathcal{A}}$ to $\mathcal{A}$ is clearly perfect. Hence, if $\mathcal{A}^{\mathcal{O}}$ succeeds in producing an accepting instance-proof pair (y, π), so does $\bar{\mathcal{A}}$. Therefore, by the adaptive KS property we have that for every such $\bar{\mathcal{A}}$ there is an extractor $\bar{\mathcal{E}}$ that takes the same input of $\bar{\mathcal{A}}$ and outputs a value w such that the probability that $\bar{\mathcal{A}}$ is successful in outputting an accepting pair (y, π) and the value w returned by $\bar{\mathcal{E}}$ is a wrong witness (i.e., $(y, w) \notin \mathcal{R}$) is bounded by a negligible function ε.

Basically, above we showed that for every $\mathcal{A}^{\mathcal{O}}$ there exists an extractor $\bar{\mathcal{E}}$. As a last step, we use this $\bar{\mathcal{E}}$ to show the existence of the extractor $\mathcal{E}_{\mathcal{A}}$.

For simplicity, parse the transcript of queries qt made by $\mathcal{A}^{\mathcal{O}}$ as a pair $(\vec{q}, \vec{a})$, where $\vec{q}$ are all the queries and $\vec{a}$ all the corresponding answers, in order. Then we define the extractor $\mathcal{E}_{\mathcal{A}}$ in the following way.

$\mathcal{E}_{\mathcal{A}}(\mathsf{crs}, \mathsf{z}, \mathsf{qt})$
  Run $w \leftarrow \bar{\mathcal{E}}(\mathsf{crs}, \bar{\mathsf{z}})$ where $\bar{\mathsf{z}} = (\mathsf{z}, \vec{a})$
  Return the same w returned by $\bar{\mathcal{E}}$

It is easy to see that, except for some syntactic changes, $\mathcal{E}_{\mathcal{A}}$ is the same as $\bar{\mathcal{E}}$. Combining all the above arguments, we have that for every $\mathcal{A}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that adaptive knowledge soundness for $\mathbb{O}$ holds with probability ε. □

5.4.4.1 The Case of Pseudorandom Functions

As an interesting corollary of Theorem 5.4.8, we show that a similar result holds even for the case in which adversaries get access to a pseudorandom function as an oracle.

Corollary 5.4.9. *Let $\mathsf{F} : \{0,1\}^{\kappa} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{L}$ be a family of pseudorandom functions. Let $\mathbb{O}_{\mathsf{F}}$ be the family of oracles $\mathcal{O}_{\mathsf{F}}(x) = \mathsf{F}_K(x)$, where every family member $\mathcal{O}$ is described by a seed K of the pseudorandom function, and thus the process $\mathcal{O}_{\mathsf{F}} \leftarrow \mathbb{O}_{\mathsf{F}}$ corresponds to sampling a random seed $K \xleftarrow{\$} \{0,1\}^{\kappa}$. Let $\mathcal{Z}$ be some distribution, $\mathcal{U}_n$ be the uniform*

distribution over strings of length n, and denote by $(\mathcal{Z},\mathcal{U})$ the distribution over pairs (z, o) such that z follows $\mathcal{Z}$ and o follows $\mathcal{U}$.

If Π is a $(\mathcal{Z}_q,\mathcal{U})$-auxiliary input adaptive SNARK for every $q = \mathsf{poly}(\lambda)$, and F is pseudorandom, then Π is a $\mathcal{Z}$-auxiliary input O-SNARK for $\mathbb{O}_{\mathsf{F}}$.

Proof. The proof of the corollary follows in two steps. The first step relies directly on Theorem 5.4.8 to see that Π is an O-SNARK for the family $\mathbb{O}$ of random oracles. This means that for every $\mathcal{A}^{\mathcal{O}}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$ such that adaptive knowledge soundness for $\mathbb{O}$ holds with probability δ.

The second step consists, informally, in replacing a random oracle with a PRF oracle and then showing that the success probability cannot change too much, unless the function is not pseudorandom.

For every $\tilde{\mathcal{A}}^{\mathcal{O}_{\mathsf{F}}}$, we can construct another adversary $\mathcal{A}^{\mathcal{O}}$ that simply runs $\tilde{\mathcal{A}}^{\mathcal{O}_{\mathsf{F}}}$, simulates all queries using its oracle, and returns the same output of $\tilde{\mathcal{A}}$. By adaptive knowledge soundness for $\mathbb{O}$ there is an extractor $\mathcal{E}_{\mathcal{A}}$. We then define the extractor $\mathcal{E}_{\tilde{\mathcal{A}}} = \mathcal{E}_{\mathcal{A}}$.

Let us now consider the adaptive KS experiment with $\tilde{\mathcal{A}}^{\mathcal{O}_{\mathsf{F}}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$ and let $\tilde{\delta}$ be the probability that the final condition holds. It is easy to prove that by the pseudorandomness of F there is a negligible function ν such that $\tilde{\delta} \leq \delta + \nu$. □

Chapter 6

Applications of O-SNARK

Malleable Signature Schemes based on O-SNARKs. We have shown in the previous chapters the necessity of defining SNARK schemes with knowledge soundness in the presence of oracles. Studying such a notion was motivated by the various applications of SNARKS in larger protocols where the provers are given access to oracles with a secret state. We have now the appropriate tool to use in these settings, our O-SNARK. We illustrate in this chapter several useful applications where our O-SNARK is very helpful: homomorphic signatures, succinct functional signatures, SNARKs on authenticated data and aggregate signatures. In the case of aggregate signatures, we further examine different ways to construct such schemes, ones that require O-SNARKs and other weaker versions that can be instantiated only with classical SNARKs. We develop non-standard proof techniques for our aggregator scheme in order to bypass the impossibility results we have seen in the previous chapter.

Contents

Tale four: A crypto reputation

After some years, teenager Alice, now in high-school continues to feed her interest in cryptography, reading a lot on the subject. She seems to be always updated with all the novelties in cryptography.
Alice has gained notoriety in her high-school for being a crypto "geek", she is known for extensively using crypto in her life to secure all her digital interactions. Some rumors say that she is a very skilled hacker as well.
One day, she is called for in the principal office. The principal has remarked that there were problems with the school database. Some students seemed to have modified their grades... Alice finds this story suspicious, is that an allusion from the principal? Is she the main suspect of such a violation? Maybe she needs to explain to the principal that being interested in cryptography does not transform you into a hacker...
She is a little anxious, but she continues to listen what the principal has to say to her.
The principal explains that the school tried to avoid this kind of attacks by using digital signatures, that make tampering with the signed data impossible. Each professor has now to sign the grades before uploading them. Anybody who downloads a student grade obtains the signatures associated and can verify if this is authentic and the grade was not modified.
However, this solution comes with a drawback. Before, parents and professors were able to compute statistics or means on the data directly on the database and immediately retrieve only the result. Now, the signing also limits this utility of the database: Instead, one has to download all the grades together with their signatures, check the authenticity of each pair grade-signature and then compute the average on the (now certified) values.
The parents are complaining that these signatures are too big to download and store and it requires a lot of time to check them individually before computing the statistics on the grades.
The principal, having explained the difficulties the school encounters, asks Alice for an advice. He tells her that she got very respected in the high-school for her interest in cryptography and that her acquirements are valuable and she can now make use of her knowledge to help the school cope with this new problem.
Alice is very happy with this outcome of her visit in the principal office.
She immediatelly offers to help with the task. She has heard of malleable signatures, that allow to compute on the signed messages and obtain the signed result directly. This final signature on the result can attest that the computation was made on inputs legitimately signed by the teachers.
In order to make this resulting signature succinct and fast to verify, Alice thinks of using SNARKs as a building block of the malleable signature schemes.
Alice has a new homework that finally she is eager to solve!

6.1 Introduction

We showed in the previous chapters the necessity of defining SNARK schemes with knowledge soundness in the presence of oracles. Studying such a notion was motivated by the various applications of these schemes where the provers are given access to oracles with a secret state. In this protocols, one has to run an extractor during the security experiment (e.g., a security reduction) where the secret state of the oracle is not available. We have now the right tool to use in these settings, our O-SNARK. We illustrate in this chapter several useful applications where our O-SNARK is very helpful.

We recognize the need of using O-SNARKs while trying to formally prove the security of a "folklore" construction of homomorphic signatures based on SNARKs and digital signatures that is mentioned in several papers (e.g., [BF11a, GW13, CF13, GVW15]).

The same problem of black-box-access appears in a generic construction of SNARKs on authenticated data in [BBFR15] (also informally discussed in [BCCT12]), where the security proof uses the existence of an extractor for the oracle-aided prover, but without giving particular justification.

A similar issue also appears in the construction of succinct functional signatures of [BGI14]. To be precise, in [BGI14] the authors provide a proof but under a stronger definition of SNARKs where the adversarial prover and the extractor are independent PPT machines without *common* auxiliary input: a notion for which we are not aware of standard model constructions. In contrast, when attempting to prove the succinct functional signatures of [BGI14] using the classical definition of SNARKs, one incurs in the same issue illustrated above.

6.2 Homomorphic Signature

In this section, we show three applications of O-SNARKs for building homomorphic signatures [BF11a], succinct functional signatures [BGI14], and SNARKs on authenticated data [BBFR15].

Generally speaking, our results show constructions of these primitives based on a signature scheme HomSig and a succinct non-interactive argument Π, and show their security by assuming that Π is an O-SNARK for signing oracles corresponding to HomSig. Once these results are established, we can essentially reach the following conclusions about the possible secure instantiations of these constructions. First, one can instantiate them by using Micali's CS proofs as O-SNARK (cf. Section 5.4.1): this solution essentially yields secure instantiations in the random oracle model that work with a specific proof system (perhaps not the most efficient one in practice). Second, one can instantiate them with a classical SNARK and a hash-and-sign signature scheme (cf. Section 5.4.3), and conjecture that replacing the random oracle with a suitable hash function preserves the overall security. Third, one can instantiate the constructions using a classical SNARK construction Π and signature scheme HomSig, and then conjecture that Π is an O-SNARK with respect to the family of signing oracles corresponding to HomSig. Compared to the first solution, the last two ones have the advantage that one could use some of the recently proposed efficient SNARK schemes (e.g., [PHGR13, BCG+13]; on the other hand these solutions have the drawback that the security of the instantiations would be heavily based on a heuristic argument. Finally, a fourth option that we provide are security proofs of these primitives which consider only

non-adaptive adversaries (i.e., adversaries that declare all their queries in advance). In this case, we can prove security based on non-adaptive O-SNARKs, and thus based on classical SNARKs (applying our Theorem 5.3.3). The advantage of this fourth option is that one obtains a security proof for these instantiations based on classical, not new, assumptions, although the proof holds only for a much weaker security notion.

6.2.1 Homomorphic Signature Scheme Definition

As the first application of O-SNARKs we revisit a "folklore" construction of homomorphic signatures from SNARKs. This construction has been mentioned several times in the literature (e.g., [BF11a, GW13, CF13, CFW14, GVW15]) and is considered as the 'straightforward' approach for constructing this primitive. In this section, we formalize this construction, and notice that its security proof is quite subtle as one actually incurs the extraction issues that we mentioned in Section 5.4.2. Namely, one needs to run an extractor in an interactive security game in the presence of a signing oracle. Here we solve this issue by giving a simple proof based on our notion of O-SNARKs (for families of signing oracles).

Definition of Homomorphic Signatures. We begin by recalling the definition of homomorphic signatures. The definition below can be seen as the public key version of the notion of homomorphic message authenticators for labeled programs of Gennaro and Wichs [GW13].

Labeled Programs [GW13]. A labeled program consists of a tuple $\mathcal{P} = (F, \tau_1, \dots \tau_n)$ such that $F : \mathcal{M}^n \to \mathcal{M}$ is a function on n variables (e.g., a circuit), and $\tau_i \in \{0,1\}^\ell$ is the label of the i-th variable input of F. Let $F_{id} : \mathcal{M} \to \mathcal{M}$ be the canonical identity function and $\tau \in \{0,1\}^\ell$ be a label. We consider $\mathcal{I}_\tau = (F_{id}, \tau)$ as the identity program for input label τ. Given t labeled programs $\mathcal{P}_1, \dots \mathcal{P}_t$ and a function $G : \mathcal{M}^t \to \mathcal{M}$, the composed program $\mathcal{P}^*$ is the one obtained by evaluating G on the outputs of $\mathcal{P}_1, \dots \mathcal{P}_t$, and is compactly denoted as $\mathcal{P}^* = G(\mathcal{P}_1, \dots \mathcal{P}_t)$. The labeled inputs of $\mathcal{P}^*$ are all distinct labeled inputs of $\mathcal{P}_1, \dots \mathcal{P}_t$, i.e., all inputs with the same label are grouped together in a single input of the new program.

Definition 6.2.1 (Homomorphic Signatures for Labeled Programs)**.** *A homomorphic signature scheme* HomSig *is a tuple of probabilistic, polynomial-time algorithms* $(\mathsf{HomKG}, \mathsf{HomSign}, \mathsf{HomVer}, \mathsf{HomEval})$ *that work as follows:*

$\mathsf{HomKG}(1^\lambda)$ takes a security parameter λ and outputs a public key VK and a secret key SK. The public key VK defines implicitly a message space $\mathcal{M}$, the label space $\mathcal{L}$, and a set $\mathcal{F}$ of admissible functions.

$\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ takes a secret key SK, a (unique) label $\tau \in \mathcal{L}$ and a message $m \in \mathcal{M}$, and it outputs a signature σ.

$\mathsf{HomEval}(\mathsf{VK}, F, (\sigma_1, \dots \sigma_n))$ takes a public key VK, a function $F \in \mathcal{F}$ and a tuple of signatures $(\sigma_1, \dots \sigma_n)$. It outputs a new signature σ.

$\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}, m, \sigma)$ takes a public key VK, a labeled program $\mathcal{P} = (F, (\tau_1 \dots \tau_n))$ with $F \in \mathcal{F}$, a message $m \in \mathcal{M}$, and a signature σ. It outputs either 0 (reject) or 1 (accept).

and satisfy *authentication correctness*, *evaluation correctness*, *succinctness*, and *security*, as described below.

- **Authentication Correctness.** Informally, we require that signatures generated by $\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ verify correctly for m as the output of the identity program $\mathcal{I} = (F_{id}, \tau)$. Formally, HomSig has authentication correctness if for all key pairs $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$, any label $\tau \in \mathcal{L}$, message $m \in \mathcal{M}$, and any signature $\sigma \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$, $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I} = (F_{id}, \tau), m, \sigma)$ outputs 1 with all but negligible probability.

- **Evaluation Correctness.** Intuitively, we require that running the evaluation algorithm on signatures $(\sigma_1, \dots \sigma_n)$, where σ_i is a signature for m_i on label τ_i, produces a signature σ which verifies for $F(m_1, \dots m_n)$. Formally, fix a key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda, \mathcal{L})$, a function $G : \mathcal{M}^t \to \mathcal{M}$ and any set of program/message/signature triples $\{(\mathcal{P}_i, m_i, \sigma_i)\}_{i=1\dots t}$ such that $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}_i, m_i, \sigma_i) = 1$. If $m^* = G(m_1 \dots m_t)$, $\mathcal{P}^* = G(\mathcal{P}_1, \dots \mathcal{P}_t)$ and $\sigma^* = \mathsf{HomEval}(\mathsf{VK}, G, (\sigma_1, \dots, \sigma_t))$, then $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \sigma^*) = 1$ holds with all but negligible probability.

- **Succinctness.** For every large enough security parameter $\lambda \in \mathbb{N}$, there is a polynomial $p(\cdot)$ such that for every $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$ the output size of $\mathsf{HomSign}$ and $\mathsf{HomEval}$ is bounded by $p(\lambda)$ for any choice of their inputs.

- **Security.** A homomorphic signature scheme HomSig is secure if for every PPT adversary $\mathcal{A}$ there is a negligible function ε such that $\Pr[\mathsf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda) = 1] \leq \varepsilon(\lambda)$ where the experiment $\mathsf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda)$ is described in the following:

  Key generation: Run $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{HomKG}(1^\lambda)$ and give VK to $\mathcal{A}$.

  Signing queries: $\mathcal{A}$ can adaptively submit queries of the form (τ, m), where $\tau \in \mathcal{L}$ and $m \in \mathcal{M}$. The challenger initializes an empty list T and proceeds as follows:

  - If (τ, m) is the first query with label τ, then the challenger computes $\sigma \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$, returns σ to $\mathcal{A}$ and updates the list of queries $T \leftarrow T \cup \{(\tau, m)\}$.
  - If $(\tau, m) \in T$ (i.e., the adversary had already queried the tuple (τ, m)), then the challenger replies with the same signature generated before.
  - If T contains a tuple (τ, m_0) for some different message $m_0 \neq m$, then the challenger ignores the query.

  Note that each label τ can be queried only once.

  Forgery: After the adversary is done with the queries of the previous stage, it outputs a tuple $(\mathcal{P}^*, m^*, \sigma^*)$. Finally, the experiment outputs 1 iff the tuple returned by the adversary is a forgery (as defined below).

  Forgeries are tuples $(\mathcal{P}^* = (F^*, (\tau_1^*, \dots \tau_n^*)), m^*, \sigma^*)$ such that $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \sigma^*) = 1$ and they satisfy one the following conditions:

  - *Type 1 Forgery:* There is $i \in [n]$ such that $(\tau_i^*, \cdot) \notin T$ (i.e., no message m has ever been signed w.r.t. label τ_i^* during the experiment).
  - *Type 2 Forgery: All* labels τ_i^* have been queried—$\forall i \in [n], (\tau_i^*, m_i) \in T$—but $m^* \neq F^*(m_1, \dots m_n)$ (i.e., m^* is not the correct output of the labeled program $\mathcal{P}^*$ when executed on the previously signed messages $(m_1, \dots m_n)$).

Context-Hiding. Another property which is useful for homomorphic signatures is context-hiding. Intuitively, this property says that a signature on the output of a function does not reveal anything about its inputs, beyond what can be trivially learned by the verifier. Here we recall a (statistical) version of the definition proposed in [BF11a] (also adapted to our syntax).

Definition 6.2.2 (Weak Context-Hiding)**.** *A homomorphic signature scheme* HomSig *is weakly context hiding if there exists a PPT simulator* $S = (S^{\mathsf{KG}}, S^{\mathsf{Eval}})$ *such that, for any fixed choice of function* F*, tuple of messages* $m_1, \ldots, m_n \in \mathcal{M}$*, set of labels* $\tau_1, \ldots, \tau_n \in \mathcal{L}$*, it holds that for any distinguisher* $\mathcal{D}$ *and large enough security parameter* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\begin{array}{l|r} \mathcal{D}(\mathsf{VK}, \mathsf{SK}, \{\sigma_i\}_{i\in[n]}, \bar{\sigma}) = 1 & (\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{HomKG}(1^\lambda) \\ & \sigma_i \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau_i, m_i) \forall i \in [n] \\ & \bar{\sigma} \leftarrow \mathsf{HomEval}(\mathsf{VK}, F, (\sigma_1, \ldots \sigma_n)) \end{array}\right] -$$

$$\Pr\left[\begin{array}{l|r} \mathcal{D}(\mathsf{VK}, \mathsf{SK}, \{\sigma_i\}_{i\in[n]}, \bar{\sigma}) = 1 & (\mathsf{VK}, \mathsf{SK}) \leftarrow S^{\mathsf{KG}}(1^\lambda) \\ & \sigma_i \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau_i, m_i) \forall i \in [n] \\ & \bar{\sigma} \leftarrow S^{\mathsf{Eval}}(\mathcal{P}, F(m_1, \ldots, m_n)) \end{array}\right] \leq negl(\lambda)$$

Note that since the definition holds for fixed $F, \{m_i\}_i, \{\tau_i\}_i$, these values can also be given to $\mathcal{D}$. The notion is called *weak* context-hiding in contrast to a *strong* notion where one can also hide the fact that a homomorphic evaluation took place.

6.2.2 Homomorphic Signatures from O-SNARKs.

To build the homomorphic signature, we use a regular signature scheme Σ (cf. Definition 2.4.7) and a fully-succinct O-SNARKΠ for NP. The resulting scheme is homomorphic for all functions F whose running time is upper bounded by some fixed polynomial $t_{\mathcal{F}}(\cdot)$, and the scheme is 1-hop, i.e., it is not possible to apply $\mathsf{HomEval}$ on signatures obtained from other executions of $\mathsf{HomEval}$.[1]

Defining the Machine $M_{\Sigma,F}$**.** Let Σ be a signature scheme, and F be the description of a function $F : \mathcal{X}^n \to \mathcal{X}$ where $\mathcal{X}$ is some appropriate domain (e.g., $\mathcal{X} = \{0,1\}^\mu$). Then $M_{\Sigma,F}(x, w)$ is the random-access machine that works as follows. It takes inputs (x, w) where values x are of the form $x = (\mathsf{vk}, m, \tau_1, \ldots, \tau_n)$ where vk is a public key of the scheme Σ, $m \in \mathcal{X}$ is a message and $\tau_i \in \{0,1\}^\ell$ are labels, for $1 \leq i \leq n$. The values w are instead tuples $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$ where for every $i \in [n]$, $m_i \in \mathcal{X}$ is a message and σ_i is a signature of the scheme Σ. On input such a pair (x, w), $M_{\Sigma,F}(x, w)$ accepts iff

$$m = F(m_1, \ldots, m_n) \ \wedge \ \mathsf{vfy}(\mathsf{vk}, \tau_i | m_i, \sigma_i) = 1, \forall i = 1, \ldots, n$$

Associated to such machine there is also a polynomial time bound $t_{\Sigma,\mathcal{F}}(k) = k^{e_{\Sigma,\mathcal{F}}}$, such that $M_{\Sigma,F}$ rejects if it does more than $t_{\Sigma,F}(|x|)$ steps. Finally, we note that given a polynomial bound $t_{\mathcal{F}}(k) = k^{e_{\mathcal{F}}}$ on the running time of every F supported by the scheme, a polynomial bound $t_\Sigma(k) = k^{e_\Sigma}$ on the running time of Σ's verification algorithm, and values n, μ, ℓ, one can efficiently deduce the constant exponent $e_{\Sigma,\mathcal{F}}$ for the time bound $t_{\Sigma,\mathcal{F}}(|x|) = |x|^{e_{\Sigma,\mathcal{F}}}$.

[1] Previous work hinted the possibility of achieving multi-hop homomorphic signatures by using SNARKs with recursive composition. However, given the issues we already notice in using classical SNARKs, it is unclear to us whether such a multi-hop construction would allow for a proof.

We call $\mathcal{R}_\Sigma$ the NP binary relation consisting of all pairs (y, w) such that, parsing $y = (M_{\Sigma,F}, x, t)$, $M_{\Sigma,F}(x, w)$ accepts in at most t steps and $t \leq t_{\Sigma,\mathcal{F}}(|x|)$. To see this observe that the running time of M_Σ is bounded by

$$\begin{aligned} t_M &\leq t_{\mathcal{F}}(n \cdot \mu) + n \cdot t_\Sigma(\lambda + \ell + \mu) = (n \cdot \mu)^{e_{\mathcal{F}}} + n \cdot (\lambda + \ell + \mu)^{e_\Sigma} \\ &\leq (\mu)^{(C+1)e_{\mathcal{F}}} + \mu^C \cdot (\lambda + \ell + \mu)^{e_\Sigma} \\ &\leq |x|^{(C+1)e_{\mathcal{F}}} + |x|^{C+e_\Sigma} \\ &\leq |x|^{e_{\Sigma,\mathcal{F}}} \end{aligned}$$

where the second equation follows from $n \leq \mu^C$, the third equation follows by observing that $\mu \leq |x| = \lambda^c + |F| + \mu + n \cdot \ell$, and finally the last one is given by choosing $e_{\Sigma,\mathcal{F}} = \max((C+1)e_{\mathcal{F}}, C + e_\Sigma) + 1$.

The Construction. Let $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ be a signature scheme and $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ be a fully-succinct O-SNARK for NP. The homomorphic signature scheme $\mathsf{HomSig}[\Sigma, \Pi]$ is defined as follows.

$\mathsf{HomKG}(1^\lambda)$**:** Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$. Define $\mathsf{SK} = \mathsf{sk}$ and $\mathsf{VK} = (\mathsf{vk}, \mathsf{crs})$. Let the message be $\mathcal{M} = \{0,1\}^\mu$ and the label space be $\mathcal{L} = \{0,1\}^\ell$. Output $(\mathsf{SK}, \mathsf{VK})$.

$\mathsf{HomSign}(\mathsf{SK}, \tau, m)$**:** Run $\sigma \leftarrow \mathsf{sign}(\mathsf{sk}, \tau|m)$. Output $\bar{\sigma} = (\mathit{signature}, (\tau, m, \sigma))$.

$\mathsf{HomEval}(\mathsf{VK}, m, F, (\bar{\sigma}_1, \ldots, \bar{\sigma}_n))$**:** Parse every $\bar{\sigma}_i = (\mathit{signature}, (\tau_i, m_i, \sigma_i))$, compute $m = F(m_1, \ldots, m_n)$, reconstruct an instance $y = (M_{\Sigma,F}, x, t)$ where $x = (\mathsf{vk}, m, \tau_1, \ldots, \tau_n)$ and $t = |x|^{e_{\Sigma,\mathcal{F}}}$, and the witness $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$. Finally, run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, y, w)$ and output $\bar{\sigma} = (\mathit{proof}, \pi)$.

$\mathsf{HomVer}(\mathsf{VK}, \mathcal{P} = (F, (\tau_1, \ldots \tau_n)), m, \bar{\sigma})$**:** Parse the signature $\bar{\sigma} = (\mathit{flag}, \cdot)$ and output the bit b computed as follows:
If $\bar{\sigma} = (\mathit{signature}, (\tau, m, \sigma))$ and $\mathcal{P} = \mathcal{I} = (F_{id}, \tau)$ run $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma) \to b$.
If $\bar{\sigma} = (\mathit{proof}, \pi)$ run $\mathsf{Ver}_{e_{\Sigma,\mathcal{F}}}(\mathsf{crs}, y, \pi) \to b$ where $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m, \tau_1, \ldots, \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$.
Recall that in a SNARK for NP, Ver_c is given a constant $c > 0$ and only works for relation $\mathcal{R}_c$.

In what follows we show that the scheme above is a homomorphic signature.

Authentication Correctness. For the present scheme the definition asks that signatures $\bar{\sigma} = (\mathit{signature}, (\tau, m, \sigma))$ as generated by $\mathsf{HomSign}(\mathsf{SK}, \tau, m)$ verify correctly for m as the output of the identity program $\mathcal{I} = (F_{id}, \tau)$. Observe that for all key pairs $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$, any label $\tau \in \mathcal{L}$ and any $\bar{\sigma} \leftarrow \mathsf{HomSign}(\mathsf{SK}, \tau, m)$, the algorithm $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I} = (F_{id}, \tau), \bar{\sigma})$ performs the verification by running $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma)$. By the correctness of the signature scheme Σ, for any signature $\sigma \leftarrow \mathsf{sign}(\mathsf{sk}, \tau|m)$, $\mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma)$ outputs 1 with all but negligible probability.

Therefore authentication correctness of $\mathsf{HomSig}(\Sigma, \Pi)$ follows.

Evaluation Correctness. To see evaluation correctness, fix a key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{HomKG}(1^\lambda)$, and a set of triples of the form $\{(\tau_i, m_i, \bar{\sigma}_i)\}_{i=1\ldots n}$ where $\bar{\sigma}_i = (\mathit{signature}, (\tau_i, m_i, \sigma_i))$ and for which $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I}_i = (F_{id}, \tau_i), m_i, \bar{\sigma}_i) = 1$.
Let $\bar{\sigma}^* \leftarrow \mathsf{HomEval}(\mathsf{VK}, F^*, (\bar{\sigma}_1, \ldots, \bar{\sigma}_n))$, where $\bar{\sigma}^* = (\mathit{proof}, \pi)$ with π a proof generated by $\mathsf{Prove}(\mathsf{prs}, y, w)$ by setting $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^*, \tau_1, \ldots, \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$ and $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$.

If $m^* = F^*(m_1 \dots m_n)$ (for an F^* that is in the class of supported functions) and since $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I}_i, m_i, \bar{\sigma}_i)$ means $\mathsf{vfy}(\mathsf{vk}, \tau_i | m_i, \sigma_i) = 1$, we have that $(y, w) \in \mathcal{R}_\Sigma$. Thus $\mathsf{HomVer}(\mathsf{VK}, \mathcal{P}^*, m^*, \bar{\sigma}^*) = 1$ holds with all but negligible probability by the correctness of the SNARK $\mathsf{Ver}_{e_{\Sigma,\mathcal{F}}}(\mathsf{crs}, y, \pi) = 1$.

Succinctness. As the output of the $\mathsf{HomEval}$ algorithm is a proof π obtained by running the Prove algorithm, the succinctness of $\mathsf{HomSign}(\Sigma, \Pi)$ follows from the succinctness of Π.

Security. As in Section 5.4.2, for every signature scheme $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ we denote by $\mathbb{O}_\Sigma$ the family of oracles $\mathcal{O}(m) = \mathsf{sign}(\mathsf{sk}, m)$ (where the verification key is returned as output of a special query $\mathcal{O}(\text{`}vk')$). We show the security of the scheme $\mathsf{HomSig}[\Sigma, \Pi]$ via the following theorem.

Theorem 6.2.3. *Let Σ be a signature scheme, and $\mathbb{O}_\Sigma$ be its corresponding signing oracle family as described above. If Π is an O-SNARK for $\mathbb{O}_\Sigma$, and Σ is* UF-CMA*-secure, then $\mathsf{HomSig}[\Sigma, \Pi]$ is a secure homomorphic signature scheme.*

Proof. To prove that this scheme is a secure homomorphic signature, we assume by contradiction the existence of an efficient adversary $\mathcal{A}$ that is able to output a forgery of one of the two types with non-negligible probability δ. Starting from this algorithm $\mathcal{A}$, we further construct a successful forger $\mathcal{B}$ against the UF-CMA-security of Σ, which leads to a contradiction. Along the way of this reduction, we also rely on the fact that Π is an O-SNARK satisfying adaptive knowledge soundness for $\mathbb{O}_\Sigma$.

By looking at the definition of the security experiment $\mathsf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A},\mathsf{HomSig}}(\lambda)$ for the HomSig scheme, adversary $\mathcal{A}$ is almost equivalent to an O-SNARK adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ for oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{sign}}$. Stated more formally, for every adversary $\mathcal{A}$ against HomSig that outputs a forgery $\mathcal{P}^* = (F^*, (\tau_1^*, \dots \tau_n^*)), m^*, \bar{\sigma}^* = (proof, \pi)$, it is possible to construct another adversary $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$ working as follows: it queries $\mathsf{vk} \leftarrow \mathcal{O}(\text{`}vk')$; it runs $\mathcal{A}^{\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK} = (\mathsf{vk}, \mathsf{crs}))$ simulating $\mathcal{A}$'s oracle queries using its own oracle $\mathcal{O}$; finally it returns the value (y, π), where $y = (M_{\Sigma,F^*}, x, |x|^{e_{\Sigma,\mathcal{F}}})$, with $x = (\mathsf{vk}, m^*, \tau_1^*, \dots, \tau_n^*)$, is obtained from $\mathcal{A}$'s output. The adversary $\tilde{\mathcal{A}}$ perfectly fits the O-SNARK definition by which we know that there exists an extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that, given the same input of $\tilde{\mathcal{A}}^{\mathcal{O}}$ and the transcript of oracle queries/answers made and received by $\tilde{\mathcal{A}}^{\mathcal{O}}$, outputs a correct witness w (i.e., such that $(y, w) \in \mathcal{R}_\Sigma$) with all but negligible probability.

Hence, we have that for every successful adversary $\mathcal{A}$ against HomSig there exists extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$, that takes the very same input of $\mathcal{A}$ (plus, the list of oracle answers). Starting from this adversary $\mathcal{A}$, we construct the forger $\mathcal{B}^{\mathcal{O}}$ that breaks the UF-CMA security of Σ. We build $\mathcal{B}$ (which gets the public key vk and can make queries to $\mathcal{O} = \mathsf{sign}(\mathsf{sk}, \cdot)$ oracle) as follows:

$\mathcal{B}^{\mathcal{O}}(\mathsf{vk})$:
Initialize $\mathsf{qt} \leftarrow (\text{`}vk', \mathsf{vk})$
Generate $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and run $\mathcal{A}(\mathsf{VK} = (\mathsf{crs}, \mathsf{vk}))$
Simulate queries (τ, m) to $\mathcal{O}$ as follows:
  query $\sigma \leftarrow \mathcal{O}(\tau | m)$ and add $(\tau | m, \sigma)$ to qt
  output σ
When $\mathcal{A}$ outputs $(\mathcal{P}^* = (F^*, (\tau_1^*, \dots \tau_n^*)), m^*, \bar{\sigma}^*)$ parse $\bar{\sigma}^* = (proof, \pi^*)$
Run $\mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$ and obtain the witness $w = (m_1^*, \sigma_1^*, \dots, m_n^*, \sigma_n^*)$
Check that $(y, w) \in \mathcal{R}_\Sigma$, i.e., that $M_\Sigma(x, w) = 1$ in at most $|x|^{e_{\Sigma,\mathcal{F}}}$ steps
where $M_\Sigma(x, w) := (m^* = F(m_1^*, \dots, m_n^*) \ \wedge \ \mathsf{vfy}(\mathsf{vk}, \tau_i^* | m_i^*, \sigma_i^*) = 1, \forall i \in [n])$

[Fail] Abort if $(y, w) \notin \mathcal{R}_\Sigma$.
Else proceed:
[Type 1] If $\exists j \in [n]$ such that $(\tau_j^*|\cdot,\cdot) \notin \mathsf{qt}$ return $(\tau_j^*|m_j^*, \sigma_j^*)$
[Type 2] If $\forall i \in [n]$ there is a tuple $(\tau_i^*|m_i, \sigma_i) \in \mathsf{qt}$
and there $\exists j \in [n]$ such that $m_j^* \neq m_j$ return $(\tau_j^*|m_j^*, \sigma_j^*)$

Let us now show that whenever $\mathcal{A}$ succeeds in the simulation described above, $\mathcal{B}$ is also successful in breaking the UF-CMA security of Σ, unless the "Fail" event happens.

First, let us condition on the event that $\mathcal{B}$ does not abort, i.e., "Fail" does not occur, that is the extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ is correct in returning a valid witness. If $\mathcal{A}$ outputs the first type of forgery, since the witness w is valid— $\mathsf{vfy}(\mathsf{vk}, \tau_i^*|m_i^*, \sigma_i^*) = 1, \forall i = 1, \ldots, n$— it follows that the signature forgery $(\tau_j^*|m_j^*, \sigma_j^*)$ is a valid one.

If the forgery returned by $\mathcal{A}$ is of the second type, recall this means that $m^* \neq F(m_1, \ldots, m_n)$ where all inputs of F are the ones in the transcript qt, i.e., qt contains the tuples $(\tau_1^*|m_1, \sigma_1), \ldots (\tau_n^*|m_n, \sigma_n)$. Combining this with the validity of w we have that $F(m_1^*, \ldots, m_n^*) = m^* \neq F(m_1, \ldots, m_n)$, and thus $(m_1^*, \ldots, m_n^*) \neq (m_1, \ldots, m_n)$. This means that there exists at least one j such that $(\tau_j^*|m_j, \sigma) \in \mathsf{qt}$ but $m_j^* \neq m_j$. Moreover, by witness validity it also holds $\mathsf{vfy}(\mathsf{vk}, \tau_j^*|m_j^*, \sigma_j^*) = 1$. Thus we can conclude that, even in this case, $(\tau_j^*|m_j^*, \sigma_j^*)$ is a valid forgery for Σ.

So far, we have proved that whenever the adversary $\mathcal{A}$ is able to output a valid forgery and $\mathcal{B}$ does not abort, then $\mathcal{B}$ is a successful adversary against the UF-CMA security of Σ. However, whenever $\mathcal{A}$ is successful (with non-negligible probability), by the O-SNARK definition we have that "Fail" occurs with negligible probability at most ε. Therefore, if $\mathcal{A}$ is successful with probability at least δ, then $\mathcal{B}$ is successful with non-negligible probability $\geq \delta - \varepsilon$.

□

Non-adaptive Security. Alternatively, one can modify the previous proof to show that the scheme has security against homomorphic signature adversaries that make non-adaptive signing queries, assuming the weaker assumption that Π is a *non-adaptive* O-SNARK (see Definition 5.3.2). In particular, combining this change with the result of Theorem 5.3.3 one obtains the following:

Theorem 6.2.4. *If Π is a SNARK, and Σ is a* UF-CMA*-secure signature scheme, then* $\mathsf{HomSig}[\Sigma, \Pi]$ *is secure against adversaries that make* non-adaptive *signing queries.*

Proof. The proof is very similar to that of Theorem 6.2.3, and thus we only sketch the main differences from that proof. To work with non-adaptive adversaries, the only main change is that for every non-adaptive adversary $\mathcal{A}$ one can define a corresponding non-adaptive O-SNARK adversary $\tilde{\mathcal{A}}$. Then the only difference is that the non-adaptive queries of $\mathcal{A}$ can be used to define the non-adaptive queries of $\tilde{\mathcal{A}}$. The rest of the proof proceeds analogously. □

Remark 6.2.5 (On the applicability of Corollary 5.4.7)**.** *We note that we cannot combine the positive result of Corollary 5.4.7 with Theorem 6.2.3 to conclude that the security of the homomorphic signature scheme holds under classical SNARKs. The inapplicability of Corollary 5.4.7 is due to its restriction for which adversaries have to query almost the entire message space. By looking at the* HomSig *construction (and the definition of homomorphic signatures too), one can note that an adversary who queries almost the entire message space of the underlying signature scheme can trivially break the security (for example he could obtain signatures on two distinct messages under the same label).*

6.2.3 Insecurity of $\mathsf{HomSig}[\Sigma^*, \Pi]$

Here we show the existence of a signature scheme Σ^* for which $\mathsf{HomSig}[\Sigma^*, \Pi]$ is insecure. Note that this insecurity result does not contradict our Theorem 6.2.3 as it is indeed possible to show that Σ^* is in the class of schemes for which the existence of an O-SNARK is ruled out. Rather, this counterexample shows that the issue with proving the security of the homomorphic signature construction is not a mere difficulty in doing the proof, but that the construction can actually become insecure, if one simply relies on an *arbitrary* signature scheme.

Construction of Σ^*. Consider the HomSig construction in which messages are bits (i.e., $\mu = 1$) and labels are strings of length $\ell = \mathsf{poly}(\lambda)$, such that the $\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ algorithm of Π is at most $(\ell + 1)$-bits long. Let $\widehat{\Sigma}$ be any $\mathsf{UF\text{-}CMA}$-secure scheme with message space $\mathcal{M} = \{0,1\}^{\ell+1}$. We construct the signature scheme Σ^* from $\widehat{\Sigma}$ as follows:

$\mathsf{kg}(1^\lambda)$**:** Run $(\widehat{\mathsf{vk}}, \widehat{\mathsf{sk}}) \xleftarrow{\$} \widehat{\Sigma}.\mathsf{kg}(1^\lambda)$, set $\mathsf{vk} = \widehat{\mathsf{vk}}$, $\mathsf{sk} = \widehat{\mathsf{sk}}$, and also set $\delta = p(\lambda) + \lambda$.

$\mathsf{sign}(\mathsf{sk}, m)$**:** Signing works as follows

- sample $r \xleftarrow{\$} \{0,1\}^\delta$, and compute $\widehat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r|m)$;
- sample $m' \xleftarrow{\$} \{0,1\}^{\ell+1}$, $r' \xleftarrow{\$} \{0,1\}^\delta$, compute $\widehat{\sigma}' \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r'|m')$, and set $\sigma' = (r', \widehat{\sigma}', 0, 0)$;
- parse $m' \in \{0,1\}^{\ell+1}$ bit by bit as $(m'_1|\cdots|m'_{\ell+1})$, set $\tau = m'_1|\cdots|m'_\ell$, $x = (\mathsf{vk}, m'_{\ell+1}, \tau)$, $w = (m'_{\ell+1}, \sigma')$, and let $\mathcal{I} : \{0,1\} \to \{0,1\}$ be the identity function;
- let $t = \#M_{\Sigma^*,\mathcal{I}}(x, w)$, and set $y = (M_{\Sigma^*,\mathcal{I}}, x, t)$;
- Interpret m as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y, w)$;
- if $|\pi| > p(\lambda)$ set $\pi' = 0$ and $y = 0$, else $\pi' = \pi$.
- output $\sigma = (r, \widehat{\sigma}, m', \pi')$.

$\mathsf{vfy}(\mathsf{vk}, m, \sigma)$**:** Parse $\sigma = (r, \widehat{\sigma}, m', \pi')$ where r is δ-bits long, and return the output of $\widehat{\Sigma}.\mathsf{vfy}(\widehat{\mathsf{vk}}, r|m, \widehat{\sigma})$.

Before proving the security of Σ^*, we provide some intuitions about the rational of the above construction:

- The signing algorithm consists of two main steps. First, given the message m, we sign $r|m$ (i.e., we prepend the random string r to m) using $\widehat{\Sigma}.\mathsf{sign}$. Second, we generate another signature on a randomly chosen message $m' \in \{0,1\}^{\ell+1}$. This is done following the same process as for m, i.e., by sampling a random $r' \xleftarrow{\$} \{0,1\}^\delta$ and signing $r'|m'$ using $\widehat{\Sigma}.\mathsf{sign}$.

- Then, we construct a theorem (y, w) for the relation $\mathcal{R}_{\Sigma^*}$ with respect to the identity function $\mathcal{I}$. In particular, recall the definition of $M_{\Sigma^*,\mathcal{I}}(x, w)$ from Section 6.2.2: this is the machine that on inputs $x = (\mathsf{vk}, m, \tau)$, where vk is a public key of the scheme Σ^*, m is a bit and $\tau \in \{0,1\}^\ell$ and $w = (m^*, \sigma^*)$ accepts iff

$$m = \mathcal{I}(m^*) \ \wedge \ \mathsf{vfy}(\mathsf{vk}, \tau|m, \sigma^*) = 1$$

- Finally, one interprets the input message m as a program $\mathcal{P}$ description and runs $\pi \leftarrow \mathcal{P}(y, w)$. Its output, together with m', is included in the signature as an extra information. As one can see, if $\mathcal{P}(\cdot,\cdot) = \mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$ then π is a valid proof. However, before returning π we make sure that (regardless of its validity) π is short enough and we set the new value π'.

- The meaning of δ: The reason for adding the random string r to the signature is to increase the entropy of the signatures generated by Σ^*. This is crucial for the signatures σ' that are used to generate π. We basically want to make sure that π cannot leak enough information about σ'. Since the output of the program is of length p, taking r' to be sufficiently long – of $p(\lambda) + \lambda$ bits – guarantees that some information about σ' is inevitably lost.

Proof of UF-CMA-Security for Σ^*. We prove that Σ^* is secure as long as $\widehat{\Sigma}$ is secure:

Lemma 6.2.6. *If $\widehat{\Sigma}$ is UF-CMA-secure scheme, then Σ^* is UF-CMA-secure. More precisely, for any PPT adversary $\mathcal{A}$ that has advantage $\varepsilon(\lambda)$ in breaking the security of Σ^* by making Q signing queries, there is a PPT adversary $\widehat{\mathcal{A}}$ that breaks the security of $\widehat{\Sigma}$ with advantage $> \varepsilon(\lambda) - Q/2^\lambda$ by making $2Q$ queries.*

Below we mention the steps for the proof of the lemma. The idea of the proof is that $\widehat{\mathcal{A}}$ runs $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)}(\widehat{\mathsf{vk}})$ and simulates every signing query m by executing $\mathsf{sign}(\mathsf{sk}, m)$ except that the executions of $\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, \cdot)$ are replaced by queries to $\widehat{\mathcal{A}}$'s signing oracle (this is why every signing query produces two queries on $\widehat{\mathcal{A}}$'s side). The only tricky part of the proof is to show that the message m^* used in the forgery leads to a *new* message $\widehat{m}^* = r^*|m^*$ in the game played by $\widehat{\mathcal{A}}$, i.e., that $r^*|m^* \neq r_i|m_i$ for all the $r_i|m_i$ queried by $\widehat{\mathcal{A}}$. We argue that this is the case with overwhelming probability, based on the random choices of all values r' in the signing query simulation. In fact, one should note that $\mathcal{A}$ never gets to see the value r' used to generate σ'; moreover, since r' is δ-bits long and $\mathcal{A}_p$ sees at most $p(\lambda)$ bits of information of it, then λ bits of r' are always lost. Therefore the probability that $r^* = r'$ is bounded by the probability that $\mathcal{A}$ predicts correctly r'.

Proof. Assume by contradiction that there exists an adversary $\mathcal{A}$ that has non-negligible advantage $\varepsilon(\lambda)$ against the UF-CMA security of Σ_p while running in polynomial time and making Q queries to the $\mathsf{sign}(\mathsf{sk}, \cdot)$ oracle. Starting from this adversary $\mathcal{A}$, we construct a PPT adversary $\widehat{\mathcal{A}}$ that is able to break the UF-CMA security of $\widehat{\Sigma}$ with non-negligible advantage and by making at most $2Q$ queries to its signing oracle $\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, \cdot)$.

We define $\widehat{\mathcal{A}}$ (which gets the public key $\widehat{\mathsf{vk}}$ and makes queries to $\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, \cdot)$ oracle) as follows:

$\widehat{\mathcal{A}}^{\widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)}(\widehat{\mathsf{vk}})$

Run $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{sign}(\widehat{\mathsf{sk}},\cdot)}(\widehat{\mathsf{vk}})$ and simulate queries m to $\mathsf{sign}(\widehat{\mathsf{sk}}, \cdot)$ as follows:
sample $r \xleftarrow{\$} \{0,1\}^\delta$, and query $\widehat{\sigma} \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r|m)$;
sample $m' \xleftarrow{\$} \{0,1\}^{\ell+1}, r' \xleftarrow{\$} \{0,1\}^\delta$;
query $\widehat{\sigma}' \leftarrow \widehat{\Sigma}.\mathsf{sign}(\widehat{\mathsf{sk}}, r'|m')$, and set $\sigma' = (r', \widehat{\sigma}', 0, 0)$;
parse $m' \in \{0,1\}^{\ell+1}$ as $(m'_1|\cdots|m'_{\ell+1})$;
set $\tau = m'_1|\cdots|m'_\ell$, $x = (\mathsf{vk}, m'_{\ell+1}, \tau)$, $w = (m'_{\ell+1}, \sigma')$;
let $t = \#M_{\Sigma^*,\mathcal{I}}(x, w)$, and set $y = (M_{\Sigma^*,\mathcal{I}}, x, t)$, where I is the identity;

interpret m as the description of program $\mathcal{P}(\cdot,\cdot)$ and thus run $\pi \leftarrow \mathcal{P}(y,w)$;
if $|\pi| > p(\lambda)$ set $\pi' = 0$ and $m' = 0$, else $\pi' = \pi$;
output $\sigma = (r, \widehat{\sigma}, m', \pi')$.
Parse $\sigma^* = (r^*, \widehat{\sigma}^*, \cdot, \cdot)$ and return $(r^*|m^*, \widehat{\sigma}^*)$

Let us now show that whenever $\mathcal{A}$ succeeds in the simulation described above, $\widehat{\mathcal{A}}$ succeeds in breaking the UF-CMA security of the scheme $\widehat{\Sigma}$, with all but negligible probability. To this end we have first to show that the simulation provided by $\widehat{\mathcal{A}}$ works correctly, and then show that $\widehat{\mathcal{A}}$ outputs a valid forgery as long as $\mathcal{A}_p$ outputs a forgery.

To ease the analysis, consider the set of all queries (and corresponding responses) made by $\mathcal{A}$:

$$\mathcal{Q}^* = \{m_i, \sigma_i = (r_i, \widehat{\sigma}_i, m'_i, \pi_i) \mid i = 1 \dots Q\}$$

Then the set of $\widehat{\mathcal{A}}$'s queries is $\widehat{\mathcal{Q}} = \widehat{\mathcal{Q}}^* \cup \widehat{\mathcal{Q}}'$ with

$$\widehat{\mathcal{Q}}^* = \{(\widehat{m}_i = r_i|m_i,\ \widehat{\sigma}_i) \mid i = 1 \dots Q\}$$

$$\widehat{\mathcal{Q}}' = \{(\widehat{m}'_j = r'_j|m'_j,\ \widehat{\sigma}'_j) \mid j = 1 \dots Q\}$$

Precisely, the first set $\widehat{\mathcal{Q}}^*$ consists of all signing queries asked by $\widehat{\mathcal{A}}$ to its oracle for signing the messages m_i queried by $\mathcal{A}$. The second set $\widehat{\mathcal{Q}}'$ instead, comprises the extra queries asked by $\widehat{\mathcal{A}}$ in the simulation for signing the sampled messages m'_j.

It is easy to see that $\widehat{\mathcal{A}}$ provides a perfect simulation to $\mathcal{A}$ as $\widehat{\mathcal{A}}$ can correctly answer every query of $\mathcal{A}$ using its own signing oracle.

So the main fact to show is that the message m^* used in the forgery leads to a *new* message $\widehat{m}^*$ in the game played by $\widehat{\mathcal{A}}$.

Let $(m^*, \sigma^* = (r^*, \widehat{\sigma}^*, \cdot, \cdot))$ be a valid forgery for Σ^* (i.e., $m^* \neq m_i$, for all $m_i \in \mathcal{Q}^*$), and let us consider the following undesired cases:

1. $(\widehat{m}^* = r^*|m^*, \cdot) \in \widehat{\mathcal{Q}}^*$: Since $m^* \neq m_i\ \forall m_i \in \mathcal{Q}^*$ this case cannot occur even if the corresponding strings r_i and r^* match.

2. $(\widehat{m}^* = r^*|m^*, \cdot) \in \widehat{\mathcal{Q}}'$: It must be that $\widehat{m}^* = r^*|m^* = r'_j|m'_j = \widehat{m}'_j$ for some $j \in \{1, \dots, Q\}$. In what follows we bound the probability that such equality happens and show that it is negligible.

Both r^* and r'_j are parsed as strings of the *same* length $\delta = p(\lambda) + \lambda$. Hence, $r^*|m^* = r'_j|m'_j$ immediately implies $m^* = m'_j$, which may be possible since m^* is of adversarial choice. To bound the probability of match we thus only look at the event that $r^* = r'_j$.

Now, the crucial observation is that the adversary $\mathcal{A}$ never sees the strings r'_j explicitly, and thus the probability of the match can be upper bounded by the probability that the adversary $\mathcal{A}$ guesses correctly the string $r'_j \in \{0,1\}^\delta$ where $\delta = p(\lambda) + \lambda$.

Below we argue that this happens with negligible probability $\leq \frac{Q}{2^\lambda}$. For $j \in \{1, \dots Q\}$ let Bad_j be the event that $r^* = r'_j$ for $(\widehat{m}'_j, \widehat{\sigma}'_j) \in \widehat{\mathcal{Q}}'$ and let $\mathsf{Bad} = \bigvee_{j=1}^{Q} \mathsf{Bad}_j$. Using the union bound we have:

$$\Pr[\mathsf{Bad}] = \Pr\left[\bigvee_{j=1}^{Q} \mathsf{Bad}_j\right] \leq \sum_{j=1}^{Q} \Pr\left[\mathsf{Bad}_j\right]$$

Now, we will bound the probability of Bad_j for *any* fixed j. The value $\pi_j \leftarrow \mathcal{P}(y,w)$ the adversary $\mathcal{A}$ gets from the j-th query is the only one that can reveal some information about

$\sigma'_j = (r'_j, \widehat{\sigma}'_j, 0, 0)$ and implicitly about r'_j. We show that the information gained from π_j does not give any advantage to the adversary.
Let us represent the process of running $\pi \leftarrow \mathcal{P}(y, (m'_\lambda, \sigma'_j = (R, \widehat{\sigma}'_j, 0, 0)))$ and returning π only if $|\pi| \leq p(\lambda)$ as a function $f(R)$ such that $f : \{0,1\}^\delta \to \{0,1\}^p$. Namely, we fix $\mathcal{P}$ and all its inputs but R. Observe that, for randomly chosen inputs, any such f is essentially performing a lossy data compression of $\delta - p(\lambda) = \lambda$ bits.
We have that $\forall f : \{0,1\}^\delta \to \{0,1\}^p$, the probability that any algorithm $\tilde{\mathcal{A}}$ guesses the random string r'_j on input $f(r'_j)$ is less than $\frac{2^p}{2^\delta} = \frac{1}{2^\lambda}$. The same holds for $\mathcal{A}$ and f defined as before. Hence, summing up these probabilities for all j and observing that $Q = \mathsf{poly}(\lambda)$, we obtain:

$$\Pr[\mathsf{Bad}] \leq \sum_{j=1}^{Q} \Pr\left[Bad_j\right] \leq \frac{Q}{2^\lambda}$$

that is negligible.

The proof is concluded by observing that

$$\mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\widehat{\mathcal{A}},\widehat{\Sigma}}(\lambda) \geq \varepsilon(\lambda)(1 - \Pr[\mathsf{Bad}]) \geq \varepsilon(\lambda) - \frac{Q}{2^\lambda}$$

□ □

Proof of Insecurity of $\mathsf{HomSig}[\Sigma^*, \Pi]$. Below we show an adversary $\mathcal{A}^*$ that has non-negligible probability of winning in $\mathsf{Exp}^{\mathsf{HomSig\text{-}UF}}_{\mathcal{A}^*,\mathsf{HomSig}[\Sigma^*,\Pi]}(\lambda)$. Our adversary wins by producing a Type 1 forgery with respect to an identity function. For ease of exposition, we give the following security experiment which is a specialization of $\mathsf{Exp}^{\mathsf{HomSig\text{-}UF}}$ to adversaries that only output Type 1 forgeries for identity functions:

Experiment $\mathsf{Exp}^{\mathsf{Type\text{-}1},Id}_{\mathcal{A},\mathsf{HomSig}[\Sigma,\Pi]}(\lambda)$
$(\mathsf{SK}, \mathsf{VK}) \xleftarrow{\$} \mathsf{HomKG}(1^\lambda)$
$(\mathcal{I}^* = (F_{id}, \tau^*), m^*, \bar{\sigma}^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK})$
If $\mathsf{HomVer}(\mathsf{VK}, \mathcal{I}^*, m^*, \bar{\sigma}^*) = 1$ and τ^* is "new", output 1
Else output 0

Lemma 6.2.7. *Let Π be an O-SNARK for $\mathbb{O}_{\Sigma^*}$ and Σ^* be the $\mathsf{UF\text{-}CMA}$-secure signature scheme defined above. Then there exists an efficient adversary $\mathcal{A}^*$ such that*

$$\Pr[\mathsf{Exp}^{\mathsf{Type\text{-}1},Id}_{\mathcal{A}^*,\mathsf{HomSig}[\Sigma^*,\Pi]}(\lambda) = 1] = 1 - \mathsf{negl}(\lambda)$$

Proof. Below is the description of our adversary $\mathcal{A}^*$:

Adversary $\mathcal{A}^{*\mathsf{HomSign}(\mathsf{SK},\cdot)}(\mathsf{VK})$
1 Query the signing oracle on $(\tau, m) := \mathsf{P}$
where P is the description of $\mathsf{Prove}(\mathsf{crs}, \cdot, \cdot)$
2 Parse the answer $\sigma = (r, \widehat{\sigma}, m', \pi')$, and $m' = (m'_1 | \cdots | m'_{\ell+1})$
3 Set $\tau^* = (m'_1 | \cdots | m'_\ell)$ and $m^* = m'_{\ell+1}$
4 Return $(\mathcal{I}^* = (F_{id}, \tau^*), m^*, \bar{\sigma}^* = (proof, \pi'))$

Note that, except with probability $2^{-\ell}$, it holds $\tau^* \neq \tau$. Moreover, by the correctness of Π (and its succinctness), the answer to $\mathcal{A}^*$'s oracle query contains a valid proof π' that verifies for the identity function with message m^* and label τ^*. In other words the output of $\mathcal{A}^*$ constitutes a Type-1 forgery with probability $1 - 2^{-\ell}$. □ □

Context-Hiding of HomSig.

Theorem 6.2.8. *If Π is a zero-knowledge O-SNARK then HomSig is weakly context-hiding.*

Proof. To show weakly context-hiding for $\mathsf{HomSig}[\Sigma, \Pi]$ we define a simulator $S^{\mathsf{Hide}} = (S^{\mathsf{KG}}, S^{\mathsf{Eval}})$. For this purpose we use the PPT simulator $S^{\Pi} = (S^{\mathsf{crs}}, S^{\mathsf{Prove}})$ from the zero-knowledge of Π.

$S^{\mathsf{KG}}(1^\lambda)$:
  Run $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{kg}(1^\lambda)$
  Run $(\mathsf{prs}, \mathsf{vst}, \mathsf{tr}) \leftarrow S^{\mathsf{crs}}(1^\lambda, T)$
  Set $\mathsf{SK} = (\mathsf{sk}, \mathsf{prs})$ and $\mathsf{VK} = (\mathsf{vk}, \mathsf{vst})$
  Output $(\mathsf{VK}, \mathsf{SK})$

$S^{\mathsf{Eval}}(\mathcal{P} = (F, \tau_1, \ldots \tau_n)), m^*)$:
  Set $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^*, \tau_1 \ldots \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$
  Run $\pi \leftarrow S^{\mathsf{Prove}}(\mathsf{prs}, \mathsf{tr}, y)$
  Output $\bar{\sigma} = (proof, \pi)$

For any distinguisher $\mathcal{D}^{\mathsf{Hide}}$ against the weakly context-hiding of $\mathsf{HomSign}[\Sigma, \Pi]$, we can easily construct a distinguisher $\mathcal{D}^{\Pi}$ against zero knowledge O-SNARK property:

$\mathcal{D}^{\Pi}(\mathsf{crs})$:
  Generate a pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{kg}(1^\lambda)$
  Set $\mathsf{SK} = (\mathsf{sk}, \mathsf{prs})$ and $\mathsf{VK} = (\mathsf{vk}, \mathsf{vst})$
  Let $F, m_1, \ldots, m_n, \tau_1 \ldots \tau_n$ be the fixed tuple:
    Run $\sigma_i \leftarrow \mathsf{HomSign}(\mathsf{sk}, \tau_i, m_i)\ \forall i \in [n]$
    Output (y, w) to its challenger, and get back π
   (where $y = (M_{\Sigma,F}, x = (\mathsf{vk}, m^* = F(m_1, \ldots, m_n), \tau_1 \ldots \tau_n), |x|^{e_{\Sigma,\mathcal{F}}})$
    and $w = (m_1, \sigma_1, \ldots, m_n, \sigma_n)$)
    run $b \leftarrow \mathcal{D}^{\mathsf{Hide}}(\mathsf{VK}, \mathsf{SK}, \sigma_1, \ldots, \sigma_n, \tilde{\sigma} = (proof, \pi))$
  Output b.

Clearly, if $\mathcal{D}^{\Pi}$ receives π and crs generated using the real algorithms, it simulates the real game to $\mathcal{D}^{\mathsf{Hide}}$. Otherwise, if these values are generated through the zero-knowledge simulator, then the view of $\mathcal{D}^{\mathsf{Hide}}$ is identical to the case where it receives values generated using our context hiding simulator described above. Therefore, the distinguishing advantage of $\mathcal{D}^{\Pi}$ in distinguishing between a real or a simulated proof is the same as that of the algorithm $\mathcal{D}^{\mathsf{Hide}}$ to distinguish the two distributions in the answers from the context-hiding definition. □

6.3 Succinct Functional Signatures

As a second application of O-SNARKs, we revisit the construction of succinct functional signatures of Boyle, Goldwasser, and Ivan [BGI14]. In [BGI14] this construction is proven

secure using a notion of SNARKs which significantly differs from the standard one [BCC+14]. To the best of our knowledge, there are no known instantiations of SNARKs under this definition, in the standard model (and is not clear whether it is possible to find some). On the other hand, if one wants to prove the security of this construction using the classical SNARK definition, the security proof incurs the same subtleties related to running an extractor in the presence of a signing oracle.

In this section, we revisit the construction of [BGI14], and we prove its security using O-SNARKs. Interestingly, this proof differs a little from the one of homomorphic signature as here we have to consider O-SNARKs for *multiple* signing oracles. Similarly to the homomorphic signature case, we can also show that the scheme is secure against adversaries that declare all their queries in advance by assuming classical SNARKs.

6.3.1 Definition of Functional Signatures

Informally speaking, functional signatures [BGI14] are digital signature schemes where, starting from a master signing key (which can be used to sign any message), one can create a specific key related to some function f that enables one to sign only outputs of f, i.e., messages $f(m)$. For security, functional signatures must be unforgeable in the sense that any PPT adversary who can ask for keys on functions $f_1, \ldots, f_\ell$ and signatures on messages $m_1, \ldots, m_Q$, can only output signatures on messages m that are in the range of either one of $f_1, \ldots, f_\ell$ (or are equal to one of the queried m_i).

We recall the formal definition of succinct functional signatures as provided in [BGI14].

Definition 6.3.1 (Functional Signatures [BGI14])**.** *A functional signature scheme* FS *for a message space* $\mathcal{M}$ *and function family* $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ *is a tuple of probabilistic, polynomial-time algorithms* $(\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$ *that work as follows*

$\mathsf{FS.Setup}(1^\lambda)$ takes a security parameter λ and outputs a master verification key mvk and a master secret key msk.

$\mathsf{FS.KeyGen}(\mathsf{msk}, f)$ takes the master secret key msk and a function $f \in \mathcal{F}$ (represented as a circuit) and it outputs a signing key sk_f for f.

$\mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m)$ takes as input a function $f \in \mathcal{F}$, a signing key sk_f, and a message $m \in \mathcal{D}_f$, and it outputs $(f(m), \sigma)$ where σ represents a signature on $f(m)$.

$\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma)$ takes as input the master verification key mvk, a message $m^* \in \mathcal{M}$ and a signature σ, and outputs either 1 (accept) or 0 (reject).

and satisfy *correctness*, *unforgeability*, and *function privacy* as described below.

- **Correctness.** A functional signature scheme is correct if the following holds with probability 1:

$$\forall f \in \mathcal{F},\ \forall m \in \mathcal{D}_f,\ (\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda),\ \mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f),$$
$$(m^*, \sigma) \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m), \mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma) = 1$$

- **Unforgeablity.** A functional signature scheme is unforgeable if for every PPT adversary $\mathcal{A}$ there is a negligible function ε such that $\Pr[\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A},\mathsf{FS}}(\lambda) = 1] \leq \varepsilon(\lambda)$ where the experiment $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A},\mathsf{FS}}(\lambda)$ is described in the following:

Key generation: Generate $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$, and gives mvk to $\mathcal{A}$.

Queries: The adversary is allowed to adaptively query a key generation oracle $\mathcal{O}_{\mathsf{key}}$ and a signing oracle $\mathcal{O}_{\mathsf{sign}}$, that share a dictionary D indexed by tuples $(f, i) \in \mathcal{F} \times \mathbb{N}$, whose entries are signing keys. For answering these queries, the challenger proceeds as follows:

- $\mathcal{O}_{\mathsf{key}}$ (f, i):
  - If $(f, i) \in D$ (i.e., the adversary had already queried the tuple (f, i)), then the challenger replies with the same key sk_f^i generated before.
  - Otherwise, generate a new $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add the entry $(f, i) \rightarrow \mathsf{sk}_f^i$ in D, and return sk_f^i.
- $\mathcal{O}_{\mathsf{sign}}$ (f, i, m):
  - If there is an entry for the key (f, i) in D, then the challenger generates a signature on $f(m)$ using this key, i.e., $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f^i, m)$.
  - Otherwise, generate a new key $\mathsf{sk}_f^i \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, add an entry $(f, i) \rightarrow \mathsf{sk}_f^i$ to D, and generate a signature on $f(m)$ using this key, i.e., $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f^i, m)$.

Forgery: After the adversary is done with its queries, it outputs a pair (m^*, σ), and the experiment outputs 1 iff the following conditions hold

- $\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \sigma) = 1$.
- there does not exist m such that $m^* = f(m)$ for any f which was sent as a query to the $\mathcal{O}_{\mathsf{key}}$ oracle.
- there does not exist a pair (f, m) such that (f, m) was a query to the $\mathcal{O}_{\mathsf{sign}}$ oracle and $m^* = f(m)$.

Function Privacy Definition

Intuitively, function privacy requires that the distribution of signatures on a message m that are generated via different keys sk_f should be computationally indistinguishable, even given the secret keys and master signing key. More formally, a functional signature scheme has function privacy if for every PPT adversary $\mathcal{A}$ there is a negligible function ν such that $\Pr[\mathsf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A},\mathsf{FS}}(\lambda) = 1] \leq \nu(\lambda)$ where experiment $\mathsf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A},\mathsf{FS}}(\lambda)$ works as follows:

- The challenger generates a key pair $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$ and gives $(\mathsf{mvk}, \mathsf{msk})$ to $\mathcal{A}$.
- The adversary chooses a function f_0 and receives an (honestly generated) secret key $\mathsf{sk}_{f_0} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_0)$.
- The adversary chooses a second function f_1 such that $|f_0| = |f_1|$ (where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key $\mathsf{sk}_{f_1} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_1)$.
- The adversary chooses a pair of values (m_0, m_1) such that $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.

- The challenger selects a random bit $b \leftarrow \{0,1\}$ and computes a signature on the image message $m^* = f_0(m_0) = f_1(m_1)$ using secret key sk_{f_b}, and gives the resulting signature $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{f_b}, m_b)$ to $\mathcal{A}$.
- The adversary outputs a bit b', and the experiment outputs 1 iff $b' = b$.

Definition 6.3.2 (Succinct Functional Signatures)**.** *A functional signature scheme is called* succinct *if there exists a polynomial $s(\cdot)$ such that, for every security parameter $\lambda \in \mathbb{N}$, $f \in \mathcal{F}$, $m \in \mathcal{D}_f$, it holds with probability 1 over $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{FS.Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f)$, $(f(m), \sigma) \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_f, m)$ that $|\sigma| \leq s(\lambda, |f(m)|)$. In particular, the size of σ is independent of the function's size, $|f|$, and the function's input size, $|m|$.*

6.3.2 Succinct Functional Signatures from O-SNARKs

In the following we show a construction for message space $\mathcal{M}$ and family of functions $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ whose running time is bounded by some fixed polynomial $t_{\mathcal{F}}(|m|)$. To build the scheme, we use two $\mathsf{UF\text{-}CMA}$-secure signature schemes, $\Sigma_0 = (\mathsf{kg}_0, \mathsf{sign}_0, \mathsf{vfy}_0)$ for message space $\mathcal{M}_0$ and $\Sigma' = (\mathsf{kg}', \mathsf{sign}', \mathsf{vfy}')$ for message space $\mathcal{D}$, together with a fully succinct zero-knowledge O-SNARK $\Pi = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Ver})$ for the NP language L defined below. While in [BGI14] a single signature scheme is used, we prefer to use two different ones as this allows for a more precise statement since we will need to apply different restrictions to $\mathcal{M}_0$ and $\mathcal{D}$ to obtain a precise proof.

We obtain an unforgeable functional signature scheme satisfying succinctness and function privacy $\mathsf{FS} = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$.

Defining the Relation $\mathcal{R}_L$. Let M_L be a random-access machine as defined below, and $t_L(k) = k^{e_L}$ be a polynomial. $\mathcal{R}_L$ is the binary relation consisting of all pairs (y, w) such that, parsing $y = (M_L, x, t)$, $M_L(x, w)$ accepts in at most t steps and $t \leq t_L(|x|)$. The values x are of the form $x = (m^*, \mathsf{mvk}_0)$ where mvk_0 is a public key of the scheme Σ_0, and $m^* \in \mathcal{M}$ is a message. The values w are instead tuples $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ such that $m \in \mathcal{D}_f$ with $\mathcal{D}_f \subset \mathcal{D}$, and $\sigma_{\mathsf{vk}'}$, σ_m are signatures for the schemes Σ_0 and Σ' respectively. On input such a pair (x, w), $M_L(x, w)$ is the random-access machine that accepts iff the following conditions (1)&(2)&(3) hold:

$$
\begin{aligned}
&(1) \quad m^* = f(m)\\
&(2) \quad \mathsf{vfy}'(\mathsf{vk}', m, \sigma_m) = 1\\
&(3) \quad \mathsf{vfy}_0(\mathsf{mvk}_0, f|\mathsf{vk}', \sigma_{\mathsf{vk}'}) = 1
\end{aligned}
$$

Given polynomial bounds on the running times of verification algorithms vfy' and vfy_0, and a (fixed) bound $t_{\mathcal{F}}(\cdot)$ on the size and running time of every $f \in \mathcal{F}$, one can deduce a polynomial time bound $t_L(|x|) = |x|^{e_L}$ for the machine M_L.

6.3.3 Construction of FS

Using the signature schemes Σ_0, Σ' and a fully-succinct zero-knowledge O-SNARK Π for NP, we construct the functional signature scheme $\mathsf{FS}[\Sigma_0, \Sigma', \Pi] = (\mathsf{FS.Setup}, \mathsf{FS.KeyGen}, \mathsf{FS.Sign}, \mathsf{FS.Ver})$ as follows:

$\mathsf{FS.Setup}(1^\lambda)$:

Generate a pair of keys for Σ_0: $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$.

Generate a crs for Π: $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.
Set the master secret key $\mathsf{msk} = \mathsf{msk}_0$, and the master verification key $\mathsf{mvk} = (\mathsf{mvk}_0, \mathsf{crs})$.

$\mathsf{FS.KeyGen}(\mathsf{msk}, f)$:
Generate a new key pair $(\mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{kg}'(1^\lambda)$ for the scheme Σ'.
Compute $\sigma_{\mathsf{vk}'} \leftarrow \mathsf{sign}_0(\mathsf{msk}_0, f|\mathsf{vk}')$, and let the certificate c be $c = (f, \mathsf{vk}', \sigma_{\mathsf{vk}'})$.
Output $\mathsf{sk}_f = (\mathsf{sk}', c)$.

$\mathsf{FS.Sign}(\mathsf{mvk}, f, \mathsf{sk}_f, m)$:
Parse sk_f as $(\mathsf{sk}', c = (f, \mathsf{vk}', \sigma_{\mathsf{vk}'}))$.
Sign m using sk' in Σ': $\sigma_m \leftarrow \mathsf{sign}'(\mathsf{sk}', m)$.
Set $y = (M_L, x, t)$ with $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$), and $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$.
Run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, y, w)$ and output $(m^* = f(m), \pi)$.
Informally, π is a zero-knowledge proof that $(f(m), \mathsf{mvk}_0) \in L$, a proof that the signer knows a key for f (constructed using Σ_0) and a valid signature of m in the underlying signature scheme Σ'.

$\mathsf{FS.Ver}(\mathsf{mvk}, m^*, \pi)$:
Parse $\mathsf{mvk} = (\mathsf{mvk}_0, \mathsf{crs})$ and set $y = (M_L, x, t)$ where $x = (\mathsf{mvk}_0, m^*)$ and $t = |x|^{e_L}$.
Output the same bit returned by $\mathsf{Ver}_{e_L}(\mathsf{crs}, y, \pi)$.

Correctness. It is not hard to see that as long as Σ_0, Σ' and Π are correct, then FS is also correct.

Succinctness. This property immediately follows from the succinctness of Π.

Unforgeability. We prove the security of FS under the unforgeability of schemes Σ_0 and Σ' and using the notion of O-SNARKs for a specific family of oracles $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ that we define below.

$\mathbb{O}_{\mathsf{m}\Sigma,Q}$ is parametrized by the algorithms of the signature schemes Σ_0, Σ' and by a polynomial $Q = Q(\lambda)$. Every member $\mathcal{O}$ of $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ is described by a set of secret keys $\mathsf{msk}_0, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_Q$ (i.e., the process of sampling $\mathcal{O} \leftarrow \mathbb{O}$ consists of running $(\mathsf{mvk}_0, \mathsf{msk}_0) \xleftarrow{\$} \mathsf{kg}_0(1^\lambda)$ and $(\mathsf{vk}'_i, \mathsf{sk}'_i) \xleftarrow{\$} \mathsf{kg}'_1(1^\lambda), \forall i \in [Q]$). The oracle $\mathcal{O}$ works as follows:

$$\mathcal{O}(i, \text{`}vk') = \begin{cases} \mathsf{mvk}_0 & \text{If } i = 0, \\ \mathsf{vk}'_i & \text{otherwise.} \end{cases} \qquad \mathcal{O}(i, \text{`}sk') = \begin{cases} \bot & \text{If } i = 0, \\ \mathsf{sk}'_i & \text{otherwise.} \end{cases}$$

$$\mathcal{O}(i, m) = \begin{cases} (\mathsf{Cnt}, \mathsf{sign}_0(\mathsf{msk}_0, m|\mathsf{vk}'_{\mathsf{Cnt}})), \mathsf{Cnt} \leftarrow \mathsf{Cnt} + 1 & \text{If } i = 0 \text{ and } \mathsf{Cnt} \leq Q, \\ \bot & \text{If } i = 0 \text{ and } \mathsf{Cnt} > Q, \\ \mathsf{sign}'(\mathsf{sk}'_i, m) & \text{otherwise.} \end{cases}$$

For the sake of simplicity, we compactly denote $\mathcal{O}_0(\cdot) = \mathcal{O}(0, \cdot)$ and $\mathcal{O}'_i(\cdot) = \mathcal{O}(i, \cdot)$ for all $i > 0$. From the above description, note that oracle $\mathcal{O}_0$ is stateful and we assume it starts with $\mathsf{Cnt} = 1$.

Finally, we point out that for some technical reasons that we mention in Remark 6.3.10 at the end of this section, it is not possible to use the notion of O-SNARK for a *single* signing oracle to prove the security of the functional signature scheme. This is the reason why we explicitly considered O-SNARKs for this more complex family of multiple signing oracles.

Theorem 6.3.3. *If* Π *is an O-SNARK for* $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ *for every* $Q = \mathsf{poly}(\lambda)$*, and* Σ_0, Σ' *are* UF-CMA*-secure, then* $\mathsf{FS}[\Sigma_0, \Sigma', \Pi]$ *is an unforgeable functional signature.*

Proof. Our proof consists of the following steps:

1. We show that for every successful $\mathcal{A}_{\mathsf{FS}}$ against the unforgeability of FS there exists an O-SNARK adversary $\tilde{\mathcal{A}}$ for an oracle from $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ such that $\tilde{\mathcal{A}}$ outputs a valid proof with the same (non-negligible) probability of success of $\mathcal{A}_{\mathsf{FS}}$. By the adaptive knowledge soundness for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ we then obtain that for such $\tilde{\mathcal{A}}$ there exists a suitable extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that outputs a valid witness with all but negligible probability.
2. From the previous point, considering adversary $\tilde{\mathcal{A}}$ and the corresponding extractor, we can partition adversary-extractor pairs in two types: (1) those that yield a witness w containing a pair (f, vk') that was never signed before, and (2) those that yield w containing (f, vk') that was signed before. We show that adversaries of type (1) can be used to break the security of the signature scheme Σ_0, whereas adversaries of type (2) can be used to break the security of Σ'.

Existence of an Extractor for $\mathcal{A}_{\mathsf{FS}}$. Consider any adversary $\mathcal{A}_{\mathsf{FS}}$ that while running in $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}$ it outputs (m^*, π^*) and makes the experiment generate Q secret keys of the scheme Σ'. For every such $\mathcal{A}_{\mathsf{FS}}$, we show there exists another adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ that, on input crs, and given oracle $\mathcal{O} \leftarrow \mathbb{O}_{\mathsf{m}\Sigma,Q}$, outputs a pair (y, π^*). We describe $\tilde{\mathcal{A}}^{\mathcal{O}}$ below. During its execution it maintains a dictionary D similar to the one in the definition of $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}$, except that instead of storing mappings like $(f, i) \to \mathsf{sk}^i_f$, it maps a pair (f, i) to a triple (j, sk', c) where $j \in [Q]$. Intuitively, this means that a queried pair (f, i) is associated to oracle $\mathcal{O}'_j$.

$\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$:

Query $\mathsf{mvk}_0 \leftarrow \mathcal{O}_0(\text{`}vk')$ and run $\mathcal{A}^{\mathcal{O}_{\mathsf{key}},\mathcal{O}_{\mathsf{sign}}}_{\mathsf{FS}}(\mathsf{mvk} = (\mathsf{crs}, \mathsf{mvk}_0))$
Simulate queries (f, i) to $\mathcal{O}_{\mathsf{key}}$ as follows:
  if $[(f, i) \to (j, \mathsf{sk}', c)] \in D$: output $\mathsf{sk}^i_f = (\mathsf{sk}', c)$
  else if $[(f, i) \to (j, \cdot, c)] \in D$:
    ask $\mathsf{sk}'_j \leftarrow \mathcal{O}'_j(\text{`}sk')$ and output $\mathsf{sk}^i_f = (\mathsf{sk}'_j, c)$
  else:
    ask $(j, \sigma_{\mathsf{vk}'_j}) \leftarrow \mathcal{O}_0(f)$, $\mathsf{vk}'_j \leftarrow \mathcal{O}'_j(\text{`}vk')$, $\mathsf{sk}'_j \leftarrow \mathcal{O}'_j(\text{`}sk')$
    add $(f, i) \to (j, \mathsf{sk}'_j, c)$ to D with $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
    output $\mathsf{sk}^i_f = (\mathsf{sk}'_j, c)$
Simulate queries (f, i, m) to $\mathcal{O}_{\mathsf{sign}}$ as follows:
  if (f, i) not assigned in D:
   ask $(j, \sigma_{\mathsf{vk}'_j}) \leftarrow \mathcal{O}_0(f)$, $\mathsf{vk}'_j \leftarrow \mathcal{O}'_j(\text{`}vk')$
   add $(f, i) \to (j, \cdot, c)$ to D with $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
   ask $\sigma_m \leftarrow \mathcal{O}'_j(m)$
   set $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$, $w = (m, f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j}, \sigma_m)$
   run $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$
   output $(f(m), \pi)$
  if $[(f, i) \to (j, \cdot, c)] \in D$: parse $c = (f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j})$
   ask $\sigma_m \leftarrow \mathcal{O}'_j(m)$
   set $x = (\mathsf{mvk}_0, f(m))$, $t = |x|^{e_L}$, $w = (m, f, \mathsf{vk}'_j, \sigma_{\mathsf{vk}'_j}, \sigma_m)$

run $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$
output $(f(m), \pi)$
When $\mathcal{A}_{\mathsf{FS}}$ outputs (m^*, π^*)
set $y = (M_L, x = (\mathsf{mvk}_0, m^*), t = |x|^{e_L})$
output (y, π^*)

As one can see, given the definition of oracles from $\mathbb{O}_{\mathsf{m}\Sigma,Q}$, the simulation provided by $\tilde{\mathcal{A}}$ to $\mathcal{A}_{\mathsf{FS}}$ is perfect. So, whenever $\mathcal{A}_{\mathsf{FS}}$ outputs a valid forgery $\tilde{\mathcal{A}}$ outputs a pair (y, π^*) that verifies correctly. Moreover, defined in this way, the adversary $\tilde{\mathcal{A}}^{\mathcal{O}}$ fits the definition of adaptive knowledge soundness for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ by which we know that there exists an extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that, given the same input of $\tilde{\mathcal{A}}^{\mathcal{O}}$ and the transcript of oracle queries/answers made and received by $\tilde{\mathcal{A}}^{\mathcal{O}}$, outputs a witness w such that the probability that (y, π^*) verifies and $(y, w) \notin \mathcal{R}_L$ is negligible.

We define the following hybrid games that involve running $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$:

$\mathbf{G}_1$ is the same experiment as $\mathsf{O\text{-}KS}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q})$ except that its outcome is defined differently. $\mathbf{G}_1$ outputs 1 iff $\mathsf{Ver}(\mathsf{crs}, y, \pi) = 1$ and the value m^* inside y constitutes a forgery according to the oracle queries made by $\tilde{\mathcal{A}}$ during the game.

By the construction of $\tilde{\mathcal{A}}$ from $\mathcal{A}_{\mathsf{FS}}$ it holds

$$\Pr[\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda) = 1] \quad = \quad \Pr[\mathbf{G}_1 \Rightarrow 1] \tag{6.1}$$

$\mathbf{G}_2$ is the same as $\mathbf{G}_1$ except that in order to output 1 it additionally checks that $(y, w) \in \mathcal{R}_L$.

Essentially, the outcome of $\mathbf{G}_2$ differs from that of $\mathbf{G}_1$ only if in $\mathbf{G}_2$ $(y, w) \notin \mathcal{R}_L$. Hence,

$$\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1] \quad \leq \quad \Pr[\mathsf{O\text{-}KS}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q}) \Rightarrow 1] \tag{6.2}$$

Moreover, let us define the following two events in game $\mathbf{G}_2$.
Let $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ be the witness returned by $\mathcal{E}_{\tilde{\mathcal{A}}}$:

Ev_1 occurs if $\forall j \in [Q] : \mathsf{vk}' \neq \mathsf{vk}'_j$, or $\exists j \in [Q] : \mathsf{vk}' = \mathsf{vk}'_j$ but $\tilde{\mathcal{A}}$ *never* made a query $\mathcal{O}_0(f)$ that returned $(j, \cdot)$;

Ev_2 occurs if $\mathsf{vk}' = \mathsf{vk}'_j$ for some $j \in [Q]$ and $\tilde{\mathcal{A}}$ did make a query $(j, \sigma) \leftarrow \mathcal{O}_0(f)$.

Clearly, it holds

$$\Pr[\mathbf{G}_2 \Rightarrow 1] \quad = \quad \Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1] + \Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2] \tag{6.3}$$

In the remaining part of the proof, we show that both $\Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1]$ and $\Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]$ are negligible under the assumption that, respectively, Σ_0 and Σ' are unforgeable.

Claim 6.3.4. *For every efficient adversary $\mathcal{A}_{\mathsf{FS}}$, there is an efficient forger $\mathcal{F}_0$ such that* $\Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_1] = \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}_0,\Sigma_0}(\lambda)$.

Proof. Let $\mathcal{A}_{\mathsf{FS}}$ be an adversary that runs in $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda)$, and let $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ be the pair of algorithms built out of $\mathcal{A}_{\mathsf{FS}}$ as defined before. Below we show how to build an efficient forger $\mathcal{F}_0$ out of $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ so that its probability of forging against Σ_0 is at least $\Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathcal{E}_1]$. $\mathcal{F}_0$ gets the public key mvk_0 and has access to oracle $\mathcal{O}_{\Sigma_0} = \mathsf{sign}_0(\mathsf{msk}_0, \cdot)$.

$\mathcal{F}_0^{\mathcal{O}_{\Sigma_0}}(\mathsf{mvk}_0)$:
Initialize $\mathsf{qt} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $\mathsf{Cnt} \leftarrow 1$
Generate $(\mathsf{sk}'_i, \mathsf{vk}'_i) \xleftarrow{\$} \mathsf{kg}'(1^\lambda)\ \forall i \in [Q]$
Generate $\mathsf{crs} = (\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda)$ and run $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$
Simulate all queries to $\mathcal{O}'_i$ using $\mathsf{sk}'_i, \mathsf{vk}'_i$ and add all queries-answers to qt
Simulate queries $\mathcal{O}_0(m)$ as follows:
if $m =$ 'vk': output mvk_0
else:
ask $\sigma \leftarrow \mathcal{O}_{\Sigma_0}(m|\mathsf{vk}'_{\mathsf{Cnt}})$
add $(m|\mathsf{vk}'_j, \sigma)$ to qt, and add $m|\mathsf{vk}'_j$ to $\mathcal{T}$
increment $\mathsf{Cnt} \leftarrow \mathsf{Cnt} + 1$
output σ
Let (y, π^*) be $\tilde{\mathcal{A}}$'s output
Run $w \leftarrow \mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$
Check that $(y, w) \in \mathcal{R}_L$:
[Fail] Abort if this does not hold.
Else parse $w = (m, f, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma_m)$ and proceed:
[A] If $(f|\mathsf{vk}') \notin \mathcal{T}$ return $(f|\mathsf{vk}', \sigma_{\mathsf{vk}'})$.
[B] If $(f|\mathsf{vk}') \in \mathcal{T}$ abort.

Algorithm $\mathcal{F}_0$ can perfectly simulate $\mathbf{G}_2$ to $\tilde{\mathcal{A}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$. Furthermore, it is easy to see that if $\mathbf{G}_2$ outputs 1 and Ev_1 occurs, then $\mathcal{F}_0$'s simulation ends up exactly in case (A), that is $\mathcal{F}_0$ returns a signature $\sigma_{\mathsf{vk}'}$ on a new message $f|\mathsf{vk}'$. Since $(y, w) \in \mathcal{R}_L$ one has that $\sigma_{\mathsf{vk}'}$ is valid, and thus is a forgery.

Finally, it is worth noting that for this simulation the adversary $\tilde{\mathcal{A}}$ can even ask $\mathcal{O}'_j(\text{'}sk'\text{'})$ for *all* j oracles without affecting our reduction. □ □

Claim 6.3.5. *For every efficient adversary $\mathcal{A}_{\mathsf{FS}}$ there is an efficient forger $\mathcal{F}'$ such that* $\Pr[\boldsymbol{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2] \leq Q \cdot \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda)$.

Proof. Let $\mathcal{A}_{\mathsf{FS}}$ be an adversary that runs in $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda)$, and let $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ be the pair of algorithms built out of $\mathcal{A}_{\mathsf{FS}}$ as defined before. Below we show how to build an efficient forger $\mathcal{F}'$ out of $\tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}$ so that its probability of forging against Σ' is at least $\Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]/Q$. $\mathcal{F}'$ gets a public key vk' and has access to oracle $\mathcal{O}_{\Sigma'} = \mathsf{sign}'(\mathsf{sk}', \cdot)$.

$\mathcal{F}'^{\mathcal{O}'}(\mathsf{vk}')$:
Initialize $\mathsf{qt} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $\mathsf{Cnt} \leftarrow 0$
Generate $\mathsf{crs} = (\mathsf{prs}, \mathsf{vst}) \leftarrow \mathsf{Gen}(1^\lambda)$
Generate a pair $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$
Choose a random $q \xleftarrow{\$} \{1, \dots Q\}$
Generate $(\mathsf{sk}'_i, \mathsf{vk}'_i) \xleftarrow{\$} \mathsf{kg}'(1^\lambda)\ \forall i \in [Q] \setminus \{q\}$
Run $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$
Simulate all queries to $\mathcal{O}_0$ using $\mathsf{mvk}_0, \mathsf{msk}_0$:
add all queries-answers to qt and all signed messages $m|\mathsf{vk}'$ to $\mathcal{T}$
Simulate all queries to $\mathcal{O}'_i$ using $\mathsf{vk}'_i, \mathsf{sk}'_i$ for all $i \in [Q] \setminus \{q\}$
add all queries-answers to qt
Simulate queries $\mathcal{O}'_q(m)$ as follows:

if $m = `vk'$: output $\mathsf{vk}'$ and add $(`vk', \mathsf{vk}')$ to qt
else if $m = `sk'$: Abort
else: ask $\sigma \leftarrow \mathcal{O}_{\Sigma'}(m)$ and add (m, σ) to qt
output σ
Let (y, π^*) be $\tilde{\mathcal{A}}$'s output
Run $w \leftarrow \mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$
Check that $(y, w) \in \mathcal{R}_L$:
[Fail] Abort if this does not hold.
Else parse $w = (m, f, \mathsf{vk}^*, \sigma_{\mathsf{vk}^*}, \sigma_m)$ and proceed:
[A] If $(f|\mathsf{vk}^*) \notin \mathcal{T}$ Abort.
[B] If $(f|\mathsf{vk}^*) \in \mathcal{T}$ and $\mathsf{vk}^* \neq \mathsf{vk}'$ Abort.
[C] If $(f|\mathsf{vk}^*) \in \mathcal{T}$ and $\mathsf{vk}^* = \mathsf{vk}'$ return (m, σ_m).

As one can see, unless it aborts, algorithm $\mathcal{F}'$ can perfectly simulate $\mathbf{G}_2$ to $\tilde{\mathcal{A}}$ and $\mathcal{E}_{\tilde{\mathcal{A}}}$. Furthermore, it is easy to see that if $\mathbf{G}_2$ outputs 1, Ev_2 occurs, and there is no abort while answering queries, then the simulation of $\mathcal{F}'$ ends up in cases (B) or (C). However, since $(f|\mathsf{vk}^*) \in \mathcal{T}$, we have that $\mathsf{vk}^* = \mathsf{vk}'_j$ for some $j \in [Q]$ (where we let $\mathsf{vk}'_q = \mathsf{vk}'$). So, if there is no abort at all, we have that $\mathsf{vk}^* = \mathsf{vk}'$ and thus $\mathcal{F}'$ returns a valid signature σ on a message m (recall that validity follows from $(y, w) \in \mathcal{R}_L$). By definition of $\mathbf{G}_2$, we also have that if it outputs 1, then the message m^* in y constitutes a forgery according to the definition of $\mathsf{Exp}^{\mathsf{FS\text{-}UF}}$. In particular, it holds that for the given $f|\mathsf{vk}'$, there was no signing query $\mathcal{O}'_q(m)$ such that $m^* = f(m)$. Therefore, if m is such that $m^* = f(m)$ (again this follows from $(y, w) \in \mathcal{R}_L$), then m cannot have been queried to $\mathcal{O}'_q$, i.e., $\mathcal{F}'$ never queried m to its signing oracle. From this we have that, as long as $\mathbf{G}_2$ outputs 1, Ev_2 occurs and there is no abort, then $\mathcal{F}'$ outputs a valid forgery. To conclude the proof, we observe that $\mathcal{F}'$ does not abort with probability $1/Q$ which is the probability that the guess of q, for which $\mathsf{vk}^* = \mathsf{vk}'$, is correct. Therefore, we have that $\mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda) = \Pr[\mathbf{G}_2 \Rightarrow 1 \wedge \mathsf{Ev}_2]/Q$.

Finally, we note that the above proof works even if the adversary $\tilde{\mathcal{A}}$ queries $\mathcal{O}'_j(`sk')$ on all oracles but the q-th one. This observation will be useful when we discuss the existence of O-SNARKs for this oracle family.

□

Putting together the bounds in equations Equation (6.1), Equation (6.2) and Equation (6.3), with the results of Claims 6.3.4 and 6.3.5, eventually we obtain:

$$\Pr[\mathsf{Exp}^{\mathsf{FS\text{-}UF}}_{\mathcal{A}_{\mathsf{FS}},\mathsf{FS}}(\lambda) = 1] \quad \leq \quad \Pr[\mathsf{O\text{-}KS}(\lambda, \tilde{\mathcal{A}}, \mathcal{E}_{\tilde{\mathcal{A}}}, \mathbb{O}_{\mathsf{m}\Sigma,Q}) \Rightarrow 1] + \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}_0,\Sigma_0}(\lambda) + \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{F}',\Sigma'}(\lambda)$$

which shows that any efficient adversary has at most negligible probability of breaking the security of scheme FS under the assumption that Π is an O-SNARK for $\mathbb{O}_{\mathsf{m}\Sigma,Q}$ and the schemes Σ_0, Σ' are unforgeable. □

Non-adaptive Unforgeability. Similarly to the homomorphic signature case, it is possible to show that the functional signature scheme achieves security against (functional signature) adversaries that make *non-adaptive* signing queries (i.e., all queries are declared at the beginning of the game). This weaker security can be proven assuming that Π is a *non-adaptive* O-SNARK (see Definition 5.3.2). Combining this change with the result of Theorem 5.3.3 we obtain the following:

Theorem 6.3.6. *If* Π *is a SNARK and* Σ_0, Σ' *are* UF-CMA*-secure signature schemes, then* $\mathsf{FS}[\Sigma_0, \Sigma', \Pi]$ *is a functional signature where unforgeability holds against adversaries that make non-adaptive signing queries.*

Proof. The proof of the theorem can be obtained via straightforward modifications to the proof of Theorem 6.3.3. Having in mind the intuition provided earlier, the main idea is that to work with non-adaptive adversaries, one can define a non-adaptive O-SNARK adversary $\tilde{\mathcal{A}}$ for every non-adaptive functional signature adversary $\mathcal{A}$. In particular, the non-adaptive queries of $\mathcal{A}$ can be used to define the non-adaptive queries of $\tilde{\mathcal{A}}$. The rest of the proof proceeds analogously. □

6.3.4 Function Privacy of FS

We show that the functional signature construction satisfies function privacy provided that the O-SNARK is zero-knowledge.

Theorem 6.3.7. *If* Π *is a zero-knowledge O-SNARK then* FS *satisfies function privacy.*

Proof. We show that for every adversary $\mathcal{A}_{\mathsf{priv}}$ against the function privacy experiment $\mathsf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A}_{\mathsf{priv}},\mathsf{FS}}(\lambda)$, we can construct a distinguisher algorithm $\mathcal{D}$ against the zero knowledge property of Π.

Consider the following two hybrid experiments:

$\mathbf{G}_0$ is the same as $\mathsf{Exp}^{\mathsf{FS\text{-}FPri}}_{\mathcal{A}_{\mathsf{priv}},\mathsf{FS}}(\lambda)$. In particular, the crs for Π is generated honestly using Gen; and the challenge functional signature $\sigma = (f_b(m_b))$ is generated as $\sigma \leftarrow \mathsf{FS.Sign}(\mathsf{sk}_{f_b}, m_b)$, i.e., by running $\pi \leftarrow \mathsf{Prove}(\mathsf{prs}, (M_L, x, t), w)$ where $x = (\mathsf{mvk}_0, f_b(m_b))$, $t = |x|^{e_L}$, and $w = (m_b, f_b, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma)$.

$\mathbf{G}_1$ is the same as $\mathbf{G}_0$ except that one uses the zero-knowledge simulator algorithm S in order to generate both the crs and the proof in the challenge. Namely, $(\mathsf{prs}, \mathsf{vst}, \mathsf{tr}) \leftarrow S^{\mathsf{crs}}(1^\lambda)$, and the challenge signature is generated by running $\pi \leftarrow S^{\mathsf{Prove}}(\mathsf{z}, \mathsf{prs}, (M_L, x, t), \mathsf{tr})$ for $x = (\mathsf{mvk}_0, m')$, $t = |x|^{e_L}$, where $m' = f_0(m_0) = f_1(m_1)$.

Denote by win_0 and win_1 the advantage of the adversary $\mathcal{A}_{\mathsf{priv}}$ in guessing the bit b in $\mathbf{G}_0$, and $\mathbf{G}_1$, respectively. Clearly $\mathsf{win}_1 = 1/2$ since the bit b is not used at all, and thus the view of $\mathcal{A}_{\mathsf{priv}}$ is independent of b. To complete the proof we show that under the assumption that Π is zero-knowledge, the following holds:

Claim 6.3.8. $\mathsf{win}_0 - \mathsf{win}_1 \leq negl(\lambda)$.

To prove this, we show that for any $\mathcal{A}_{\mathsf{priv}}$ such that $\mathsf{win}_0 - \mathsf{win}_1 = \varepsilon$ is non-negligible there is a distinguisher $\mathcal{D}$ that succeeds against the zero-knowledge property of Π with the same advantage ε. $\mathcal{D}$ is defined as follows:

$\mathcal{D}(\mathsf{crs})$:
Generate a pair $(\mathsf{msk}_0, \mathsf{mvk}_0) \leftarrow \mathsf{kg}_0(1^\lambda)$
Run $\mathcal{A}_{\mathsf{priv}}(\mathsf{mvk} = (\mathsf{crs}, \mathsf{mvk}_0))$
$\mathcal{A}_{\mathsf{priv}}$ adaptively chooses function queries f_0, f_1 and message pairs m_0, m_1
such that $f_0(m_0) = f_1(m_1)$:
For each f_b asked by $\mathcal{A}_{\mathsf{priv}}$, return the secret key $\mathsf{sk}_{f_b} \leftarrow \mathsf{FS.KeyGen}(\mathsf{msk}, f_b)$

To answer the challenge $\mathcal{D}$ proceeds as follows:
pick $b \xleftarrow{\$} \{0,1\}$
set $x = (\mathsf{mvk}_0, f_b(m_b))$, $t = |x|^{e_L}$, and $w = (m_b, f_b, \mathsf{vk}', \sigma_{\mathsf{vk}'}, \sigma)$
output (y, w) (where $y = (M_L, x, t)$) to its challenger, and get back π
return $(f_b(m_b), \pi)$ to $\mathcal{A}_{\mathsf{priv}}$
Let b' be $\mathcal{A}_{\mathsf{priv}}$'s output
If $b' = b$ output 1, else output 0.

Note that when $\mathcal{D}$ receives crs and π that are generated using the real algorithms, then $\mathcal{D}$ is perfectly simulating $\mathbf{G}_0$ to $\mathcal{A}_{\mathsf{priv}}$. Otherwise, if $\mathcal{D}$ receives crs and π that are generated using the simulator, then $\mathcal{D}$ perfectly simulates $\mathbf{G}_1$. Therefore it is easy to see that $\mathcal{D}$'s advantage is $\mathsf{win}_0 - \mathsf{win}_1$. □

Remark 6.3.9 (On the applicability of Corollary 5.4.7)**.** *For the same reasons discussed in Remark 6.2.5, it is not possible to apply the result of Corollary 5.4.7 to conclude the that the (adaptive) security of the functional signature scheme holds under classical SNARKs.*

Remark 6.3.10. *[On the use of multiple signing oracles] In order to prove the security of the functional signature scheme, one might be tempted to use the notion of O-SNARK with a single signing oracle. Precisely, one might use O-SNARKs for $\mathbb{O}_{\Sigma_0}$ when making a reduction to Σ_0 and O-SNARKs for $\mathbb{O}_{\Sigma'}$ when making a reduction to Σ'. Unfortunately, this approach does not work for an intricate technical reason that we explain here. Intuitively, assume that one wants to build an O-SNARK adversary $\tilde{\mathcal{A}}$ that has access to a single signing oracle, say from $\mathbb{O}_{\Sigma_0}$. Then the secret keys needed to simulate all the other oracles have to be given to $\tilde{\mathcal{A}}$ as part of its auxiliary input ($\tilde{\mathcal{A}}$ needs them to simulate $\mathcal{A}_{\mathsf{FS}}$). At this point the issue is that such secret keys in fact give an efficient way to compute a witness for several y in the relation $\mathcal{R}_L$. Therefore, if the extractor gets these secret keys as auxiliary information, we then have no guarantee that, while doing a reduction to the unforgeability of the signature scheme, the extractor will output a witness of the form we expect.*

6.4 SNARKs on Authenticated Data

As another application of O-SNARKs, we consider the generic construction of SNARKs on authenticated data that is given in [BBFR15]. Since this construction is very similar to the homomorphic signature scheme that we present in Section 6.2.1, we only provide an informal discussion of this application. In [BBFR15] Backes et al. introduce the notion of SNARKs on authenticated data to capture in an explicit way the possibility of performing (zero-knowledge) proofs about statements that are authenticated by third parties, i.e., to prove that $(x, w) \in \mathcal{R}$ for some x for which there is a valid signature. While the main focus of that work is on a concrete construction based on quadratic arithmetic programs, the authors also show a generic construction based on SNARKs and digital signatures. Roughly speaking, this construction consists in letting the prover use a SNARK to prove a statement of the form "$\exists x, w, \sigma : (x, w) \in \mathcal{R} \wedge \mathsf{vfy}(\mathsf{vk}, \tau | x, \sigma) = 1$", for some public label τ of the statement. The formalization of their model is rather similar to that of homomorphic signatures in this paper (e.g., they also use labels). Noticeable differences are that their construction uses pre-processing SNARKs for arithmetic circuit satisfiability, and that to handle several functions they use different SNARK instantiations (one per function).

In [BBFR15] the security proof of this generic construction is only sketched, and in particular, they use the existence of an extractor for an adversary that interacts with a signing oracle without providing a particular justification on its existence. With a more careful look, it is possible to see that this security proof incurs the same issue of extraction in the presence of oracles. Using the same techniques that we developed in this paper for the homomorphic signature scheme,[2] it is possible to prove the security of that generic construction using O-SNARKs for signing oracles (or non-adaptive security based on classical SNARKs). In conclusion, for this construction one can either conjecture that a specific SNARK scheme (e.g., [PHGR13]) is secure in the presence of oracles, or, more conservatively, argue only the non-adaptive security of the primitive under the existence of classical SNARKs.

6.5 Universal Signature Aggregators

In this section, we discuss another application where the combined use of SNARKs and digital signatures give rise to a natural construction with a difficult proof of security. The considered application is that of universal signatures aggregators [HKW15]. Informally, this cryptographic primitive generalizes the well-known notion of aggregate signatures [BGLS02] to a setting in which the signatures to aggregate are generated under different (already existing) schemes. This is in contrast to previous work on aggregate signatures in which solutions were ad-hoc, i.e., one designed a specific scheme with an aggregation capability.

6.5.1 Definition

We recall the definition of universal signature aggregators from [HKW15].

Let $\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}}$ be polynomials such that, for a given security parameter λ, $\ell_{\mathrm{ver}}(\lambda)$ is a bound on the size of the verification circuit, $\ell_{\mathrm{vk}}(\lambda)$ is a bound on the size of a verification key, $\ell_{\mathrm{msg}}(\lambda)$ is a bound on the size of a message, and $\ell_{\mathrm{sig}}(\lambda)$ is a bound on the size of signatures. For compactness, let $\ell = (\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$ be the tuple of these polynomials in which we drop λ (when this is clear from the context).

A universal signature aggregator ℓ-UAgg is a tuple of three algorithms ($\mathsf{UniversalSetup}$, $\mathsf{UniversalAgg}$, $\mathsf{UniversalVerify}$) working as follows.

$\mathsf{UniversalSetup}(1^\lambda)$ takes as input the security parameter and outputs public parameters pp.

$\mathsf{UniversalAgg}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)\}_{i=1}^n)$ is an algorithm that takes as input public parameters pp, and n tuples $(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)$ where vfy_i is the verification circuit of a signature scheme, vk_i is a public key, m_i a message, and σ_i a signature. Moreover, every tuple $(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)$ is supposed to be ℓ-length qualified. The algorithm outputs an aggregate signature σ_{agg}, of length polynomial in λ, but independent of n.

$\mathsf{UniversalVerify}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ is a (deterministic) algorithm that takes as input public parameters pp, an aggregate signature σ_{agg}, and t tuples $(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)$ where vfy_i is the verification circuit of a signature scheme, vk_i is a public key, and m_i a message. The algorithm outputs 0 (reject) or 1 (accept).

[2] The only significant difference is that one has to consider a specification of our definitions to the case of pre-processing SNARKs.

Correctness. Let $\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)\}_{i=1}^n$ be a collection of tuples such that $\mathsf{vfy}_i(\mathsf{vk}_i, m_i, \sigma_i) = 1$ for all $i = 1$ to n. Then, for all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{UniversalSetup}(1^\lambda)$ and $\sigma_{agg} \leftarrow \mathsf{UniversalAgg}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)\}_{i=1}^n)$ it must be $\mathsf{UniversalVerify}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)\}_{i=1}^t, \sigma_{\mathsf{agg}}) = 1$.

Security. Let $\Sigma = (\mathsf{kg}, \mathsf{sign}, \mathsf{vfy})$ be an unforgeable signature scheme. The security property of a universal signature aggregator scheme is defined via the following experiment, denoted $\mathsf{Exp}_{\mathcal{A},\Sigma,\mathsf{UAgg}}(\lambda)$, between an adversary and a challenger.

Setup The challenger generates $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{kg}(1^\lambda)$ and $\mathsf{pp} \xleftarrow{\$} \mathsf{UniversalSetup}(1^\lambda)$ and gives vk, pp to $\mathcal{A}$.

Signing queries The adversary adaptively asks for signatures on messages of its choice, i.e., $\mathcal{A}$ sends m, and the challenger answers with $\sigma \xleftarrow{\$} \mathsf{sign}(\mathsf{sk}, m)$.

Forgery $\mathcal{A}$ outputs a tuple $\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)\}_{i=1}^n$ and a signature σ_{agg}.

The experiment outputs 1 (i.e., $\mathcal{A}$ wins) if $\mathsf{UniversalVerify}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)\}_{i=1}^n, \sigma_{\mathsf{agg}}) = 1$ and there exist an index $j \in [n]$ such that $\mathsf{vfy}_j = \mathsf{vfy}$, $\mathsf{vk}_j = \mathsf{vk}$, and the message m_j was not queried in the experiment. We let the advantage of $\mathcal{A}$ be $\mathsf{Adv}_{\mathcal{A},\Sigma,\mathsf{UAgg}}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{A},\Sigma,\mathsf{UAgg}}(\lambda) = 1]$.

A universal signature aggregator scheme UAgg is secure if for all PPT adversaries $\mathcal{A}$ there is a negligible function ε such that $\mathsf{Adv}_{\mathcal{A},\Sigma,\mathsf{UAgg}}(\lambda) \leq \varepsilon(\lambda)$.

Weak Security. In this chapter, we also consider a weaker security notion for universal signature aggregators in which adversaries make (adaptive) signing queries *before* receiving the public parameters of the universal aggregator. We call this notion *weak security*.

6.5.2 Universal Signatures Aggregators from SNARKs

In this section, we show how to construct universal signatures aggregators by using a SNARK for NP. This construction is rather simple and can be seen as a natural way to solve the universal aggregation problem. The idea is to simply prove knowledge of valid signatures and rely on the SNARK's succinctness to argue the shortness of the aggregator.

In this work, however, our interest is on proving the security of this construction, and understanding under what assumptions it can be based. Indeed, despite the simplicity of the construction, the proof is tricky due to the issues discussed in this work, that is the need to run the SNARK extractor in an experiment in the presence of a (signing) oracle.

We address this problem by giving three different results on the security of this construction. Letting Π be the SNARK scheme used in the aggregator construction:

(1) We prove security by assuming that Π is an O-SNARK. Next, we can combine this fact with our existence results of O-SNARKs for signing oracles (Corollary 5.4.7), and then conclude security of our aggregator if it is used with signature schemes that have "small" (i.e., polynomially large) message spaces. Alternatively, security holds for schemes with superpolynomially large message spaces if one relies on sub-exponential hardness of both Π and the signature scheme.

(2) We prove the aggregator construction *weakly* secure by assuming that Π is a non-adaptive O-SNARK, which can be later reduced to assuming that Π is a SNARK for arbitrary auxiliary input.

(3) We prove the (adaptive) security of the aggregator when it is restricted to the use of *unique* signature schemes that are unforgeable based on a non-interactive problem. In this case, we can prove security of our aggregator by only assuming that Π is a SNARK for a specific (rather benign) auxiliary input. For this last proof, we develop a new technique, that is also interesting as it bypasses our impossibility results in a non-trivial way.

Below we describe our construction; the security proofs are presented in the following sections.

Defining the Machine $M_{\ell,t}$. Let $\ell = (\ell_{\mathrm{ver}}, \ell_{\mathrm{vk}}, \ell_{\mathrm{msg}}, \ell_{\mathrm{sig}})$ and n be polynomials in λ as defined before. For any such ℓ, n, we define $M_{\ell,n}(x, w)$ to be the random-access machine that works as follows. It takes inputs (x, w) where $x \in \{0,1\}^{n(\ell_{\mathrm{ver}}+\ell_{\mathrm{vk}}+\ell_{\mathrm{msg}})}$ is parsed as $x = (\mathsf{vfy}_1, \mathsf{vk}_1, m_1, \ldots, \mathsf{vfy}_n, \mathsf{vk}_n, m_n)$, and $w \in \{0,1\}^{n\ell_{\mathrm{sig}}}$ is parsed as $w = (\sigma_1, \ldots, \sigma_n)$. Then, given such a pair (x, w), $M_{\ell,n}$ accepts iff

$$\bigwedge_{i=1}^{n} \mathsf{vfy}_i(\mathsf{vk}_i, m_i, \sigma_i) = 1$$

Associated to such machine there is also a polynomial time bound $t_{\ell,n}(k) = k^{e_{\ell,n}}$, such that $M_{\ell,n}$ rejects if it does more than $t_{\ell,n}(|x|)$ steps. Note that from a given ℓ, n, such constant $e_{\ell,n}$ can be derived efficiently from a time upper bound on the evaluation of circuits of given size. We call $\mathcal{R}_{\ell,n}$ the NP binary relation consisting of all pairs (y, w) such that, parsing $y = (M_{\ell,n}, x, t)$, $M_{\ell,n}(x, w)$ accepts in at most t steps and $t \leq t_{\ell,n}(|x|)$.

The Construction. Let Π be a fully-succinct SNARK for NP. We define the universal signature aggregator scheme $\mathsf{UAgg}[\Pi]$ as follows.

$\mathsf{UniversalSetup}(1^\lambda)$ runs $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and outputs $\mathsf{pp} = \mathsf{crs}$.

$\mathsf{UniversalAgg}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i, \sigma_i)\}_{i=1}^n)$ aggregation proceeds as follows:

- build strings $x = (\mathsf{vfy}_1, \mathsf{vk}_1, m_1, \ldots, \mathsf{vfy}_n, \mathsf{vk}_n, m_n)$ and $w = (\sigma_1, \ldots, \sigma_n)$;
- run $M_{\ell,n}(x, w)$ letting $t = \#M_{\ell,n}(x, w)$ be the number steps;
- set $y = (M_{\ell,n}, x, t)$ and run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$;
- output $\sigma_{\mathsf{agg}} := \pi$.

$\mathsf{UniversalVerify}(\mathsf{pp}, \{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ to verify the aggregate signature σ_{agg} proceed as follows

- build string $x = (\mathsf{vfy}_1, \mathsf{vk}_1, m_1, \ldots, \mathsf{vfy}_n, \mathsf{vk}_n, m_n)$;
- reconstruct constant $e_{\ell,n}$ such that $t_{\ell,n}(|x|) = |x|^{e_{\ell,n}}$;
- set $y = (M_{\ell,n}, x, |x|^{e_{\ell,n}})$ and run $b \leftarrow \mathsf{Ver}_{e_{\ell,n}}(\mathsf{crs}, x, \pi)$;
- output b.

6.5.3 Security from O-SNARKs

In this section, we show that the construction UAgg described in the previous section is a secure universal signature aggregator based on the unforgeability of signature schemes and by assuming that Π is an O-SNARK for the corresponding family of signing oracles.

Theorem 6.5.1. *For all ℓ-length qualified unforgeable signature schemes Σ, if Π is an O-SNARK for $\mathbb{O}_\Sigma$, then ℓ-$\mathsf{UAgg}[\Pi]$ is a secure universal signature aggregator with respect to Σ.*

Proof. We proceed by contradiction, assuming the existence of an algorithm $\mathcal{A}$ that breaks the security of UAgg with non-negligible advantage ε. Starting from this $\mathcal{A}$, we show how to build an efficient forger $\mathcal{F}$ for Σ. Along the way of the reduction, we rely on that Π is an O-SNARK for oracles from $\mathbb{O}_\Sigma$.

First of all, for any adversary $\mathcal{A}$ that outputs a forgery $(\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i^*)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ we define an adversary $\tilde{\mathcal{A}}^{\mathcal{O}}(\mathsf{crs})$, with $\mathcal{O} \xleftarrow{\$} \mathbb{O}_\Sigma$ working as follows: it queries $\mathsf{vk} \leftarrow \mathcal{O}(\text{`}vk')$; it runs $\mathcal{A}^{\mathsf{sign}(\cdot)}(\mathsf{pp} := \mathsf{crs}, \mathsf{vk})$ simulating all signing queries using $\mathcal{O}(\cdot)$; finally it returns (y, π) where $y = (M_{\ell,n}, x, |x|^{e_{\ell,n}})$ is reconstructed as in $\mathsf{UniversalVerify}$ and $\pi = \sigma_{\mathsf{agg}}$. The adversary $\tilde{\mathcal{A}}$ perfectly fits the O-SNARK definition, by which there exists an extractor $\mathcal{E}_{\tilde{\mathcal{A}}}$ that given the same input of $\tilde{\mathcal{A}}$ and the transcript of oracle queries/answers made and received by $\tilde{\mathcal{A}}^{\mathcal{O}}$, outputs a w such that (y, π) verifies and $(y, w) \notin \mathcal{R}_{\ell,n}$ happens only with negligible probability ν.

Using $\mathcal{A}$ and the above extractor, we build a forger $\mathcal{F}^{\mathcal{O}}$ that breaks that $\mathsf{UF\text{-}CMA}$ security of Σ as follows:

$\mathcal{F}^{\mathcal{O}}(\mathsf{vk})$:
- Initialize $\mathsf{qt} \leftarrow (\text{`}vk', \mathsf{vk})$
- Generate $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and run $\mathcal{A}(\mathsf{pp} = \mathsf{crs}, \mathsf{vk})$
- Simulate signing queries $\mathcal{O}(m)$ as follows:
  - query $\sigma \leftarrow \mathcal{O}(m)$ and add (m, σ) to qt
  - output σ
- Let $(\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i^*)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ be the tuple returned by $\mathcal{A}$
- Reconstruct y as in $\mathsf{UniversalVerify}$
- If $\mathsf{Ver}(\mathsf{crs}, y, \sigma_{\mathsf{agg}}) = 1 \wedge (\exists j \in [n] : \mathsf{vfy}_j = \mathsf{vfy} \wedge \mathsf{vk}_j = \mathsf{vk} \wedge (m_j^*, \cdot) \notin \mathsf{qt})$:
  - Take the first $j \in [n]$ such that $\mathsf{vfy}_j = \mathsf{vfy}$ and $\mathsf{vk}_j = \mathsf{vk}$ and $(m_j^*, \cdot) \notin \mathsf{qt}$
  - Execute $w \leftarrow \mathcal{E}_{\tilde{\mathcal{A}}}(\mathsf{crs}, \mathsf{qt})$
  - If $(y, w) \notin \mathcal{R}_{\ell,n}$: output $\perp$
  - Else: parse $w = (\sigma_1^*, \ldots, \sigma_n^*)$
    - Output (m_j^*, σ_j^*).
- Else: Output $\perp$

Whenever $\mathcal{A}$ succeeds, unless $\mathcal{F}$ outputs $\perp$ because $(y, w) \notin \mathcal{R}_{\ell,n}$, one can see that $\mathcal{F}$ is successful in breaking the unforgeability of Σ. By the adaptive knowledge soundness with respect to $\mathbb{O}_\Sigma$, $\mathcal{F}$ fails due to $(y, w) \notin \mathcal{R}_{\ell,n}$ only with negligible probability ν. Therefore, if $\mathcal{A}$ succeeds with probability ε, $\mathcal{F}$ outputs a valid forgery with probability at least $\varepsilon - \nu$.

□

Theorem 6.5.2. *For all ℓ-length qualified unforgeable signature schemes Σ, if Π is a SNARK (for arbitrary auxiliary input), then ℓ-$\mathsf{UAgg}[\Pi]$ is a weakly secure universal signature aggregator with respect to Σ.*

Proof. This proof is very similar to that of the previous theorem. The only difference is that here we use the adversary $\mathcal{A}$ for the weak security of UAgg to define a *non-adaptive* O-SNARK adversary $\tilde{\mathcal{A}}$. The rest of the proof proceeds analogously. □

6.5.4 Security for Unique Signatures from SNARKs

Here we show that the scheme of Section 6.5.2 can be proven secure by only relying on the classical knowledge soundness property of the SNARK Π. For this, however, we have to restrict the use to signature schemes that have unique signatures, and that are proven unforgeable under a non-interactive hard problem.

Interestingly, we show how to leverage these restrictions to give a proof of security that bypasses our impossibility results. At the core of this result is a technique which uses the security reduction of the signature scheme to construct a SNARK adversary that does not have access to any oracle, and for which the existence of an extractor can thus be argued using classical definitions.

6.5.4.1 Non-Interactive Computational Problems and Reductions.

We formally define the type of computational problems and reductions that we use in our proofs.

Definition 6.5.3 (Non-Interactive Computational Problem)**.** *A non-interactive computational problem $P = (\mathsf{IG}, \mathsf{V})$ is defined by two algorithms:*

Instance Generation $\mathsf{IG}(1^\lambda)$*: on input the security parameter, it outputs an instance I.*

Instance Verification $\mathsf{V}(I, S)$*: on input an instance I and a value S (a possible solution) it outputs a decision bit.*

We say that an algorithm $\mathcal{A}$ (ε, t)-solves P if $\mathcal{A}$ runs in time t and

$$\Pr[\mathsf{V}(I, S) = 1 \mid I \xleftarrow{\$} \mathsf{IG}(1^\lambda); S \leftarrow \mathcal{A}(1^\lambda, I)] \geq \varepsilon$$

A problem P is said hard *if there is no $\mathcal{A}$ that (ε, t)-solves P with $t = \mathsf{poly}(\lambda)$ and $\varepsilon = 1/\mathsf{poly}(\lambda)$.*

Next, we formalize the type of reductions considered in our proofs. Intuitively, a black-box reduction is an algorithm that can solve a certain problem P by interacting with an adversary which breaks the security of a particular cryptographic scheme. In our work, we are specifically interested into reductions from the unforgeability of signature schemes to non-interactive computational problems. Moreover, we consider reductions that invoke the underlying adversary only once. As in [LW14] we call these *simple reductions*.

Definition 6.5.4 (Simple Reductions for Signature Schemes)**.** *An algorithm $\mathcal{R}$ is a simple $(\varepsilon, t, q, \delta, t')$-reduction from the unforgeability of a signature scheme Σ to a computational problem P if, given black-box access to any adversary $\mathcal{A}$ that (ε, t, q)-breaks the security of Σ, $\mathcal{R}$ (δ, t')-solves the problem P, after simulating the security experiment* once *for $\mathcal{A}$.*

A simple reduction $\mathcal{R}$ from unforgeability of signatures to a non-interactive computational problem can be modeled as a stateful algorithm such that:

- It is first invoked on input a problem instance I, and it outputs a public key vk and state st_0: $(\mathsf{vk}, st_0) \leftarrow \mathcal{R}(I)$.
- Later, for $i = 1$ to Q (for some $Q = \mathsf{poly}(\lambda)$ which depends on $\mathcal{A}$), it is executed on input its previous state st_{i-1} and a message m_i and outputs a signature σ_i and the successive state st_i: $(\sigma_i, st_i) \leftarrow \mathcal{R}(m_i, st_{i-1})$.

- Finally, at some point of its execution (i.e., for any $j \geq 0$), it can be invoked on input a message-signature pair, and outputs a solution S: $S \leftarrow \mathcal{R}(m^*, \sigma^*, st_j)$.

Proof of Adaptive Security. We are now ready to give our security proof for the universal aggregator scheme proposed in Section 6.5.2.

Theorem 6.5.5. *For all ℓ-length qualified unique signature schemes Σ that have a simple security reduction to a non-interactive computational problem P, ℓ-$\mathsf{UAgg}[\Pi]$ is a secure universal signature aggregator with respect to Σ, if Π is a SNARK for auxiliary input consisting of an instance of P and a random string.*

Proof. Assume by contradiction the existence of a PPT adversary $\mathcal{A}$ against the security of UAgg such that $\Pr[\mathsf{Exp}_{\mathcal{A},\Sigma,\mathsf{UAgg}}(\lambda) = 1] \geq \varepsilon$ for some non-negligible ε. Let $\mathcal{R}$ be the simple reduction from the unforgeability of Σ to the hardness of the problem P.

An Ideal Solver. To begin with, we show that by using the adversary $\mathcal{A}$ against UAgg and the reduction $\mathcal{R}$, it is possible to define an algorithm $\mathcal{IF}$ that solves P with non-negligible probability. This algorithm $\mathcal{IF}$ is, however, "ideal" in the sense that one of its steps cannot be executed in polynomial time. Its definition will ease our analysis later.

$\mathcal{IF}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})$:

1 Generate $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$
2 Initialize $\mathcal{T} \leftarrow \emptyset$
3 Execute $(\mathsf{vk}, st) \leftarrow \mathcal{R}(I; \rho_{\mathcal{R}})$ to obtain a public key vk of Σ
4 Run $\mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{vk}; \rho_{\mathcal{A}})$ and simulate every signing query $\mathcal{O}(m)$ as follows:
5 Compute $(\sigma, st) \leftarrow \mathcal{R}(m, st)$, add $\mathcal{T} \leftarrow \mathcal{T} \cup \{m\}$
6 Output σ to $\mathcal{A}$
7 Let $(\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i^*)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ be the tuple returned by $\mathcal{A}$
8 Reconstruct y as in $\mathsf{UniversalVerify}$
9 If $\mathsf{Ver}(\mathsf{crs}, y, \sigma_{\mathsf{agg}}) = 1 \wedge (\exists j \in [n] : \mathsf{vfy}_j = \mathsf{vfy} \wedge \mathsf{vk}_j = \mathsf{vk} \wedge m_j^* \notin \mathcal{T})$:
10 Take the first $j \in [n]$ such that $\mathsf{vfy}_j = \mathsf{vfy}$ and $\mathsf{vk}_j = \mathsf{vk}$ and $m_j^* \notin \mathcal{T}$
11 Search for $\sigma^* \in \{0,1\}^{\ell_{\mathrm{sig}}}$ such that $\mathsf{vfy}(\mathsf{vk}, m_j^*, \sigma_j^*) = 1$
12 Run $S \leftarrow \mathcal{R}(m_j^*, \sigma_j^*, st)$
13 Output S.
14 Else: Output $\perp$

$\mathcal{IF}$ takes as input an instance I of the problem P and a set of random coins that we write explicitly. For ease of analysis, we view the coins as three separate strings $\rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}}$. The strings $\rho_{\mathcal{R}}$ and $\rho_{\mathcal{A}}$ are the set of coins provided to $\mathcal{R}$ and $\mathcal{A}$ respectively, whereas ρ_{crs} are the coins used to generate crs on line 1.

Due to line 11, $\mathcal{IF}$ is not a polynomial-time algorithm. However, notice that $\mathcal{IF}$ is (internally) breaking the unforgeability of Σ with the same probability ε with which $\mathcal{A}$ breaks UAgg. Since $\mathcal{IF}$ interacts with the reduction $\mathcal{R}$ as if it was a forger, we have that $\mathcal{R}$ succeeds with probability $\geq \delta$ in solving P, and so does $\mathcal{IF}$, i.e.,

$$\Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} [\mathsf{V}(I, S) = 1 \mid S \leftarrow \mathcal{IF}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})] \geq \delta \tag{6.4}$$

Existence of an Extractor. Here we show that from any adversary $\mathcal{A}$ against UAgg we can define a knowledge extractor that outputs the witness corresponding to the statement used

by $\mathcal{A}$ in its forgery. To argue the existence of this extractor, we use the adversary $\mathcal{A}$ and the reduction $\mathcal{R}$ to define an algorithm $\mathcal{B}$ that takes as input a common reference string crs of Π, an auxiliary input consisting of an instance I of problem P, two random strings $\rho_{\mathcal{R}}, \rho_{\mathcal{A}}$, and it proceeds as follows:

$\mathcal{B}(\mathsf{crs}, I, \rho_{\mathcal{R}}, \rho_{\mathcal{A}})$:
1 Execute $(\mathsf{vk}, st) \leftarrow \mathcal{R}(I; \rho_{\mathcal{R}})$ to obtain a public key vk of Σ
2 Run $\mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{vk}; \rho_{\mathcal{A}})$ and simulate every signing query $\mathcal{O}(m)$ as follows:
3 Compute $(\sigma, st) \leftarrow \mathcal{R}(m, st)$,
4 Output σ to $\mathcal{A}$
5 Let $(\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i^*)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ be the tuple returned by $\mathcal{A}$
6 Reconstruct y as in $\mathsf{UniversalVerify}$
7 Output $(y, \sigma_{\mathsf{agg}})$.

As one can see, $\mathcal{B}$ can be seen as an adversary for the adaptive proof of knowledge, taking an auxiliary input from distribution $\mathcal{Z}$, i.e., consisting of $I \in \mathsf{IG}(\lambda)$ and two random strings. By the adaptive knowledge soundness property of Π we then have that for any such $\mathcal{B}$ there exists an extractor $\mathcal{E}_{\mathcal{B}}$ that, given the same input of $\mathcal{B}$, outputs a witness w such that the joint probability that $(y, \sigma_{\mathsf{agg}})$ verifies correctly and $(y, w) \notin \mathcal{R}_{\ell,n}$ is negligible. That is, for any $\mathcal{B}$ there is $\mathcal{E}_{\mathcal{B}}$ and a negligible function ν such that

$$\Pr[\mathsf{AdPoK}(\lambda, \mathcal{B}, \mathcal{E}_{\mathcal{B}}, \mathcal{Z}) \Rightarrow 1] \leq \nu \tag{6.5}$$

Building the Efficient Solver. We are now ready to describe the main part of our proof, that is the description of an algorithm $\mathcal{F}$ that, by using the above extractor, the adversary $\mathcal{A}$ and the reduction $\mathcal{R}$, is able to solve P with non-negligible probability. The algorithm $\mathcal{F}$ works as follows.

$\mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})$:
1 Generate $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$
2 Initialize $\mathcal{T} \leftarrow \emptyset$
3 Execute $(\mathsf{vk}, st) \leftarrow \mathcal{R}(I; \rho_{\mathcal{R}})$ to obtain a public key vk of Σ
4 Run $\mathcal{A}^{\mathcal{O}}(\mathsf{crs}, \mathsf{vk}; \rho_{\mathcal{A}})$ and simulate every signing query $\mathcal{O}(m)$ as follows:
5 Compute $(\sigma, st) \leftarrow \mathcal{R}(m, st)$, and add $\mathcal{T} \leftarrow \mathcal{T} \cup \{m\}$
6 Output σ to $\mathcal{A}$
7 Let $(\{(\mathsf{vfy}_i, \mathsf{vk}_i, m_i^*)\}_{i=1}^n, \sigma_{\mathsf{agg}})$ be the tuple returned by $\mathcal{A}$
8 Reconstruct y as in $\mathsf{UniversalVerify}$
9 If $\mathsf{Ver}(\mathsf{crs}, y, \sigma_{\mathsf{agg}}) = 1 \wedge (\exists j \in [n] : \mathsf{vfy}_j = \mathsf{vfy} \wedge \mathsf{vk}_j = \mathsf{vk} \wedge m_j^* \notin \mathcal{T})$:
10 Take the first $j \in [n]$ such that $\mathsf{vfy}_j = \mathsf{vfy}$ and $\mathsf{vk}_j = \mathsf{vk}$ and $m_j^* \notin \mathcal{T}$
11 Execute $w \leftarrow \mathcal{E}_{\mathcal{B}}(\mathsf{crs}, I, \rho_{\mathcal{R}}, \rho_{\mathcal{A}})$
12 If $(y, w) \notin \mathcal{R}_{\ell,n}$: output $\perp$
13 Else: parse $w = (\sigma_1^*, \ldots, \sigma_n^*)$
14 Run $S \leftarrow \mathcal{R}(m_j^*, \sigma_j^*, st)$ and output S.
15 Else: Output $\perp$

$\mathcal{F}$ proceeds almost identically to our ideal solver $\mathcal{IF}$ except that the line 11 of $\mathcal{IF}$ is replaced by lines 11–13 in $\mathcal{F}$. Namely, instead of brute forcing the search for the signature σ_j^*, $\mathcal{F}$ makes use of the extractor $\mathcal{E}_{\mathcal{B}}$.

In what follows we analyze success probability of $\mathcal{F}$.

Let Bad be the event that $\mathcal{F}$ outputs $\perp$ in line 12. Then we have:

$$\Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} [\mathsf{V}(I,S) = 1 \mid S \leftarrow \mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})]$$

$$\geq \Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} \left[\mathsf{V}(I,S) = 1 \wedge \overline{\mathsf{Bad}} \mid S \leftarrow \mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})\right]$$

$$\geq \Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} \left[\mathsf{V}(I,S) = 1 \mid S \leftarrow \mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}}) \wedge \overline{\mathsf{Bad}}\right] - \Pr[\mathsf{Bad}]$$

By noting that lines 3–8 of $\mathcal{F}$ are the same as lines 1–6 of $\mathcal{B}$ (except for some bookkeeping in $\mathcal{T}$), we can use equation Equation (6.5) to bound $\Pr[\mathsf{Bad}] \leq \nu$.

To conclude the proof we claim that

$$\Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} \left[\mathsf{V}(I,S) = 1 \mid S \leftarrow \mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}}) \wedge \overline{\mathsf{Bad}}\right] = \Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} [\mathsf{V}(I,S) = 1 \mid S \leftarrow \mathcal{IF}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})]$$

The above equality follows by observing two main facts in the execution of $\mathcal{F}$. First, since Bad did not occur, we have $(y, w) \in \mathcal{R}_{\ell,n}$ which implies $\mathsf{vfy}(\mathsf{vk}, m_j^*, \sigma_j^*) = 1$. Second, by the uniqueness of Σ, given vk and m_j^* there is only one string σ_j^* which makes the verification algorithm accept. In particular, this is the *very same* string that is computed in line 11 of $\mathcal{IF}$. Essentially, in the view of $\mathcal{R}$ its execution interacting with $\mathcal{IF}$ is identical to the execution interacting with $\mathcal{F}$. Hence, the fact that $\mathcal{E}_{\mathcal{B}}$ gets to see the coins of $\mathcal{R}$ (at that point of the execution) does not give it an additional power to let the reduction fail.

Therefore, by putting together the bounds given above, we have that $\mathcal{F}$ solves P with non-negligible probability

$$\Pr_{\substack{I \xleftarrow{\$} \mathsf{IG}(1^\lambda) \\ \rho_{\mathcal{R}},\ \rho_{\mathsf{crs}},\ \rho_{\mathcal{A}}}} [\mathsf{V}(I,S) = 1 \mid S \leftarrow \mathcal{F}(I; \rho_{\mathcal{R}}, \rho_{\mathsf{crs}}, \rho_{\mathcal{A}})] \geq \delta - \nu$$

which concludes the proof. □

Chapter 7

SNARKs with Data Privacy

Boosting Verifiable Computation on Encrypted Data. We have considered along this thesis the setting in which a user delegates computation to an untrusted server and then uses SNARKs to check the correctness of the results. In this study, we never took into account the other downside of this scenario, more precisely, the confidentiality concern. In this chapter, we aim for solutions where the untrusted server does not learn information about the data it computes on, while it is still prevented from cheating on the results.

Contents

Tale five: A research problem

Alice is a grown-up now, she had to choose between her passions, and she has decided to pursue an academic career as a biologist researcher.
In her exciting life as a researcher, she still encounters cryptographic problems.
At this moment, she would like to make an ambitious study on cancer cells.
However, for this study to be relevant, she needs to analyse big databases with information about such cells.
To obtain such a collection of data on cancer cells, she would need to contact some hospital and ask them for sharing their records.
Also, another obstacle to her approach is the limitations of her laboratory.
They do not possess the computational means to run very complicated analysis algorithms on the data. She needs instead to contact a Statistical Center that provides computational services to researchers.
This threesome Alice-Hospital-Statistical Center makes the things very complicated. First, because the hospital Alice contacted wants to keep its sensitive data private from the Statistical Center, and even from Alice.
Ideally, the hospital would like the Statistical Center to learn nothing on the data, and Alice to learn only the global results she is interested in, but not the precise details about the patients and their medical condition.
Alice has a solution in mind in order to enable this collaboration, in a way that respects the needs of all parties.
The hospital can encrypt the data using the public key of Alice. There are encryption schemes that enable specific manipulations on the encrypted data, meaning that someone can compute functions on the ciphertexts and obtain a ciphertext of the result.
This allows the Statistical Center to do the necessary analysis on the encypted data without learning anything. The output from the Center will be just an encrypted result that will be further transmitted to Alice.
Alice then, having her secret key, can open this encrypted result of the analysis and obtain the values that she needs for her further interpretations and researches.
Like that, she would not have any idea of the initial data from patients used to compute this result.
Alice is also concerned about the validity of these results and she does not trust the Statistical Center completely. She would like to be able to verify the correctness of the result even without knowing the initial data used in the computation. She is not sure that a usual SNARK, like the ones seen before would do the job...
If she would be able to find such a complex scheme that protects the medical data of the patients (age, gender, medical history, illnesses, etc.) but at the same time enables verifiable computation on it, the confidentiality problem of the hospital would be solved and this would eventually help the advancement of her crucial study.
Alice's mission now is to apply cryptography in the service of medical research advancement.

7.1 Introduction

We have seen that SNARKs are a powerful tool to enable integrity of delegated computations. However, we did not discussed the problem of data confidentiality in the previous chapters. This can be a big concern when we outsource data and computation. We will discuss in this chapter some solutions intended to deal with both the problem of data privacy and computation integrity.

Privacy-Preserving Computation. The problem of data privacy is related to fully homomorphic encryption (FHE) [RAD78, Gen09]. While for a long time it was only known how to construct homomorphic encryption schemes supporting a single operation (e.g., only addition [Pai99] or multiplication [ElG84]), Gentry's breakthrough showed the first FHE scheme that enables computing any function on encrypted data. If Gentry's first FHE was mostly a feasibility result, research in this area has progressed significantly giving rise to many new FHE schemes (e.g., [SV10, BV11b, BGV12, GSW13, DM15, CGGI16, CGGI17]) that are efficient and see their first practical applications.

Ensuring Correctness of Computation. The second problem is related to verifiable computation (VC) [GGP10] and related notions such as interactive proofs [GMR85], probabilistically checkable proofs [AS92] and succinct arguments [Kil92]. Briefly speaking, these are protocols that enable a powerful prover to convince a verifier that a statement (e.g., the correctness of a computation, $y = f(x)$) is true in such a way that the verifier can run with fewer resources, e.g., faster than re-executing the function. Similarly to FHE, also in this research area, results have been confined to theory for a long time. However, several recent works have shown a change in this trend, and today we have several VC protocols that are efficient and have been experimented in practical scenarios, see e.g., [GKR08, CMT12, GGPR13, PHGR13, BCG+13, ZGK+17, WJB+17, AHIV17, WTS+18] and references therein.

7.1.1 Ensuring Correctness of Privacy-Preserving Computation

Despite the research mentioned above, the problem of ensuring both the correctness and the privacy of computation performed on untrusted machines has received much less attention in the literature. There are three main works that considered explicitly this problem.

The first one is the seminal paper of Gennaro et al. [GGP10] who introduced the notion of non-interactive verifiable computation. In [GGP10] they indeed show how to combine garbled circuits and FHE in order to build a VC scheme for arbitrary functions that also preserves the privacy of the computation's inputs and outputs against the computing machine.

The second work is that of Goldwasser et al. [GKP+13] that shows how to use their succinct single-key functional encryption scheme in order to build a VC protocol that preserves the privacy of the inputs (but not of the outputs).

Both these two solutions [GGP10, GKP+13] are however not very satisfactory in terms of efficiency. The main issue in the construction of [GGP10] is that they need the full power of FHE to perform homomorphic evaluations of garbled circuits. Some of the efficiency issues in [GKP+13] include the use of several instances of an attribute-based encryption that must support an expressive class of predicates (at NC1 circuits), and an inherent design limitation (due to following the approach of [PRV12]) by which their scheme supports functions with a single bit of output (which in practical scenarios like computing on large integers would require multiple instances of their protocol).

A third work that considered the problem of ensuring correctness of privacy-preserving computation is the one of Fiore et al. [FGP14] who proposed a solution that combines an FHE and a VC scheme. The idea of their generic construction is rather simple and consists into using a VC in order to prove that the homomorphic evaluation on ciphertexts has been done correctly. As discussed in [FGP14], even this solution may encounter efficiency limits. This is due to the fact that the VC scheme must be executed on a computation that, due to the FHE ciphertext expansion, is of much larger representation than the computation that would be executed on plain text. Motivated by this issue, [FGP14] also proposed an efficient solution that, for the case of quadratic functions, can avoid this issue. The efficient construction in [FGP14] overcomes the problem of ciphertext expansion in two ways: (1) they consider homomorphic encryption schemes working in the Ring-LWE setting in which ciphertexts are represented by polynomials in a given polynomial ring; (2) they develop, as the VC building block, an homomorphic MAC scheme especially tailored to handle messages that are polynomials in which the prover execution can be independent of the degree of such polynomials. However, for reasons that we will detail later (see Section 7.6), their technique is inherently bound to computations of multiplicative depth 1. Also, by using an homomorphic MAC as VC, verification requires a secret key, the same secret key used to encode the inputs. This limits the applicability of these solutions to scenarios where users and verifiers are either the same entity or they share a secret key.

7.1.2 Our Results

We propose a new protocol for verifiable computation on encrypted data that improves on the state-of-the-art solution of Fiore et al. [FGP14] in multiple aspects. Notably, we can support HE computations of multiplicative depth larger than 1. Second, we achieve public verifiability whereas [FGP14] is only privately verifiable. Finally, our scheme has an additional property that guarantees that verifiers may be convinced of outputs correctness without learning information on the original inputs. This latter property is particularly relevant in the publicly verifiable setting where the users who encrypt the data and the verifiers are distinct entities. Technically, we achieve this property because our protocol allows for re-randomizing the encrypted results, which was not possible in [FGP14] that only considered deterministic HE evaluations.

Our key tool to obtain this result is a new SNARK that can efficiently handle computations that are arithmetic circuits f over a quotient polynomial ring $\mathbb{R}_q \coloneqq \mathbb{Z}_q[X]/\langle R(X)\rangle$ (exactly like the popular choice for many Ring-LWE schemes) in which the prover's costs have a minimal dependence on the degree d of $R(X)$. Specifically, let f be the circuit over $\mathbb{R}_q$ and $\widehat{f}$ be the corresponding circuit over $\mathbb{Z}_q$ (i.e., the one that would be computed on plaintexts where additions and multiplications in $\mathbb{R}_q$ are replaced by the corresponding operations in $\mathbb{Z}_q$). Then, whereas a naive application of [FGP14]'s generic solution would incur into prover's costs at least $O(d \cdot |\widehat{f}|)$ where $|\widehat{f}|$ is $\widehat{f}$'s circuit size, our scheme lets the prover work in time $O(d \cdot n + |\widehat{f}|)$ where n is $\widehat{f}$'s input size. To see how this efficiency feature concretely improves, consider for example an $\widehat{f}$ that is a multivariate polynomial of degree $c \geq 2$ by which $|\widehat{f}|$ can be n^c, and consider that for Ring-LWE security the degree d can be a rather large integer (e.g., $d \approx 8000$). Then removing the multiplicative factor $d \cdot |\widehat{f}|$ can significantly speed the prover's costs. Let us also notice that the factor $d \cdot n$ is unavoidable as the prover must read the input.

Our SNARK for arithmetic circuits over polynomial rings is built in a modular way

using two building blocks: (1) a commit-and-prove SNARK for arithmetic circuits (e.g., [CFH+15, Vee17]) and (2) a dedicated (commit-and-prove) SNARK for polynomial evaluations that we develop in this paper. This scheme is our main technical result and we believe it can be of independent interest.

7.2 New Bivariate Computational Assumptions

In this section we introduce some assumptions in the same spirit as *q-type* assumptions mentioned in Section 2.3.2. They are simple extensions of these well-known assumptions to bivariate setting and they are assumed to hold in a bilinear group. We refer to Figure 2.3 for the definition and properties of a bilinear group of prime order.

7.2.0.1 The (d,ℓ)–Bivariate PKE Assumption ((d,ℓ)-BPKE).

We introduce a bivariate power knowledge of exponent assumption that suits our purposes. This is a simple extension.

The (d,ℓ)–Bivariate Power Knowledge of Exponent Assumption for a bilinear group $(q,\mathbb{G},\mathfrak{G},\mathbb{G}_T,e)$, noted by (d,ℓ)-BPKE is a hybrid between PKE assumption for d different powers of s and ℓ powers of t and KEA assumption for input $(h,\widehat{h}:=h^{\alpha})\in\mathbb{G}^2$. It takes the two basis $(g,\widehat{g}:=g^{\alpha}),(h,\widehat{h}:=h^{\alpha})$ and all the powers $\{g^{s^it^j},\widehat{g}^{s^it^j}\}_{i,j=0}^{d,\ell}$ and claims that it is infeasible to create $c,\widehat{c}$ such that $\widehat{c}=c^{\alpha}$ without knowing $\delta,\{a_{ij}\}_{i,j=0}^{d,\ell}$, that satisfy $c=h^{\delta}\prod_{i,j=0}^{d,\ell}(g^{s^it^j})^{a_{ij}}$. More formally:

Assumption 7.2.1 ((d,ℓ)-BPKE)**.** *The* (d,ℓ)-BPKE *assumption holds relative to a bilinear group* $(p,\mathbb{G},\mathfrak{G},\mathbb{G}_T,e)$ *for the class* $\mathcal{Z}$ *of auxiliary input generators if, for every* $\mathsf{aux}\in\mathcal{Z}$ *and PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* $\mathcal{E}$ *such that, on the probability space* $\mathsf{gk}:=(q,\mathbb{G},\mathfrak{G},\mathbb{G}_T,\mathsf{e})\leftarrow\mathcal{G}(1^{\lambda})$*,* $\Sigma\leftarrow(g,\{g^{s^it^j}\}_{i,j=0}^{d,\ell},\{\widehat{g}^{s^it^j}\}_{i,j=0}^{d,\ell};(h,\widehat{h},h^s);(\mathfrak{g},\widehat{\mathfrak{g}},\mathfrak{g}^s))$*,* $\mathsf{aux}\leftarrow\mathcal{Z}(\mathsf{gk},\Sigma)$*,* $g,h\leftarrow_{\$}\mathbb{G}$*,* $\mathfrak{g}\leftarrow_{\$}\mathfrak{G}$*,* $\alpha,s,t\leftarrow_{\$}\mathbb{Z}_q$*,* $\widehat{g}:=g^{\alpha}$*,* $\widehat{h}:=h^{\alpha}$*, and* $\widehat{\mathfrak{g}}:=\mathfrak{g}^{\alpha}$*:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{q\text{-}PKE}}:=\Pr\left[\begin{array}{c}(c,\widehat{c};\delta,\{a_{ij}\}_{i,j=0}^{d,\ell})\leftarrow(\mathcal{A}\|\mathcal{E})(\mathsf{gk},\Sigma;\mathsf{aux})\\ \mathsf{e}(\widehat{c},\mathfrak{g})=\mathsf{e}(c,\mathfrak{g}^{\alpha})\ \wedge\ c\neq h^{\delta}\prod_{i,j=0}^{d,\ell}(g^{s^it^j})^{a_{ij}}\end{array}\right]=\mathsf{negl}(\lambda).$$

7.2.0.2 The d–Strong Diffie-Hellman Assumption (d-SDH).

The Strong Diffie-Hellman assumption [BB08] says that given $(g,g^s,\ldots,g^{s^d})$ it is infeasible to compute $y=g^{\frac{1}{s-r}}$ for a chosen $r\in\mathbb{Z}_q$. In our applications, a few group elements are given as input to the adversary:

Assumption 7.2.2 (d-SDH)**.** *The* d*–Strong Diffie-Hellman assumption holds relative to a bilinear group* $(q,\mathbb{G},\mathfrak{G},\mathbb{G}_T,\mathsf{e})$ *if for all* PPT *adversaries* $\mathcal{A}$ *we have, on the probability space* $\mathsf{gk}:=(q,\mathbb{G},\mathfrak{G},\mathbb{G}_T,\mathsf{e})\leftarrow\mathcal{G}(1^{\lambda})$*,* $\Sigma\leftarrow(\mathsf{gk},(g,g^s,\ldots g^{s^d});(\mathfrak{g},\mathfrak{g}^s))$*,* $g\leftarrow_{\$}\mathbb{G}$*,* $\mathfrak{g}\leftarrow_{\$}\mathfrak{G}$*, and* $s\leftarrow_{\$}\mathbb{Z}_q$*:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{d\text{-}sdh}}:=\Pr\left[\ (r,y)\leftarrow\mathcal{A}(\mathsf{gk},\Sigma)\ \wedge\ y=g^{\frac{1}{s-r}}\ \right]=\mathsf{negl}(\lambda).$$

An adaptation of the proof in Boneh and Boyen [BB08] shows that our variant of the d-SDH assumption holds in the generic bilinear group model.

7.3 SNARK for Bivariate Polynomial Evaluation

7.3.1 Knowledge Commitment for Bivariate Polynomials

The definition of a knowledge commitment scheme and the Pederson Commitment example can be find in Section 2.4.5. We briefly recall a commitment scheme functionality: a *non-interactive* commitment scheme allows a sender to create a commitment to a secret value and later open the commitment and reveal the value in a verifiable manner. A commitment should be hiding and binding in the sense that a commitment does not reveal the secret value and cannot be opened to two different values.

We now present a variant of the Pedersen commitment scheme introduced by [Gro10], in order to build a commitment scheme specialized for (bivariate) polynomials $P \in \mathbb{Z}_q[X, Y]$, we call $\mathsf{Biv}.\mathcal{P}olyCom$, in the crs model: the message space M_{ck} is defined by polynomials in $\mathbb{Z}_q[X, Y]$ of degree in X bounded by a value d and degree in Y bounded by some value ℓ.

Based on an efficient construction of a polynomial commitment scheme proposed by [KZG10] we further construct a knowledge commitment scheme for bivariate polynomials that allows us to use it in a Commit-and-Prove SNARK protocol, for polynomial partial evaluation.

The commitment scheme $\mathsf{Biv}.\mathcal{P}olyCom = (\mathsf{Biv.ComGen}, \mathsf{Biv.Com}, \mathsf{Biv.ComVer}, \mathsf{Biv.OpenVer})$ consists in 4 algorithms and it is perfectly hiding and computationally biding. We call $\mathsf{Biv}.\mathcal{P}olyCom$ a knowledge commitment, since the prover cannot make a valid commitment without "knowing" the committed values. We will rely on the (d, ℓ)-BPKE assumption for extracting the committed values.

$\mathsf{Biv.ComGen}(1^\lambda, d, \ell) \to \mathsf{ck}$**:** Given some bounds d, ℓ on the degrees in X and in Y of the polynomials $P \in \mathbb{Z}_q[X, Y]$ to be commited, it generates the necessary key for producing commitments:

- Run $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$,
- Sample $g, h \leftarrow_\$ \mathbb{G}, \mathfrak{g} \leftarrow_\$ \mathfrak{G}$ and $\alpha, s, t \leftarrow_\$ \mathbb{Z}_q$,
- Define $\widehat{g} := g^\alpha, \widehat{h} := h^\alpha, \widehat{\mathfrak{g}} := \mathfrak{g}^\alpha$,
- Compute $\{g^{s^i t^j}\}_{i,j=0}^{d,\ell}, \{\widehat{g}^{s^i t^j}\}_{i,j=0}^{d,\ell}$,
- For simplicity, use notation $g_{ij} := g^{s^i t^j}$, $\widehat{g}_{ij} := \widehat{g}^{s^i t^j}$,
- Output a commitment public key:

$$\mathsf{ck} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}; (h, \widehat{h}); (\mathfrak{g}, \widehat{\mathfrak{g}})\}.$$

$\mathsf{Biv.Com}(\mathsf{ck}, P) \to (C, \rho)$**:** Given a bivariate polynomial $P = \sum_{i,j=0}^{d,\ell} a_{ij} X^i Y^j$, the committer picks a randomness $\rho \leftarrow_\$ \mathbb{Z}_q$ and computes the commitment $C = (c, \widehat{c})$ as defined below and sets the opening value $o := \rho$

$$c = h^\rho \prod_{i=0,j=0}^{d,\ell} g_{ij}^{a_{ij}} \qquad \widehat{c} = \widehat{h}^\rho \prod_{i=0,j=0}^{d,\ell} \widehat{g}_{ij}^{a_{ij}}.$$

$\mathsf{Biv.ComVer}(\mathsf{ck}, C = (c, \widehat{c})) \to 0/1$**:** Verifies whether $\widehat{c} = c^\alpha$ by checking whether $\mathsf{e}(c, \widehat{\mathfrak{g}}) = \mathsf{e}(\widehat{c}, \mathfrak{g})$.

$\mathsf{Biv.OpenVer}(\mathsf{ck}, C, P, \rho) \to P$**:** Parses $C := (c, \widehat{c})$ and $P = \sum_{i,j=0}^{d,\ell} a_{ij} X^i Y^j$, and checks whether both $\mathsf{ComVer}(\mathsf{ck}, C = (c, \widehat{c})) = 1$ and if $c = h^\rho \prod_{i,j=0}^{d,\ell} (g_{ij})^{a_{ij}}$. It outputs 1 if these checks hold. Else it outputs 0.

We can state the following theorem, whose proof can be found in the supplementary material.

Theorem 7.3.1. *The commitment scheme* $\mathsf{Biv}.\mathcal{P}olyCom$ *is perfectly hiding and computationally binding assuming the* q-SDH *assumption holds in* $\mathbb{G}$*. Moreover, assuming* (d, ℓ)-BPKE*, the scheme is knowledge binding.*

Now that we have built a compact commitment scheme for bivariate polynomials, we show how to prove certain relations about such polynomials.

7.3.2 Relations for Bivariate Polynomial Partial Evaluation

We want to construct a succinct non-interactive zero-knowledge argument system for some relation $\mathcal{R}$ of partial evaluation of bivariate polynomials:

$$\mathcal{R} := \{(u = (P(X, Y), k); w = Q(Y)) : Q(Y) = P(k, Y)\}.$$

Following ideas from [Lip16], we define Commit-and-Prove (CaP) argument systems for for bivariate polynomials partial evaluation. Intuitively, a Commit-and-Prove succinct non-interactive zero knowledge argument system for $\mathcal{R}$ uses a commitment scheme to compactly commit to some values $P \in \mathbb{Z}_q[X, Y], Q \in \mathbb{Z}_q[Y]$ as C, C', and then prove that $((P, k); Q) \in \mathcal{R}$ for a value $k \in \mathbb{Z}_q$. Note that, while Q is a uni-variate polynomial in Y, it can also be seen as a bivariate polynomial.

Our CaP argument system will exploit our computationally binding commitment scheme $\mathsf{Biv}.\mathcal{P}olyCom$ for bivariate polynomials. However, without their openings, commitments (C, C') themselves do not mean anything since they can be a commitment to any value (from the perfect hiding property). Rather, we define a new (compact) relation indexed by the commitment public key ck:

$$\begin{aligned} \mathcal{R}_{\mathsf{ck}} := \{(u = (C, C', k); w = (P, Q, \rho, \rho')) : \\ (C, \rho) = \mathsf{Biv.Com}(P) \ \wedge \ (C', \rho') = \mathsf{Biv.Com}(Q) \ \wedge \ ((P, k), Q) \in \mathcal{R}\}. \end{aligned} \tag{7.1}$$

The CaP-$\mathsf{P.SNARK}$ we construct uses an ($\mathcal{R}$-independent) bivariate polynomial commitment scheme $\mathsf{Biv}.\mathcal{P}olyCom$, follows the syntax below, and has to satisfy *completeness*, *succinctness*, *zero-knowledge* and *knowledge-soundness*, as usual.

$\mathsf{P.Gen}(1^\lambda, \mathcal{R}) \to \mathsf{crs}$**:** On input a security parameter $\lambda \in \mathbb{N}$ and an NP relation for partial polynomial evaluation $\mathcal{R} := \{(u = (P(X, Y), k); w = Q(Y)) : Q(Y) = P(k, Y)\}$, the generation algorithm runs $\mathsf{ck} \leftarrow \mathsf{Biv.ComGen}(1^\lambda)$ and defines a new relation R_{ck} depending on the commitment scheme as in Equation (7.1). a the common reference string crs that enables proving on $\mathcal{R}_{\mathsf{ck}}$.

$\mathsf{P.Prove}(\mathsf{crs}, x = (C, C', k), w = (P, Q, \rho, \rho')) \to \pi$**:** Given the crs, two commitments C, C' to the corresponding polynomials $P \in \mathbb{Z}_q[X, Y]$ and $Q \in \mathbb{Z}_q[Y]$, their randomness ρ, ρ' and a point k, this algorithm produces a proof π that $P(k, Y) = Q(Y)$, $(C, \rho) = \mathsf{Biv.Com}(P)$, and $(C', \rho') = \mathsf{Biv.Com}(Q)$.

$\mathsf{P.Ver}(\mathsf{crs}, u, \pi) \to b$**:** Parse $u = (C, C', k)$. On input crs, and a proof π, the verifier algorithm outputs $b = 0$ (reject) or $b = 1$ (accept).

7.3.3 SNARK for Bivariate Polynomial (Partial) Evaluation

We aim to build an efficient and compact $\mathsf{CaP}-\mathsf{P.SNARK}$ dedicated to partial evaluation for bivariate polynomials $P \in \mathbb{Z}_q[X,Y]$ in some random point $k \in \mathbb{Z}_q$.

Our scheme is based on an algebraic property of polynomials. We remark that $(X-k)$ perfectly divides the polynomial $P(X,Y)-P(k,Y)$ for $k \in \mathbb{Z}_q$. Under the bivariate power knowledge of exponent assumption ((d,ℓ)-BPKE), this $\mathsf{CaP}-\mathsf{P.SNARK}$ scheme is knowledge-sound.

$\underline{\mathsf{P.Gen}(1^\lambda,\mathcal{R}) \to \mathsf{crs}}$

1: $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda),\ g,h \leftarrow_\$ \mathbb{G}, \mathfrak{g} \leftarrow_\$ \mathfrak{G},\ \alpha,s,t \leftarrow_\$ \mathbb{Z}_q$
2: $\widehat{g} := g^\alpha, \widehat{h} := h^\alpha, \widehat{\mathfrak{g}} := \mathfrak{g}^\alpha$
3: $g_{ij} := g^{s^i t^j},\ \ \widehat{g}_{ij} := \widehat{g}^{s^i t^j}\ \forall\ i<d, j<\ell$
4: $\mathfrak{g}_1 := \mathfrak{g}^s, h_1 := h^s$
5: return crs as in Equation (7.2): $\mathsf{crs} = \{\mathsf{ck}, (h_1,\mathfrak{g}_1), \mathsf{RO}(\cdot,\cdot,\cdot)\}$

$\underline{\mathsf{P.Prove}(\mathsf{crs},u,w)}$

1: $(C,C',k) := u, (P,Q,\rho,\rho') := w$
2: $W := (P-Q)/(X-k)$
3: $(D,\omega) \leftarrow \mathsf{Biv.Com}(W)$
4: $\tilde{g} := h_1/h^k,\ \ x,y \leftarrow_\$ \mathbb{Z}_q$
5: $\mathbf{U} := \mathsf{e}(h^x\tilde{g}^y, \mathfrak{g})$
6: $e \leftarrow \mathsf{RO}(u,D,\mathbf{U})$
7: $\sigma = x-(\rho'-\rho)e \mod q$
8: $\tau = y-\omega e \mod q$
9: return $\pi := (D,e,\sigma,\tau)$

$\underline{\mathsf{P.Ver}(\mathsf{crs},u,\pi) \to b}$

1: $(C,C',k) := u, (D,e,\sigma,\tau) := \pi$
2: $(c,\widehat{c}) := C, (c',\widehat{c}') := C', (d,\widehat{d}) := D$
3: $b_1 \leftarrow \mathsf{Biv.ComVer}(C)$
4: $b_2 \leftarrow \mathsf{Biv.ComVer}(C')$
5: $b_3 \leftarrow \mathsf{Biv.ComVer}(D)$
6: $\mathbf{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$
7: $\mathbf{U} := \mathsf{e}(h^\sigma\tilde{g}^\tau, \mathfrak{g})\mathbf{A}^e$
8: $b_4 \leftarrow (e = \mathsf{RO}(u,D,\mathbf{U}))$
9: return $(b_1 \ \wedge\ b_2 \ \wedge\ b_3 \ \wedge\ b_4)$

Figure 7.1: Our $\mathsf{CaP}-\mathsf{P.SNARK}$ for bivariate polynomial partial evaluation

Description of Our $\mathsf{CaP}-\mathsf{P.SNARK}$ Protocol. Let $\mathsf{Biv}.\mathcal{P}olyCom$ be a bi-variate polynomial knowledge commitment scheme. We construct a zero-knowledge SNARK scheme for any relation $\mathcal{R}$ with respect to some bounds d,ℓ on the degrees in X and in Y of the polynomials $P \in \mathbb{Z}_q[X,Y]$ supported by $\mathsf{Biv}.\mathcal{P}olyCom$. Our protocol is formally depicted in Figure 7.1.

CRS generation. The setup algorithm $\mathsf{P.Gen}(1^\lambda,\mathcal{R})$, given a security parameter $\lambda \in \mathbb{N}$ and an NP relation $\mathcal{R}$ including the bound parameters d,ℓ, outputs a crs enabling the proof and verification of statements for the associated relation $\mathcal{R}_{\mathsf{ck}}$ defined and explained in Section 7.3.2.

We remark that steps 1 to 3 in the $\mathsf{P.Gen}$ algorithm from Figure 7.1 are the same as the generation of a commitment key ck for the $\mathsf{Biv}.\mathcal{P}olyCom$ scheme, as by running $\mathsf{ck} \leftarrow \mathsf{Biv.ComGen}(1^\lambda,d,\ell)$, *i.e.*, $\mathsf{ck} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}; (h,\widehat{h}); (\mathfrak{g},\widehat{\mathfrak{g}})\}$. The algorithm computes two extra values (step 4) that are not part of ck: $\mathfrak{g}_1 := \mathfrak{g}^s, h_1 := h^s$. Starting from this commitment key ck we can define the R_{ck} from $\mathcal{R}$ as discussed before.

Remark 7.3.2. *We stress that the generation of the* ck *cannot be done black-box, because of the need of generating together the values* $\mathfrak{g}_1, h_1$.

$$\mathsf{crs} : \left\{\mathsf{ck} = \left(\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}, (h, \widehat{h}), (\mathfrak{g}, \widehat{\mathfrak{g}})\right); (h_1, \mathfrak{g}_1); \mathsf{RO}(\cdot,\cdot,\cdot)\right\} \tag{7.2}$$

Prover. Given crs, the statement $u = (C, C', k)$ and the witness $w = (P, Q, \rho, \rho')$, the prover proceeds to compute a proof π in two steps:

Step 1. (From 1 to 3 in the $\mathsf{P.Prove}$ algorithm from Figure 7.1.) The prover computes a witness to the correct (partial) evaluation in $k \in \mathbb{Z}_q$ of the polynomial $P \in \mathbb{Z}_q[X, Y]$ as $P(k, Y) = Q \in \mathbb{Z}_q[Y]$. The witness of this evaluation is a polynomial $W \in \mathbb{Z}_q[X, Y]$ defined as the quotient $W := \frac{P(X,Y)-Q(Y)}{X-k}$. This is a well-defined polynomial in $\mathbb{Z}_q[X, Y]$ if and only if $P(k, Y) = Q \in \mathbb{Z}_q[Y]$. The element of the proof π that enables checking this algebraic property over the polynomials P and Q will be a commitment $(D = (d, \widehat{d}), \omega)$ to the polynomial W, where $\omega \leftarrow\!\!\$\, \mathbb{Z}_q$ is a fresh randomness.

Remark 7.3.3. *To this point, the verifier should be convinced that the polynomial Q is the good evaluation in k of P, only by checking the corresponding polynomial equation evaluated in a random hidden point $(s, t) : W(s, t)(s - k) = P(k, t) - Q(t)$, which can be reduced to the (d, ℓ)-BPKE assumption. This can be translated in terms of commitments $(C, \rho)(C', \rho'), (D, \omega)$ to P, Q, W as a pairing check: $\mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^{(s-t)\omega-(\rho-\rho')}, \mathfrak{g})$ where $C = (c, \widehat{c}), C' = (c', \widehat{c}'), D = (d, \widehat{d})$.*

Because of the hiding property, the verifier does not have access to the openings of the commitments, as it does not know the randomness ρ, ρ', ω.

We therefore need the prover to provide something more together with the commitment D to the witness $W \in \mathbb{Z}_q[X, Y]$ of the evaluation. The prover needs to compute an extra proof of knowledge of the randomnesses ω used to create this comitment and of the correct relation to satisfy with respect to the randomness ρ, ρ' of the statement commitments C, C' such that the pairing expression cancels the respective terms $h^{(\rho-\rho')}$ and $h^{(s-t)\omega}$.

This is easily solved by building a Schnoor proof of knowledge of the exponents $\omega, (\rho' - \rho)$ that appear in $\mathbf{A} = \mathsf{e}(h^{(s-k)\omega-(\rho-\rho')}, \mathfrak{g}) = \mathsf{e}(h^{(\rho'-\rho)}h^{(s-k)\omega}, \mathfrak{g})$. If we define $\tilde{g} := h_1/h^k = h^{s-k}$, then this proof is a classical Schnorr proof for the public value $\mathbf{A} = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g}) = \mathsf{e}(h, \mathfrak{g})^{\rho'-\rho} \cdot \mathsf{e}(\tilde{g}, \mathfrak{g})^{\omega}$ in the target group $\mathfrak{G}$. But we will show we can make it more efficient.

Step 2. (From 4 to 7 in the $\mathsf{P.Prove}$ algorithm from Figure 7.1.) This step consists in this non-interactive Schnorr proof associated to the value $\mathbf{A} = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g})$:

- Choose $x, y \in \mathbb{Z}_q$,
- Define $\mathbf{U} = \mathsf{e}(h^x \tilde{g}^y, \mathfrak{g})$, this corresponds to the first round in the interactive Schnorr proof protocol, where the prover sends its commitment.
- Sample the challenge to the Schnorr proof by running the random oracle (hash function) on input the statement to be proven and the commitment $\mathbf{U}$: $e \leftarrow \mathsf{RO}(u, D, \mathbf{U})$,
- Compute the answers $\sigma = x - (\rho' - \rho)e \bmod q$ and $\tau = y - \omega e \bmod q$.

The values sent as Schnorr proof are three scalars e, σ, τ, where e is the output of the hash function $\mathsf{RO}(u, D, \mathbf{U})$ and does not depend on the size of $\mathbf{U} \in \mathbb{G}_T$. After the two described steps, the prover algorithm outputs $\pi := (D, e, \sigma, \tau)$.

Verifier. First, the verifier parses the received statement and proof (steps 1 and 2 in the P.Ver algorithm from Figure 7.1), then it makes sure the commitments C, C', D are well-formed (steps 3 to 5 in the P.Ver algorithm from Figure 7.1) by running the Biv.ComVer algorithm. If this is not the case, we discard the proof π. To verify the proof π, one needs the polynomial equation $W(X,Y)(X-k) = P(k,Y) - Q(Y)$ to hold for some secret evaluation points (s,t). We can rewrite this equation in terms of pairings applied to the commitments (C, C', D): $\mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$. If the polynomials W, P, Q evaluated in the secret points s, t satisfy the equation $W(s,t)(s-k) = P(k,t) - Q(t)$, then all the exponents in base g cancel out in the pairing expression. It is not the case for the exponents in base h which correspond to the randomness used in the commitments. The important remark is that if D is correct, the remaining value $\mathbf{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$ can be written only in terms of the 3 randomness ρ, ρ', ω used to commit to P, Q, W:

$$\mathbf{A} = \mathsf{e}(h^{(s-k)\omega} h^{(\rho'-\rho)}, \mathfrak{g}) = \mathsf{e}(h^{\rho'-\rho}\tilde{g}^{\omega}, \mathfrak{g}).$$

This can be checked by the usual verification procedure of the Schnorr proof transmitted in π, *i.e.*, the values (e, σ, τ): Compute $\mathbf{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1}$ and $\mathbf{U} = \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathbf{A}^e$ then run the RO function to check whether $e = \mathsf{RO}(u, D, \mathbf{U})$.

7.4 Security Analysis of our CaP − P.SNARK

In this section, we prove the main result of this chapter:

Theorem 7.4.1. *Assuming both the* q-SDH *and* DLog *assumptions hold in the bilinear group* gk*, the protocol* CaP − P.SNARK *is a zero-knowledge Succinct Non-Interactive Argument of Knowledge in the random oracle model.*

In what follows we prove correctness and knowledge soundness. We defer the reader to the supplementary material for the proof of zero-knowledge.

7.4.1 Correctness

To prove the correctness of our protocol we show that if the commitments C, C', D are honestly generated, the checks the Ver algorithm does all pass through. We consider $C = h^{\rho} g^{P(s,t)}, C' = h^{\rho'} g^{p}, D = h^{\omega} g^{W(s)}$, where $Q = P(k,Y)$ and $W(X,Y) = \frac{P(X,Y)-P(k,Y)}{(x-k)}$. We have that $\mathbf{U} \coloneqq \mathsf{e}(h^x \tilde{g}^y, \mathfrak{g})$ and the verifier computes $\mathbf{U}$ from u and π as follows:

$$\begin{aligned}
\mathbf{U} &= \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathbf{A}^e = \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k)^e \cdot \mathsf{e}(c/c', \mathfrak{g})^{-e} \\
&= \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathsf{e}(h^{\omega} g^{W(s)}, \mathfrak{g}^{s-k})^e \cdot \mathsf{e}(h^{\rho-\rho'} g^{P(s,t)-p}, \mathfrak{g})^{-e} \\
&= \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathsf{e}(h, \mathfrak{g})^{e(s-k)\omega} \cdot \mathsf{e}(g, \mathfrak{g})^{e(s-k)W(s)} \cdot \mathsf{e}(h, \mathfrak{g})^{e(\rho'-\rho)} \cdot \mathsf{e}(g, \mathfrak{g})^{-e(P(s,t)-p)} \\
&= \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathsf{e}(h, \mathfrak{g})^{e(s-k)\omega+e(\rho'-\rho)} = \mathsf{e}(h^{\sigma}\tilde{g}^{\tau}, \mathfrak{g}) \cdot \mathsf{e}(h^{e(\rho'-\rho)}\tilde{g}^{e\omega}, \mathfrak{g}) = \mathsf{e}(h^x \tilde{g}^y, \mathfrak{g}).
\end{aligned}$$

7.4.2 Soundness

Before going into the technical details of the proof, we provide some intuition about its strategy. The polynomial commitment scheme Biv.$\mathcal{P}olyCom$ requires the prover Prove to exhibit two values $(c, \widehat{c})$, that are the same encoding of coefficients of a polynomial $P(X,Y)$ in the exponent, but with respect to different bases. The reason that we require the prover

to duplicate its effort w.r.t. α is so that the simulator in the security proof can extract representations of $(c, \widehat{c})$ as a polynomial $P(X, Y)$, under the q-PKE assumption.

Suppose an adversary $\mathcal{A}$ manages to forge a SNARK of a false statement that nonetheless passes the verification test. The intuition behind the proof is to use the adversary $\mathcal{A}$ and the fact that the commitment scheme $\mathsf{Biv}.\mathcal{P}olyCom$ is extractable to be able to solve the q-SDH assumption for $d = \deg(P)$ in X. There is a similar complementary case that allows this adversary to solve the q-SDH assumption for $d = \deg(P)$ in Y (actually ℓ in our notations).

We first need two preliminary lemmas.

Lemma 7.4.2 (Global Extractor)**.** *Assume that* $\mathsf{Biv}.\mathcal{P}olyCom$ *is an extractable commitment scheme with perfect hiding and computational binding and that* BPKE *class of assumptions holds in the bilinear group* gk*. For any* PPT *adversary* $\mathcal{A}^{\mathsf{KS}}$ *agains the knowledge soundness of* $\mathsf{CaP} - \mathsf{P.SNARK}$ *that has non-negligeable probability of success in breaking the scheme, there exists an extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c|r} C = \mathsf{Biv.Com}(P, \rho) & \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}), z \leftarrow \mathcal{Z}(\mathsf{crs}) \\ \wedge\ C' = \mathsf{Biv.Com}(Q, \rho') & ((u, \pi); \mathsf{wit}) \leftarrow (\mathcal{A}^{\mathsf{KS}} \| \mathcal{E}^*)(\mathsf{crs}, z) \\ \wedge\ D = \mathsf{Biv.Com}(W, \omega) & u := (C, C', k),\ \pi := (D, \mathbf{U}, \sigma, \tau) \\ & \mathsf{wit} := (P, \rho, Q, \rho', W, \omega) \\ & \mathsf{Ver}(\mathsf{crs}, u, \pi) = 1 \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

Proof. We show the existence of an extractor $\mathcal{E}^*$ that will output the polynomials $P(X, Y)$, $Q(Y), W^*(X, Y)$ and the randomness ρ, ρ', ω corresponding to the commitments C, C', D, with overwhelming probability.

Let $\mathcal{A}^{\mathsf{KS}}$ be an adversary that breaks the KS of the protocol $\mathsf{CaP}-\mathsf{P.SNARK}$ with overwhelming probability, meaning it outputs a false proof that passes the verifier checks. Consider now the adversary $\mathcal{B}^{\mathsf{BPKE}}$ that takes as input $\sigma \leftarrow (g, \{g^{s^i t^j}\}_{i,j=0}^{d,\ell}, \{\widehat{g}^{s^i t^j}\}_{i,j=0}^{d,\ell}; (h, \widehat{h}, h^s); (\mathfrak{g}, \mathfrak{g}^\alpha, \mathfrak{g}^s))$ and runs the adversary $\mathcal{A}^{\mathsf{KS}}$ against the scheme. $\mathcal{B}^{\mathsf{BPKE}}$ can provide a valid CRS to $\mathcal{A}^{\mathsf{KS}}$ by using its inputs:

$$\mathsf{crs} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}; (h, \widehat{h}, h_1); (\mathfrak{g}, \mathfrak{g}^\alpha, \mathfrak{g}_1)\}.$$

The statement u, corresponding to $\pi \leftarrow \mathcal{A}^{\mathsf{KS}}(\mathsf{crs})$, contains the values $C := (c, \widehat{c}), C' := (c', \widehat{c}')$ that verify $\mathsf{e}(c, \widehat{\mathfrak{g}}) = \mathsf{e}(\widehat{c}, \mathfrak{g})$ and $\mathsf{e}(c', \widehat{\mathfrak{g}}) = \mathsf{e}(\widehat{c}', \mathfrak{g})$. The same holds for the value D provided in the proof $\pi = (D, e, \sigma, \tau)$, i.e. $\mathsf{e}(d, \widehat{\mathfrak{g}}) = \mathsf{e}(\widehat{d}, \mathfrak{g})$.

Provided that for any adversary $\mathcal{B}^{\mathsf{BPKE}}$ that outputs valid commitment pair $(c, \widehat{c})$, there exists an extractor that returns the corresponding witness (the opening). We run the extractor $\mathcal{E}_{\mathcal{B}}$ associated to $\mathcal{B}^{\mathsf{BPKE}}$ for each of the inputs $C = (c, \widehat{c}), C' = (c', \widehat{c}'), D = (d, \widehat{d})$. This returns the description of polynomials $P(X, Y), Q(Y), W^*(X, Y)$ and some scalars ρ, ρ', ω. Note that the existence and efficacy of $\mathcal{E}_{\mathcal{B}}$ is guaranteed by the (d, ℓ)-BPKE assumption. We will then define a *general* extractor $\mathcal{E}^*$ associated to the adversary $\mathcal{A}^{\mathsf{KS}}$ by running $\mathcal{E}_{\mathcal{B}}$ on the same input. We call this global algorithm composed of the adversary $\mathcal{A}^{\mathsf{KS}}$ and the general extractor $\mathcal{E}^*$, machine $M := \mathcal{A}^{\mathsf{KS}} || \mathcal{E}^*$. □

Lemma 7.4.3 (Extended Adversary Machine)**.** *Assume that* BPKE *class of assumptions holds in the bilinear group* gk *and that Schnorr proof used in the* $\mathsf{CaP} - \mathsf{P.SNARK}$ *protocol is sound. For any* PPT *adversary* $\mathcal{A}^{\mathsf{KS}}$ *against the knowledge soundness of the scheme* $\mathsf{CaP} - \mathsf{P.SNARK}$ *that outputs* $u = (C, C', k), \pi = (D, e, \sigma, \tau)$*, where* C, C', D *are well-formed commitments*

under Biv.$\mathcal{P}olyCom$ *and the proof* π *verifies, i.e.,* $\mathsf{Ver}(\mathsf{crs}, u, \pi)$*, there exists a machine,* extended adversary $\mathcal{A}^*$ *that outputs the same as* $\mathcal{A}^{\mathsf{KS}}$ *together with an extended witness* $\mathsf{wit} = (P, \rho, Q, \rho', W, \omega, \delta, \gamma)$*, where* $P, W \in \mathbb{Z}_q[X, Y], Q \in \mathbb{Z}_q[Y]$ *are the openings of the commitments* (C, C', D) *under randomness* ρ, ρ', ω *and* δ, γ *are such that* $\mathbf{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$.

Proof. We use the previously defined machine M from Lemma 7.4.2 and the rewinding technique [PS00] for proving the soundness of the Schnorr's proof to extract the scalars δ, γ such that $\mathbf{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$: Consider the game between the challenger and the machine M against the soundness of the Schnorr's proof. The challenger runs M by fixing the values (C, C', D) and changing the oracle definition to get a fork with $e' \leftarrow \mathsf{RO}(U, D, \mathbf{U}) \neq e$. The forger M will output two distinct forgeries corresponding to the same random oracle query, but for two distinct answers of the random oracle, e and e'. The Forking Lemma shows that by rewinding the adversary $\mathcal{O}(q_h/\varepsilon)$ times, where q_h is the maximal number of random oracle queries of the machine M and ε its success probability, then one finds two such forgeries $(\sigma, \tau), (\sigma', \tau')$ with constant probability, which enables to compute the values δ, γ such that $\mathbf{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$.

Using the the existence of $\mathcal{E}^*$ extractor and of the algorithm that rewinds the machine M in order to obtain the output δ, γ as described before, we can define an aggregate machine $\mathcal{A}^*$ corresponding to the concatenation of both. This machine $\mathcal{A}^*$ takes the same input as $\mathcal{A}^{\mathsf{KS}}$ and outputs the witness corresponding to the commitment openings $(P, \rho), (Q, \rho'), (W, \omega)$ and two scalars δ, γ satisfying $\mathbf{A} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$. □

We now have all the tools to prove the soundness:

Proof. This proof is in two steps, with two distinct cases in the second step.

Step 1. First we show that for every PPT adversary $\mathcal{A}^{\mathsf{KS}}$ against the soundness of the protocol, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ that runs on the same input and random coins as $\mathcal{A}^{\mathsf{KS}}$ and outputs a witness. Defining the extractor $\mathcal{E}_{\mathcal{A}}$ is straightforward from the Lemma 7.4.2 by running the $\mathcal{E}^*$ and keeping just the values (P, ρ, Q, ρ') from its output.

Assuming the existence of an adversary $\mathcal{A}^{\mathsf{KS}}$ and extractor $\mathcal{E}_{\mathcal{A}}$ that has a non-negligible success probability in winning the soundness game against the protocol CaP – P.SNARK, we now show that we can either solve the discrete logarithm problem, or break the q-SDH assumption.

Step 2. Suppose the machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined in the Lemma 7.4.3 is able to output a cheating pair statement-proof $u = (C, C', k), \pi = (D, e, \sigma, \tau)$ and a witness $\mathsf{wit} = (\rho, \rho', \omega, P, Q^*, W^*, (\delta, \gamma))$ such that it passes verification checks, but the extracted values $P \in \mathbb{Z}_q[X, Y], Q^* \in \mathbb{Z}_q[Y]$ are not satisfying the expected relation $Q^*(Y) = P(k, Y)$.

For simplicity, we will call $\Delta = \rho' - \rho$. Assuming that the commitment scheme is binding, then one of the following scenarios must hold:

1. The polynomials extracted do not satisfy the correct relation not even when evaluated in s: $W^*(s, t) \neq \frac{P(s,t) - Q^*(t)}{s-k}$. This type of forgery can be reduced to the DLog problem for $(g, h) \in \mathbb{G}$, in the case 1 below (see Lemma 7.4.4);

2. The polynomial $W^* \in \mathbb{Z}_q[X, Y]$ committed in D does not satisfy the correct relation with respect to the other extracted values P, Q^*, but still evaluated in s, t we have that

$W^*(s,t) = \frac{P(s,t)-Q^*(t)}{s-k}$. We reduce the case to the q-SDH assumption, in the case 2 below (see Lemma 7.4.5).

□

Lemma 7.4.4 (Case 1)**.** *Consider the adversarial machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined by the Lemma 7.4.3 that outputs some values $u = (k, C, C', D, e, \sigma, \tau)$ and $(\rho, P, \rho', Q^*, \omega, W^*, \delta, \gamma)$, such that $P(k,Y) \neq Q^*(Y)$, where $P, W^* \in \mathbb{Z}_q[X,Y]$, $Q^* \in \mathbb{Z}_q[Y]$ and $(P,\rho), (Q^*,\rho'), (W^*,\omega)$ are the openings of the commitments (C, C', D) and (δ,γ) satisfy $\mathbf{A} \coloneqq \mathsf{e}(h^{\omega} g^{W^*(s,t)}, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(h^{-\Delta} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$. Given that the verification check outputs 1 for π, there is a negligible probability that the values k, P, Q^*, W^* are such that $W^*(s,t) \neq \frac{P(s,t)-Q^*(t)}{(s-k)}$ under* DLog *assumption with respect to the group $\mathbb{G}$.*

Proof. Let $\mathcal{B}^{\mathsf{DLog}}$ be an adversary that gets the challenge $(g,h) \in \mathbb{G}$ and simulates the crs to $\mathcal{A}^*$ by picking $\alpha, s \in \mathbb{Z}_p$ and computing the missing elements. We define $\tilde{g} \coloneqq h^{s-k}$, and we denote δ, γ, the two outputs of $\mathcal{A}^*$ such that $\mathbf{A} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$. Assuming the binding of the commitment scheme, the check in the verification step of the scheme gives us: $\mathsf{e}(h^{(s-k)\omega} g^{(s-k)W^*(s,t)}, \mathfrak{g}) \cdot \mathsf{e}(h^{\rho-\rho'} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma})$. By the non-degeneracy of the pairing map, it must be that

$$h^{\delta+(s-k)\gamma-\Delta-(s-k)\omega} = g^{(s-k)W^*(s,t)-P(s,t)+Q^*(t)}.$$

Since $(s-k)W^*(s,t) - P(s,t) + Q^*(t) \neq 0$, we can extract the discrete logarithm of h in basis g. □

Lemma 7.4.5 (Case 2)**.** *Consider the adversarial machine $\mathcal{A}^*$ associated to $\mathcal{A}^{\mathsf{KS}}$ defined by the Lemma 7.4.3 that outputs some values $u = (k, C, C', D, e, \sigma, \tau)$ and $(\rho, P, \rho', Q^*, \omega, W^*, \delta, \gamma)$, such that $P(k,Y) \neq Q^*(Y)$, where $P, W^* \in \mathbb{Z}_q[X,Y]$, $Q^* \in \mathbb{Z}_q[Y]$ and $(P,\rho), (Q^*,\rho'), (W^*,\omega)$ are the openings of the commitments (C, C', D) and (δ,γ) satisfy $\mathbf{A} \coloneqq \mathsf{e}(h^{\omega} g^{W^*(s,t)}, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(h^{-\Delta} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$. Given that the verification check outputs 1 for π, there is a negligible probability that the values k, P, Q^*, W^* satisfy $W^*(s,t) = \frac{P(s,t)-Q^*(t)}{(s-k)}$ under q-SDH assumption with respect to the bilinear group gk, where $d' = \max\{d, \ell\}$.*

Proof. Consider the adversary $\mathcal{B}^{\mathsf{SDH}}$ against q-SDH assumption, having as auxiliary input $z = (k, \Delta, \omega, \delta, \gamma)$. Using its challenge $(g, g^s, \ldots, g^{s^d}, \widehat{g}^s, \ldots, \widehat{g}^{s^d}; h, h^s, \widehat{h}; (\mathfrak{g}, \mathfrak{g}^s))$ it picks random scalars $\alpha, t \leftarrow_{\$} \mathbb{Z}_q$ and computes $g_{ij} \coloneqq g^{s^i t^j}$, $\mathfrak{g}_{ij} \coloneqq \mathfrak{g}^{s^i t^j}$ for $i = 0, \ldots, d$ and $j = 0, \ldots, \ell$, $\widehat{\mathfrak{g}} \coloneqq \mathfrak{g}^{\alpha}$, and computes a commitment key $\mathsf{ck} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}; (h, \widehat{h}); (\mathfrak{g}, \widehat{\mathfrak{g}})\}$. It further sends the following crs corresponding to the relation $\mathcal{R}_{\mathsf{ck}}$ to $\mathcal{A}^*$: $\mathsf{crs} = \{\mathsf{ck}, (h_1, \mathfrak{g}_1), \mathsf{RO}(\cdot,\cdot,\cdot)\}$. Note that h_1 and $\mathfrak{g}_1$ are in the initial challenge.

From the output of the aggregated machine $\mathcal{A}^*$, $\mathcal{B}^{\mathsf{SDH}}$ gets the values $u = (k, C, C', D, e, \sigma, \tau)$ and $(\rho, P, \rho', Q^*, \omega, W^*, \delta, \gamma)$. The verification check of the Schnorr proof implies

$$\begin{aligned}\mathbf{A} &= \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^{\omega} g^{W^*(s,t)}, \mathfrak{g}^{s-k}) \cdot \mathsf{e}(h^{-\Delta} g^{P(s,t)-Q^*(t)}, \mathfrak{g})^{-1} \\ &= \mathsf{e}(h^{\Delta} h^{(s-k)\omega}, \mathfrak{g}) \cdot \mathsf{e}(g^{-(P(s,t)-Q^*(t))} g^{(s-k)W^*(s,t)}, \mathfrak{g}) = \mathsf{e}(h^{\Delta}\tilde{g}^{\omega}, \mathfrak{g}).\end{aligned}$$

The outputs δ, γ from the extended adversary machine are such that $\mathbf{A} = \mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g})$ (from the soundness of the Schnorr's proof). This, together with the previous equation leads to the conclusion $\mathsf{e}(h^{\delta}\tilde{g}^{\gamma}, \mathfrak{g}) = \mathsf{e}(h^{\Delta}\tilde{g}^{\omega}, \mathfrak{g})$ and so $\mathsf{e}(h^{\Delta-\delta}\tilde{g}^{\omega-\gamma}, \mathfrak{g}) = 1$. The adversary $\mathcal{B}^{\mathsf{SDH}}$ is able

to produce a solution to the equation $(s-k)(\omega-\gamma)+\Delta-\delta=0 \bmod q$ and though to find $s=k+\frac{\delta-\Delta}{\omega-\gamma} \bmod q$, unless $\gamma=\omega$ and $\delta=\Delta$.

In the former case, from the value s, one can easily break any SDH problem. In the latter case, the two writings for $\mathbf{A}$, and the non-degeneracy of the pairing map lead to $g^{(s-k)W^*(s,t)-P(s,t)+Q^*(t)}=1$.

Using $P \in \mathbb{Z}_q[X,Y]$, compute $Q(Y) \coloneqq P(k,Y)$ and define the polyomial $W=\frac{P(X,Y)-Q(Y)}{(X-k)} \in \mathbb{Z}_q[X,Y]$. We have also that $g^{(s-k)W(s,t)}=g^{P(s,t)-Q(t)}$, then $\mathcal{B}^{q\text{-}\mathsf{SDH}}$ computes $g^{Q(t)-Q^*(t)}=g^{(s-k)(W^*(s,t)-W(s,t))}$. Define the polynomial in $\mathbb{Z}_q[X,Y]: W'(X,Y)=(X-Y)(W(X,Y)-W^*(X,Y))-Q(Y)+Q^*(Y)$, we have that $g^{W'(s,t)}=1$. This splits in two cases:

1. If $W'(X,t)=\sum_i(\sum_j w'_{ij}t^j)X^i$ is the zero polynomial in X, meaning that all the coefficients $\tilde{w}_i=\sum_j w'_{ij}t^j$ are zero $\forall i \in \{0,\cdots,d\}$, then by choosing an index $i_0 \in \{0,\dots d\}$ with a non-zero element $w'_{i_0 j}$ (unless $W'=0$ which contradicts the hypothesis of the Lemma) we have that t is a root of the polynomial $\sum_j w'_{i_0 j}Y^j$.

2. If there exists at least one index $\tilde{\imath}$, such that $\tilde{w}_{\imath}=\sum_j w'_{\tilde{\imath}j}t^j \neq 0$, then we have that s is a root of the polynomial defined by fixing t: $W'(X,t)=\sum_i \tilde{w}_i X^i$.

If the first case happens with non-negligible probability, algorithm $\mathcal{B}_1^{\mathsf{SDH}}$ receives an $\ell-\mathsf{SDH}$ instance in t, and choosing s, can complete the input for the aggregate machine $\mathcal{A}^*$. Knowing that $\sum_j w'_{i_0 j}t^j=0$, the adversary $\mathcal{B}_1^{\mathsf{SDH}}$ is able to compute $g^{\frac{1}{t}}$ in the following way: Consider $w'_{i_0 j_0}$ the first non-zero coefficient of the polynomial $\sum_j w'_{i_0 j}Y^j$: $g^{w'_{i_0 j_0}t^{j_0}}=g^{t^{j_0+1}W''(t)}$ for some polynomial W'' of lower degree, and so $g^{w'_{i_0 j_0}}=g^{tW''(t)}$, or equivalently $g^{1/t}=(g^{W''(t)})^{1/w'_{i_0 j_0}}$. $\mathcal{B}_1^{\mathsf{SDH}}$ is then able to solve the $\ell-\mathsf{SDH}$ problem, as $g^{W''(t)}$ can be computed from the initial instance.

If the second case happens with non-negligible probability, the algorithm $\mathcal{B}_2^{\mathsf{SDH}}$ receives a $d-\mathsf{SDH}$ instance in s, and can complete it as an input for $\mathcal{A}^*$ by choosing t. Doing as above, with a polynomial that has s as a root, it can compute $g^{1/s}$. This solves the $\ell-\mathsf{SDH}$ instance. □

7.5 SNARK for Simultaneous Evaluations

From our $\mathsf{CaP}-$SNARK for the evaluation of *one* bivariate polynomial on a point k, we show how we can use it for the evaluation of *many* univariate polynomials on the same point k. This will be our mail tool for verifiable computation using FHE on Ring-LWE. More precisely, we show how to use the $\mathsf{Biv}.\mathcal{P}olyCom$ and our previous SNARK to define a commitment scheme and a compact proof system dedicated to multi-polynomials evaluation in the same random point k: given a single compact knowledge commitment C for a set of univariate polynomials $\{P_j(X)\}_j \in \mathbb{Z}_q[X]$ and a public evaluation point $k \in \mathbb{Z}_q$, we want to prove that some values $\{p_j\}_j$ committed in C' are indeed evaluations of the committed polynomials in this point k.

7.5.1 Commitment for Multiple Univariate Polynomials

We describe below, $\mathsf{Uni}.\mathcal{M}ultiCom$, our new knowledge commitment for a set of univariate polynomials. It is obtained in a straightforward way from $\mathsf{Biv}.\mathcal{P}olyCom$. It is defined as follows, where for simplicity $\ell+1$ is the number of committed univariate polynoimals:

$\mathsf{Uni.ComGen}(1^\lambda, d, \ell) \to \mathsf{ck}$**:** Given some degree bound d and some maximal bound $\ell+1$ on the cardinal of the polynomial set to be committed, it runs $\mathsf{ck} \leftarrow \mathsf{Biv.ComGen}(1^\lambda, d, \ell)$, where d, ℓ are the bounds on the degrees on X and Y of the bivariate polynomials in $\mathbb{Z}_q[X,Y]$.

$$\mathsf{ck} = \{\mathsf{gk}, (g_{ij})_{i,j=0}^{d,\ell}, (\widehat{g}_{ij})_{i,j=0}^{d,\ell}; (h, \widehat{h}); (\mathfrak{g}, \widehat{\mathfrak{g}})\};$$

$\mathsf{Uni.Com}(\mathsf{ck}, \{P_j\}_{0\le j\le \ell}) \to (C, \rho)$**:** Given a set of $\ell+1$ polynomials in $\mathbb{Z}_q[X]$, $\{P_j = \sum_{i=0}^{d} p_{ij} X^i\}_{0\le j\le \ell}$, we can define the bivariate polynomial $P = \sum_{i,j=0}^{d,\ell} p_{ij} X^i Y^j$ and run $(C, \rho) \leftarrow \mathsf{Biv.Com}(\mathsf{ck}, P)$;

$\mathsf{Uni.ComVer}(\mathsf{ck}, C = (c, \widehat{c})) \to 0/1$**:** Runs $b \leftarrow \mathsf{Biv.ComVer}(\mathsf{ck}, C = (c, \widehat{c}))$;

$\mathsf{Uni.OpenVer}(\mathsf{ck}, C, \{P_j\}_{0\le j\le \ell}, \rho) \to \{P_j\}_j$**:** Runs $P \leftarrow \mathsf{Biv.OpenVer}(\mathsf{ck}, C, P, \rho)$ where $P = \sum_{i,j=0}^{d,\ell} p_{ij} X^i Y^j$. If $P_j = \sum_{i=0}^{d} p_{ij} X^i$ for all $0 \le j \le \ell$, then output 1, else reject and output 0.

We state the following theorem. Its proof simply follows from the way we encode multiple polynomials into a bivariate one.

Theorem 7.5.1. *This commitment scheme* $\mathsf{Uni.}\mathcal{M}ultiCom$ *is perfectly hiding, computationally binding, and knowledge binding assuming the scheme* $\mathsf{Biv.}\mathcal{P}olyCom$ *also is so.*

Proof. Let us show this three properties from the initial $\mathsf{Biv.}\mathcal{P}olyCom$ commitment.

Perfect hiding: Since $C = (c, \widehat{c})$ in $\mathbb{G}^2$ is generated by $\mathsf{Biv.Com}$ algorithm, and $\mathsf{Biv.}\mathcal{P}olyCom$ is hiding, we have that $\mathsf{Uni.}\mathcal{M}ultiCom$ is perfectly hiding as well.

Computational binding: Assume that there exists a non-uniform probabilistic time adversary $\mathcal{A}$ that given a commitment $C = (c, \widehat{c})$ creates two valid openings $(\{A_j\}_j, \rho), (\{B_j\}_j, \tau)$, where all $A_j, B_j \in \mathbb{Z}_q[X]$. Then we create an adversary $\mathcal{B}$ against the binding of $\mathsf{Biv.}\mathcal{P}olyCom$ that runs $\mathcal{A}$ and outputs the pair of polynomials in $\mathbb{Z}_q[X,Y]$ as follows $A \coloneqq \sum_{j=0}^{\ell} A_j Y^j$ and $B \coloneqq \sum_{j=0}^{\ell} B_j Y^j$. We have that (A, ρ) and (B, ρ) are two valid openings for the commitment C. This breaks the binding property of $\mathsf{Biv.}\mathcal{P}olyCom$.

Knowledge binding: This property follows directly from the knowledge binding of $\mathsf{Biv.}\mathcal{P}olyCom$ scheme. □

7.5.2 Succinct Proof of Simultaneous Evaluations in a Point k

The construction of an efficient and compact $\mathsf{M.SNARK}$ dedicated to multiple univariate-polynomial evaluations in some common point k follows as well from the $\mathsf{P.SNARK}$ we built for partial evaluation. More precisely, for some parameters d, ℓ and some given knowledge commitments C, C' for polynomials of maximal degree d, $\{P_j\}_{0\le j\le \ell} \in \mathbb{Z}_q[X]$ and scalars $\{p_j\}_{0\le j\le \ell} \in \mathbb{Z}_q$ and a public evaluation point $k \in \mathbb{Z}_q$, we want to prove that p_j is the evaluation $P_j(k)$ for any $0 \le j \le \ell$.

7.5.2.1 Description of the $\mathsf{CaP-M.SNARK}$ Protocol.

We now describle our protocol for proving multiple univariate-polynomial evaluations in some common point k, where the j index is always considered as $0 \le j \le \ell$, and thus for $\ell+1$ polynomials:

$\mathsf{M.Gen}(1^\lambda, \mathcal{R}_m) \to \mathsf{crs}$**:** On input a security parameter $\lambda \in \mathbb{N}$ and a NP relation $\mathcal{R} := \{(u = (\{P_j\}_j, k); w = \{p_j\}) : P_j(k) = p_j\}$, define the associated relation $\mathcal{R} := \{(u = (P(X,Y), k); w = Q(Y)) : Q(Y) = P(k,Y)\}$ where $P(X,Y) := \sum_{j=0}^{\ell} P_j Y^j, Q(Y) := \sum_{j=0}^{\ell} p_j Y^j$. Output the common reference string by running $\mathsf{crs} \leftarrow \mathsf{P.Gen}(1^\lambda, \mathcal{R})$;

$\mathsf{M.Prove}(\mathsf{crs}, u = (C, C', k), w = (\{P_j\}_j, \{p_j\}_j, \rho, \rho')$**:** Given crs, the instance u and the witness w, the prover defines new bi-variate polynomials $P(X,Y) := \sum_{j=0}^{\ell} P_j Y^j, Q(y) := \sum_{j=0}^{\ell} p_j Y^j$ and compute the proof π for those: $\pi \leftarrow \mathsf{P.Prove}(\mathsf{crs}, u = (C, C', k), w = (P, Q, \rho, \rho')$. Output $\pi := (D, e, \sigma, \tau)$;

$\mathsf{M.Ver}(\mathsf{crs}, u, \pi) \to b$**:** Same algorithm as for partial-evaluation P.SNARK.

Remark 7.5.2. *oof can be seen as a commitment to a set of univariate polynomials $\{W_j\}_j$ using the* Uni.$\mathcal{M}ultiCom$ *as follows: Write $W_j = \sum_{i=0}^{d} w_{ij} X^i$, then running* $\mathsf{Uni.Com}(\mathsf{ck}, \{W_j\}_j)$ *gives the same output (D, ω) as running* $\mathsf{Biv.Com}(\mathsf{ck}, W)$.

Theorem 7.5.3. *Assuming the* P.SNARK *is a public coin argument of knowledge of openings of C and C' to some polynomials $P \in \mathbb{Z}_q[X,Y], Q \in \mathbb{Z}_q[Y]$ such that $P(k,Y) = Q(Y)$, then* M.SNARK *is a public coin argument of knowledge of openings of C and C' to a set of polynomials $\{P_j\}_j \in \mathbb{Z}_q[X]$ and a set of scalars $\{p_j\}_j \in \mathbb{Z}_q$ such that $P_j(k) = p_j \forall 0 \leq j < \ell$.*

Proof. We prove the required properties:

Correctness. Follows from the P.SNARK correctness.

Knowledge Soundness. We first build a knowledge-extractor. This knowledge extractor directly follows from Lemma 7.4.2 and an extended extractor can be defined as in Lemma 7.4.3. For any adversary $\mathcal{B}$ against P.SNARK, there exists an aggregated machine $\mathcal{B}^*$ that outputs the same as $\mathcal{B}$ together with a extended witness $\mathsf{wit} = (P, \rho, Q, \rho', W, \omega)$.

From the output of this extended machine $\mathcal{B}^*$ we can further extract $\{P_j := \sum_{i=0}^{d} p_{ij} X^i\}_j, \{p_j := q_j\}_j, W_j := \sum_{i=0}^{d} w_{ij} X^i\}_j$ just by reading the respective coefficients p_{ij}, q_j, w_{ij} from the bivariate polynomials $P = \sum_{i,j=0}^{d,\ell} p_{ij} X^i Y^j$, $Q = \sum_{j=0}^{\ell} q_j Y^j$, and $W = \sum_{i,j=0}^{d,\ell} w_{ij} X^i Y^j$.

Therefore, for any adversary $\mathcal{A}$ against the M.SNARK protocol, there exists an extended machine $\mathcal{A}^*$ that runs the aggregate machine $\mathcal{B}^*$ under its output and further returns the same statement and proof as $\mathcal{A}$ toghether with an extended witness $\mathsf{wit} = (\{P_j\}_j, \rho, \{p_j\}_j, \rho', \{W_j\}, \omega; \delta, \gamma)$, where $P_j, W_j \in \mathbb{Z}_q[X], p_j \in \mathbb{Z}_q$ are the openings of the commitments (C, C', D) under randomness ρ, ρ', ω and δ, γ are such that $\mathbf{A} = \mathsf{e}(d, \mathfrak{g}_1/\mathfrak{g}^k) \cdot \mathsf{e}(c/c', \mathfrak{g})^{-1} = \mathsf{e}(h^\delta \tilde{g}^\gamma, \mathfrak{g})$.

Soundness. We reduce the soundness of M.SNARK to the soundness of P.SNARK. Suppose there exists an adversary $\mathcal{A}$ against the soundness of M.SNARK, with the corresponding associated extended machine $\mathcal{A}^*$ that outputs a cheating proof π^* that passes the verification check with non-negligible probability. We then build an efficient adversary $\mathcal{B}$ against P.SNARK that runs the machine $\mathcal{A}^*$ to break the protocol with non-negligible probability.

$\mathcal{B}$ runs $\mathcal{A}^*$ that outputs the corresponding tuple proof-statement-witness $u = (C, C', k), \pi^* = (D, e, \sigma, \tau), \mathsf{wit} = (\{P_j\}_j, \rho, \{p_j\}_j, \rho', \{W_j\}, \omega; \delta, \gamma)$. Then, we can define some corresponding bivariate polynomials as follows and build an extractor for $\mathcal{B}$: We have the corresponding polynomials $P \in \mathbb{Z}_q[X,Y]$ and $Q \in \mathbb{Z}_q[Y]$ defined from the univariate polynomials extracted above: $P(X,Y) := \sum_{j=0}^{\ell} P_j(X) Y^j, Q(Y) := \sum_{j=0}^{\ell} p_j Y^j$.

We know by our assumption that in the output of $\mathcal{A}^*$ there exists at least one $j_0 \in \{0, \dots \ell\}$ such that $P_{j_0}(k) \neq p_{j_0}$. Then it follows that the previous defined P and Q do not satisfy the required statement: $P(k, Y) \neq Q(Y)$, wich breaks the soundness of the $\mathsf{P.SNARK}$. □

7.6 Proof Systems for Arithmetic Function Evaluation over Polynomial Rings

In this section we describe our SNARK for arithmetic computations in quotient polynomial rings.

Let $\mathbf{R}$ be the quotient ring $\mathbb{Z}/\langle R(X)\rangle$ for some polynomial $R \in \mathbb{Z}[X]$ of degree d. For a prime $q \gg d$ we define $\mathbb{F} = \mathbb{Z}_q$ a finite field and $\mathbb{R}_q = \mathbf{R}/q\mathbf{R}$. We want to construct a succinct non-interactive zero-knowledge argument system for some relation $\mathcal{R}_f$ of correct evaluation of an arithmetic function $f(\cdot) : {\mathbb{R}_q}^n \to \mathbb{R}_q$ taking $n \in \mathbb{N}$ inputs in the quotient ring $\mathbb{R}_q = \mathbf{R}/q\mathbf{R}$. The function f to be evaluated on polynomials $\{P_j\}_{j=1}^n$ in the quotient ring $\mathbb{R}_q$ is considered to be public.

Our scheme is a Commit-and-Prove (CaP) argument system. Given a compact commitment (C, ρ) to some polynomials $P_j \in \mathbb{R}\ \forall j = 1 \dots n$, the protocol allows to efficiently prove that some polynomial $P \in \mathbb{R}_q$ is the result of the evaluation of the function f on input the polynomials committed in C, evaluation done in the polynomial ring $\mathbb{R}_q$. The witness is the tuple $(\{P_j\}_{j=1}^n, \rho)$ where $(\{P_j\}_{j=1}^n, \rho)$ is the opening of C. We define the relation for which we construct a SNARK depending on the commitment public information ck for the commitment scheme $\mathsf{Uni}.\mathcal{M}ultiCom$ and on the function f:

$$\begin{aligned}\mathcal{R}_f \coloneqq \{(u = (C, P); w = (\{P_j\}_{j=1}^n, \rho,)) :\\ (C, \rho) = \mathsf{Uni.Com}(\{P_j\}) \ \wedge\ \exists\ T \in \mathbb{Z}_q[X] \text{ s.t. } P = f(P_j) - TR\}\end{aligned}$$

The relation $\mathcal{R}_f$ implicitly contains two bounds ℓ, ν on the number of inputs of f and the degree d_f of f as an arithmetic circuit.

The CaP-SNARK we will construct consists of the following building blocks: an ($\mathcal{R}_f$-independent) univariate-polynomial commitment scheme $\mathsf{Uni}.\mathcal{M}ultiCom = (\mathsf{Uni.Com}, \mathsf{Uni.ComVer}, \mathsf{Uni.OpenVer})$, a non-interactive zero-knowledge argument system $\mathsf{M.SNARK} = (\mathsf{M.Gen}, \mathsf{M.Prove}, \mathsf{M.Ver})$ for the simultaneous evaluation of n univariate polynomials $\{P_j\}_{j=1}^n$ in a point k, and a classical SNARK for arithmetic circuits over $\mathbb{Z}_q$, $\mathsf{CaP} - \Pi = (\Pi.\mathsf{Gen}, \Pi, \Pi.\mathsf{Ver})$ that must be compatible with the $\mathsf{Uni}.\mathcal{M}ultiCom$ scheme. One example of such a scheme is the adaptation of Pinocchio proposed by Veeningen in [Vee17].

High-Level Description of our SNARK. First, the prover computes the function f on polynomials $\{P_j\}_{j=1}^n$ and writes the result $f((P_j)_j) \in \mathbb{Z}_q[X]$ as $f(P_j) = P + TR$, for a quotient polynomial $T \in \mathbb{Z}_q[X]$ in order to compute the required value $P \in \mathbb{R}_q$ as $P = f(P_j) - TR$; Second, the prover builds a commitment C_T to this polynomial $T \in \mathbb{Z}_q[X]$ (which may have degree higher than that of R); In the next steps of the protocol, the prover will build a proof that $P = f((P_j)_j) - TR$ for the committed polynomials $\{P_j\}$ and T. However, instead of creating directly this proof, which would have to work for a large arithmetic circuits f, we use the homomorphic properties of the polynomial ring to "compress the computation". Namely, to prove $P = f((P_j)_j) - TR$, we evaluate all the polynomials in a random point k and then prove the relation on the resulting scalars, using the fact that:

$$\widehat{f}(P_j(k)) - R(k)T(k) = (f(P_j) - RT)(k) = P(k).$$

where $\widehat{f} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ is an arithmetic circuit that is the same as f except that every addition (resp. multiplication) in $\mathbb{R}_q$ is replaced by an addition (resp. multiplication) in $\mathbb{Z}_q$.

This idea is similar to the homomorphic hash function defined by Fiore et al. [FGP14]. In [FGP14], they let this idea work by evaluating the polynomials "in the exponent", i.e., they publish a set of group elements g^{k^i}, and then they compute homomorphically over these encodings to get $g^{P(k)}$. This technique however hits two problems: first, they cannot deal with modular reductions, and second, to compute homomorphically a multiplication on these encodings, they have to "consume" a pairing, and thus only degree-2 computations can be supported. In our case, we solve these issues by exploiting the power of the commit and prove paradigm in order to obtain, for every evaluation, a fresh random k. Then, having $k \in \mathbb{Z}_q$ allows us to support higher-degree computations as well as to deal with modular reductions.

To proceed with the protocol, the prover thus needs to get a random point k, not of its choice and independent of the values commited in C_T and C and of the statement P. This is possible by using the random oracle RO to obtain a value k on which it evaluates the polynomials $\{P_j(k) = p_j\}_{j=1}^n, R(k) = r, P(k) = p$ and $T(k) = t'$. Next, the prover compactly commits to the respective evaluations $(C', \rho') \leftarrow \mathsf{Uni.Com}(t', \{p_j\}_{j=1}^n)$.

At this point the prover will use: (1) the $\mathsf{M.SNARK}$ scheme to prove that C' is a commit to a vector of scalars $(t, \{p_j\}_{j=1}^n)$ that are the results of evaluating at the point k a vector of polynomials $(T, \{P_j\}_{j=1}^n)$ that are committed in $C_T \cdot C$[1]; (2) the $\mathsf{CaP} - \Pi$ scheme to prove that $p = P(k) = \widehat{f}((p_j)_j) - rt'$.

More formally, the algorithms of the protocol are described in Figure 7.2. A detailed intuition of the functionalities of each algorithm follows.

Formal Description of Our SNARK Protocol. Let $\mathsf{Uni}.\mathcal{M}ultiCom$ be a commitment scheme allowing to commit to multiple univariate polynomials. We construct a zero-knowledge SNARK scheme for any relation $\mathcal{R}_f$ with respect to some bounds ℓ, ν on the cardinal of $\{P_j\}_j$ and on the degree d_f of f.

CRS Generation. The setup algorithm $\mathsf{Gen}(1^\lambda, \mathcal{R}_f)$, given a security parameter $\lambda \in \mathbb{N}$ and an NP relation $\mathcal{R}_f$ including the bound parameters ℓ, ν, outputs a crs enabling the proof and verification of a function f of degree $d_f < \nu$ over a set of polynomials $\{P_j\}_{j=1}^n$ of cardinality $n \leq \ell$.

The crs elements are generated in order to allow the derivation of a commitment key ck compatible with $\mathsf{Uni}.\mathcal{M}ultiCom$ with some special property (steps 3 and 4 in the Gen algorithm from Figure 7.2): It supports commitments up to ℓ different polynomials $P_j \in \mathbb{R}_q$ (all of degrees $\leq d$) and has the two sets of bases $\{g^{s^i}\}_{i=0}^\nu$ and $\{\widehat{g}^{s^i}\}_{i=0}^\nu$ that enable higher degree up to ν to commitment to polynomials $T \in \mathbb{Z}_q[X], T = \sum_{i=0}^\nu T_i X^i$. The generation of such string is not necessaraly black-box, and can exploit some elements from the one generated previously to define the two bases: $\{\mathsf{gk}, (g_{i0})_{i=0}^\nu, (\widehat{g}_{i0})_{i=0}^\nu\}$.

Then we need a common reference string for the $\mathsf{CaP} - \Pi$ that will be used for proving computations of $\widehat{f}$. For this part, we exploit the structure of the commit-and-prove scheme proposed in [Vee17]. In particular, this scheme works with a commitment key that has the same structure as (a portion of) ours, i.e., it consists of $\mathbb{G}_1$ elements[2] $\widehat{\mathsf{ck}} = (g^{\widehat{r}}, g^{\widehat{\alpha r}}, \{g^{\widehat{x^i}}, g^{\widehat{\alpha x^i}}\}_{i=1}^\ell)$. Thus, we can use our bases and set $\widehat{\mathsf{ck}} = (\mathsf{gk}, h, \widehat{h}, \{g_{0j}, \widehat{g}_{0j}\}_{j=0}^\ell)$.

[1] By the homomorphic property of the commitments and a specific organization of the bases in ck, $C_T \cdot C$ results in a commitment to the concatenation $(T, \{P_j\}_{j=1}^n)$, as we will explain in detail.

[2] In [Vee17] the elements used to make the commitment knowledge-extractable are in $\mathbb{G}_2$, but the protocol can be trivially modified to work with them in $\mathbb{G}_1$ as in our case.

$\underline{\mathsf{Gen}(1^\lambda, \mathcal{R}_f) \to \mathsf{crs}}$

1: $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda);\ g, h \leftarrow_\$ \mathbb{G},\ \mathfrak{g} \leftarrow_\$ \mathfrak{G};\ \alpha, s, t \leftarrow_\$ \mathbb{Z}_q$
2: $\widehat{g} := g^\alpha, \widehat{h} := h^\alpha, \widehat{\mathfrak{g}} := \mathfrak{g}^\alpha,\ \mathfrak{g}_1 := \mathfrak{g}^s, h_1 := h^s$
3: $g_{i0} := g^{s^i},\ \widehat{g}_{i0} := \widehat{g}^{s^i} \quad \forall\ 0 \le i \le \nu$
4: $g_{ij} := g^{s^i t^j},\ \widehat{g}_{ij} := \widehat{g}^{s^i t^j}\ \forall\ 0 \le i \le d, 1 \le j \le \ell$
5: $\mathsf{crs}_C := (\mathsf{ck}, (h_1, \mathfrak{g}_1), \mathsf{RO}),\ \mathsf{crs}' \leftarrow \mathsf{G}(\widehat{\mathsf{ck}}, \mathcal{R}_{\widehat{\mathsf{f}}})$
6: return crs as in Equation (7.3)

$\underline{\mathsf{Prove}(\mathsf{crs}, x, w)}$

1: $(C, P) := x, (\{P_j\}_{j=1}^n, \rho) := w$
2: $T := \frac{f(P_j) - P}{R}$
3: $(C_T, \tau) \leftarrow \mathsf{Uni.Com}(\mathsf{ck}, T)$
4: $k \leftarrow \mathsf{RO}(C, P, C_T)$
5: $p = P(k), r = R(k),$
6: $t' = T(k), p_j = P_j(k)$
7: $(C', \rho') \leftarrow \mathsf{Uni.Com}(t', \{p_j\})$
8: $u_C := (C_T \times C, C', k)$
9: $\pi_C \leftarrow \mathsf{M.Prove}(\mathsf{crs}_C, u_C, w_C),$
10: $\pi' \leftarrow \Pi(\mathsf{crs}', u' = (C', p, r), w')$
11: returns $\pi = (C_T, C', \pi_C, \pi')$

$\underline{\mathsf{Ver}(\mathsf{crs}, x = (C, P), \pi)}$

1: $\pi := (C_T, C', \pi_C, \pi')$
2: $k \leftarrow \mathsf{RO}(C, P, C_T)$
3: $p := P(k), r := R(k)$
4: $u_C := (C_T \times C, C', k)$
5: $u' := (C', p, r)$
6: $b_C \leftarrow \mathsf{M.Ver}(\mathsf{crs}_C, u_C, \pi_C)$
7: $b' \leftarrow \Pi.\mathsf{Ver}(\mathsf{crs}', u', \pi')$
8: return $(b_C \wedge b') = 1.$

Figure 7.2: A SNARK for evaluations in polynomial rings

Once defined the commitment key $\widehat{\mathsf{ck}}$ as above, in step 5, we invoke the key generation algorithm $\mathsf{G}(\widehat{\mathsf{ck}}, \mathcal{R}_{\widehat{\mathsf{f}}})$ of the scheme $\mathsf{CaP}-\Pi$ in [Vee17] in order to obtain a crs crs' corresponding to the relation $\mathcal{R}_{\widehat{f}}$.We will call the extra part of the crs' related to the QAP generation $\mathsf{crs}_{\mathsf{QAP}}$. The rest of crs' is found in the already generated elements of our global crs.

The resulting crs has then the following expression:

$$\mathsf{crs} : (\mathsf{ck} = (\mathsf{gk}, (g_{i0}, \widehat{g}_{i0})_{i=0}^{\nu}, (g_{ij}, \widehat{g}_{ij})_{i=0,j=1}^{d,\ell}, (h, \widehat{h}), (\mathfrak{g}, \widehat{\mathfrak{g}})); (h_1, \mathfrak{g}_1); \mathsf{crs}_{\mathsf{QAP}}; \mathsf{RO}) \tag{7.3}$$

Prover. Given a reference string crs, a compact commitment (C, ρ) and the corresponding polynomials $\{P_j\}_{j=1}^n \in \mathbb{R}_q$, the prove algorithm produces a proof π that $f(P_j) = P$ as follows:

After having computed $f(P_j) \in \mathbb{Z}_q[X]$ and written it as $f(P_j) = P + TR$ for a quotient polynomial $T \in \mathbb{Z}_q[X]$, the prover algorithm has then to commit to $T = \sum_{i=0}^{\nu} T_i X^i$. This is possible by using the bases $\{g^{s^i}\}_{i=0}^{\nu}$ and $\{\widehat{g}^{s^i}\}_{i=0}^{\nu}$ in ck for degrees up to ν. The commitment $(C_T = (c_T, \widehat{c}_T), \tau)$ is computed with $c_T = h^\tau \prod_{i=0}^{\nu} g_{i0}^{T_i}$, $\widehat{c}_T = \widehat{h}^\tau \prod_{i=0}^{\nu} \widehat{g}_{i0}^{T_i}$.

The prover then runs the $\mathsf{RO}(C, P, C_T)$ function to obtain a random value k on which it evaluates the polynomials $\{P_j(k) = p_j\}_{j=1}^n, R(k) = r$, $P(k) = p$ and $T(k) = t$ (see steps 4 to 6 in Figure 7.2).

The prover carefully commits the respective evaluations $(C' = (c', \widehat{c}', \rho') \leftarrow \mathsf{Uni.Com}(t', \{p_j\}_{j=1}^n)$,

by using the appropriate basis as follows: $c' = h^{\rho'} g_{00}^{t'} \times \prod_{j=1}^{\ell} g_{0j}^{p_j}$, $\widehat{c}' = \widehat{h}^{\rho'} \widehat{g}_{00}^{t'} \times \prod_{j=1}^{\ell} \widehat{g}_{0j}^{p_j}$. Then, it computes a unique compact commitment to the polynomials $T, \{P_j\}_{j=1}^n$ by multiplying the two commitments $C_T \times C$. From the homomorphic property of commitment scheme, we obtain a new valid commitment with randomness the sum of initial randomnesses $(C_T \times C, \rho + \tau) \coloneqq \mathsf{Uni.Com}(T, \{P_j\}_{j=1}^n)$.

The prover runs the algorithm $\mathsf{M.Prove}(\mathsf{crs}_C, u_C = (C_T \times C, C', k), w_C = (T, \{P_j\}, t', \{p_j\}, \rho + \tau, \rho'))$ and the proving algorithm of the $\mathsf{CaP} - \Pi$ for proving the evaluation of $\widehat{f}$ on scalars $\pi' \leftarrow \Pi(\mathsf{crs}', u' = (C', p, r), w')$ where w' is the QAP witness for the computation $p = \widehat{f}(p_j) - rt$ and for the opening of C'. The prover eventually outputs $\pi = (C_T, C', \pi_C, \pi')$.

Verifier. The algorithm Ver on input a statement $u = (C, P)$ and a proof $\pi \coloneqq (C_T, C', \pi_C, \pi')$ recomputes the randomness k by running $k \leftarrow \mathsf{RO}(C, P, C_T)$. Then the Verifier has only to evaluate the known polynomials P, R in k obtaining $p \coloneqq P(k), r \coloneqq R(k)$. Once it has all the elements to redefine the two statements $u_C \coloneqq (C_T \times C, C', k)$ and $u' \coloneqq (C', p, r)$ for the proofs π_C and π' it runs the corresponding verification algorithms of these two SNARKs, $\mathsf{M.Ver}$ and $\Pi.\mathsf{Ver}$ to check the proofs and outputs the conjunction of the two answers.

7.6.1 Relations for the Two SNARKs

We define the intermediate statements $\mathcal{R}_1, \mathcal{R}_2$ to be proven using the two SNARKs, $\mathsf{M.SNARK}$ and $\mathsf{CaP} - \Pi$:

St_1: $\mathcal{R}_{\mathsf{eval}}$. We first define the relation for simultaneous evaluation of multiple polynomials on a point k, to be used for $\mathsf{M.SNARK}$. The prover has to convince the verifier that for a given point k (that in our case is random, but part of the statement) and two commitments $C_T \times C$ and C', it knows the corresponding opening values $(T, \{P_j\}_j, \rho + \tau)$ and $(t', \{p_j\}_j, \rho')$ such that $P_j(k) = p_j$ for all j, and $T(k) = t'$.

More formally, the first proof takes as input a tuple $u_C = (C_T \times C, C', k)$, and proves there is a witness $w_C = (T, \{P_j\}, t', \{p_j\}, \rho + \tau, \rho')$ such that the verifier accepts iff $\mathcal{R}_{\mathsf{eval}}$ holds for (u_C, w_C):

$$\mathcal{R}_{\mathsf{eval}} \coloneqq \{(u_C, w_C) : \forall j,\ p_j = P_j(k) \ \wedge\ t' = T(k) \ \wedge\ (C', \rho') = \mathsf{Uni.Com}(t', \{p_j\}) \\ \wedge\ (C_T \times C, \rho + \tau) = \mathsf{Uni.Com}(T, \{P_j\}_j)\}.$$

St_2: $\widehat{\mathcal{R}}_f$. We then define the relation for correct computation of $\widehat{f}$, to be used for $\mathsf{CaP} - \Pi$. The prover has to convince the verifier that an equality holds for some scalar values $t', \{p_j\}, p, r \in \mathbb{Z}_q$. The inputs p, r are known by the verifier (they are public) and $t', \{p_j\}$ are given implicitly in a committed form: $(C', \rho') = \mathsf{Uni.Com}(t', \{p_j\})$. More formally, for the statement $u' = (C', p, r)$, there exists a witness w' for the satisfiability of the circuit verifying whether the equation $p = \widehat{f}(p_j) - rt'$ holds or not:

$$\widehat{\mathcal{R}}_f \coloneqq \{(u', w') : p = \widehat{f}(p_j) + rt' \ \wedge\ t' = T(k) \ \wedge\ (C', \rho') = \mathsf{Uni.Com}(t', \{p_j\})\}.$$

With these two relations defined, we can consider the two common reference strings $\mathsf{crs}_C, \mathsf{crs}'$ generated by running $\mathsf{M.Gen}(1^\lambda, \mathcal{R}_{\mathsf{eval}})$ and $\Pi.\mathsf{Gen}(1^\lambda, \widehat{\mathcal{R}}_f)$, and run the algorithms of the two SNARKs using these crs.

7.6.2 Security Analysis

About the above construction, we can state the following security result.

Theorem 7.6.1. *Assuming the* $\mathsf{CaP} - \Pi$ *and* $\mathsf{M.SNARK}$ *schemes are secure arguments of knowledge, the new construction* $\mathsf{CaP} - SNARK$ *described before satisfies* completeness, succinctness, zero-knowledge *and* knowledge-soundness.

Before proceeding to the detailed proof of the theorem, we provide a short intuition. Correctness is rather straightforward, and zero-knowledge follows from the the zero-knowledge property of the two SNARKs and the perfect hiding of the commitment scheme. For knowledge soundness, the proof consists of two main steps. First, we rely on the knowledge-soundness of the two SNARKs to show that for any adversary creating an accepting proof there is a knowledge extractor that, with all but negligible probability, returns witnesses that correctly satisfy the two relations mentioned in the previous section. Second, the only remaining possibility is that the polynomial $V = P^* - f(P_j) + TR$ is nonzero. However, $V(k) = 0$ and this holds for a random point k sampled by the random oracle independently of V, which can happen only with probability $\deg(V)/q$ which is negligible.

7.7 Applications to Computing on Encrypted Data

In this section we detail on how we can use our SNARK for computations over polynomial rings to build a VC scheme that guarantees input and output privacy.

7.7.1 Verifiable Computation

Here we recall the notion of *verifiable computation* from [GGP10]. We adapt the definitions to fit the setting (that is in the scope of our construction) where we have public verifiability and public delegatability [PRV12], as well as privacy of the inputs and outputs. A VC scheme $\mathcal{VC} = (\mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Verify}, \mathsf{Decode})$ consists of the following algorithms:

$\mathsf{KeyGen}(1^\lambda, \widehat{f}) \to (PK_f, SK_f)$: Given the security parameter, the key generation algorithm outputs a public key and a matching secret key for the function f.

$\mathsf{ProbGen}_{PK_f}(\mathbf{x}) \to (\sigma_x, \tau_x)$: The problem generation algorithm uses the public key PK_f to encode the input x into a public value σ_x, to be given to the computing party, and a public value τ_x to be given to the verifier.

$\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$: Given the public key PK_f and the encoded input, the compute algorithm returns an encoded version of the function's output.

$\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: Given the public key PK_f for function f, and the public verifier information τ_x, the verification algorithm accepts (output $acc = 1$) or rejects (output $acc = 0$) an output encoding σ_y.

$\mathsf{Decode}_{SK_f}(\sigma_y) \to y$: Given the secret key SK_f for function f, and an output encoding σ_y, the decoding algorithm outputs a value y.

The correctness of a VC scheme is the obvious property: if one runs $\mathsf{Compute}$ on an honestly generated input encoding of $\mathbf{x}$, then the output must verify and its decoding should be $y = f(\mathbf{x})$.

$\mathbf{Exp}_{\mathcal{A}}^{PubVerif}(\mathcal{VC}, f, \lambda)$

$(PK, SK) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$
$\mathbf{x} \leftarrow \mathcal{A}(PK_f)$
$(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK_f}(\mathbf{x})$
$\widehat{\sigma}_y \leftarrow \mathcal{A}(PK_f, \sigma_x, \tau_x)$
$\widehat{acc} \leftarrow \mathsf{Verify}_{PK_f}(\tau_x, \widehat{\sigma}_y)$
$\widehat{y} \leftarrow \mathsf{Decode}_{SK_f}(\widehat{\sigma}_y)$
return $(\widehat{acc} = 1) \ \wedge \ (\widehat{y} \neq f(\mathbf{x}))$

$\mathbf{Exp}_{\mathcal{A}}^{Priv}(\mathcal{VC}, f, \lambda)$

$b \xleftarrow{\$} \{0, 1\}$
$(PK_f, SK_f) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$
$(\mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}(PK_f)$
$(\sigma_b, \tau_b) \leftarrow \mathsf{ProbGen}_{PK_f}(\mathbf{x}_b)$
$b' \leftarrow \mathcal{A}(PK_f, \sigma_b)$
return $(b' = b)$

Figure 7.3: Experiments for publicly verifiable and private VC.

For security, intuitively we want to say that an adversary that receives the public parameters for a function f and an encoding of an input $\mathbf{x}$ cannot create an encoding that passes verification and decodes to $y' \neq f(\mathbf{x})$. More formally, we say that a publicly verifiable computation scheme $\mathcal{VC}$ is *secure* for a function f, if for any PPT adversary $\mathcal{A}$, we have that

$$\Pr[\mathbf{Exp}_{\mathcal{A}}^{PubVerif}[\mathcal{VC}, f, \lambda] = 1] = \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{Exp}^{PubVerif}$ is described in Figure 7.3.

The *input privacy* notion intuitively says that no information about the inputs is leaked. This is defined using a typical indistinguishability experiment. Note that input privacy implies also *output privacy*. More formally, we say that a publicly verifiable (and publicly delegatable) VC scheme $\mathcal{VC}$ is *private* for a function f, if for any PPT adversary $\mathcal{A}$, we have that:

$$\Pr[\mathbf{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC}, f, \lambda] = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the experiment $\mathbf{Exp}_{\mathcal{A}}^{Priv}$ is described in Figure 7.3.

7.7.2 Our VC scheme

We describe our VC scheme below. The construction is essentially an instantiation of the generic solution of Fiore et al. [FGP14] when using an homomorphic encryption scheme whose homomorphic evaluation algorithm fits our relation $\mathcal{R}_f$. This can be obtained by using HE schemes in the Ring-LWE setting where the ciphertext space works over the same ring $\mathbb{R}_q$ supported by our SNARK constructions, and where the evaluation algorithm does not involve modulus switches and rounding operations. An example of such a scheme is the one of Brakerski and Vaikunthanatan [BV11b].

$\mathsf{KeyGen}(1^\lambda, \widehat{f}) \rightarrow (PK_f, SK_f)$:

- Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(\lambda)$ to generate a key pair for HE.
- Run $\mathsf{crs} \leftarrow \mathsf{Gen}(\mathcal{R}_f, \lambda)$ to generate the common reference string of our SNARK for the relation $\mathcal{R}_f$.

- Set $PK_f = (\mathsf{pk}, \mathsf{crs}, \widehat{f})$ and $SK_P = (\mathsf{sk}, \mathsf{crs})$.

$\mathsf{ProbGen}_{PK_f}(\mathbf{x}) \to (\sigma_x, \tau_x)$:

- Parse $\mathbf{x} = \{x_i\}_{i=1}^n$ and compute ciphertexts $P_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, x_i)$
- Compute the commitment $(C, \rho) = \mathsf{Uni.Com}(\{P_i\})$ and define $\sigma_x = (C, \{P_i\}, \rho)$ and $\tau_x = C$.

$\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$:

- Parse $\sigma_x = (C, \{P_i\}, \rho)$;
- Compute the result ciphertext $P \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, \widehat{f}, \{P_i\}) = f(\{P_i\})$.
- Run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, (C, P), (\{P_i\}, \rho))$.
- Define $\sigma_y = (P, \pi)$

$\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: output $b \leftarrow \mathsf{Ver}(\mathsf{crs}, (C, P), \pi)$.

$\mathsf{Decode}_{SK_f}(\tau_x, \sigma_y) \to y$: Decrypt $y = \mathsf{HE.Dec}(\mathsf{sk}, P)$.

Following the general result in [FGP14], the scheme satisfies correctness, security and privacy. In particular, privacy relies on the semantic security of HE, and security on the soundness of the SNARK.

7.7.3 Preserving Privacy of the Inputs Against the Verifier

The VC scheme described in the previous section works when the homomorphic computation $P \leftarrow f(\{P_i\})$ on the ciphertexts is deterministic. This can raise the issue that the result ciphertext P may reveal information on the plaintexts $\{x_i\}$ underlying $\{P_i\}$ (e.g., in lattice-based schemes such information may be inferred by looking at the distribution of the noise recovered as P's decryption time).

It would be therefore interesting to capture the setting where one wants to hide information on the x_i's even from the decryptor. Such a property would turn useful in scenarios where the data encryptor and decryptor are different entities. As an example, consider the case of users that store medical data x on a cloud server which computes some query f on behalf of an analyst, who however is not entitled to learn more than $f(x)$.

In this section, we provide a formal definition of this property, that we call context-hiding, and then describe how our scheme from the previous section can be extended to achieve this additional property.

Defining Context-Hiding. Informally, this property says that output encodings σ_y, as well as the input verification tokens τ_x, do not reveal any information on the input $\mathbf{x}$. Notably this should hold even against the holders of the secret key SK_f. We formalize this definition in a zero-knowledge style, requiring the existence of simulator algorithms that, without knowing the input, should generate (τ_x, σ_y) that look like the real ones. More precisely, a VC scheme is context-hiding for a function f if there exist simulator algorithms S_1, S_2 such that:

- the keys (PK_f, SK_f) and (PK'_f, SK'_f) are statistically indistinguishable, where $(PK_f, SK_f) \leftarrow \mathsf{KeyGen}(1^\lambda, f)$ and $(PK_f, SK_f, \mathsf{td}) \leftarrow S_1(1^\lambda, f)$;
- for any input $\mathbf{x}$, the following distributions are negligibly close

$$(PK_f, SK_f, \sigma_x, \tau_x, \sigma_y) \approx (PK_f, SK_f, \sigma_x, \tau'_x, \sigma'_y)$$

where $(PK_f, SK_f, \mathsf{td}) \leftarrow S_1(1^\lambda, f)$, $(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{PK_f}(\mathbf{x})$, $\sigma_y \leftarrow \mathsf{Compute}_{PK_f}(\sigma_x)$, and $(\sigma'_y, \tau'_x) \leftarrow S_2(\mathsf{td}, SK_f, f(\mathbf{x}))$.

Our Context-Hiding Secure VC scheme. Before describing the scheme in detail, let us provide some intuition. The first observation is that for the HE scheme this problem can be solved by adding to the result P an encryption of 0, P_0^*, whose noise can statically hide that in P (a so called noise flooding technique). However if we do this change in our VC scheme we have two issues: (1) the computation is not deterministic anymore; (2) the prover may create a bogus encryption of 0, not of the correct distribution, in order to make decryption fail. We can solve these issues by using the fact that, as underlying tool for verifiability, we are using a SNARK that can handle deterministic computations. In particular, we can do the following.

For (2) we add to the public key s honestly generated encryptions of 0 $\{P_i^*\}_{i=1}^s$, and then ask the untrusted party to compute the result as $P' = P + P_0^*$ with $P_0^* = \sum_{i=1}^n b_i \cdot P_i^*$, for uniformly random bits b_i. By choosing appropriately the noise parameters in the P_i^*'s and by taking $s \approx \lambda$, based on the leftover hash lemma, P_0^* can statistically hide the noise in P. Formally, adding such a randomization at the end of computing a function f guarantees leveled circuit privacy. In a nutshell, a somewhat-FHE HE is leveled circuit private if there exists a simulator algorithm $\mathsf{HE.S}$ such that $\mathsf{HE.S}(\mathsf{pk}, d, f(\mathbf{x})) \approx \mathsf{HE.Eval}(\mathsf{pk}, f, \mathsf{HE.Enc}(\mathbf{x}))$ are statistically close. Here the input d taken by the simulator represents information on the depth of f.

For (1), we simply consider proving a slightly different relation, that is:

$$\begin{aligned}\mathcal{R}_f^* \coloneqq \{(u = (C, P', \{P_i^*\}_{i=1}^s); w = (\{P_j^*\}_{j=1}^n, \rho, b_1, \ldots, b_s)) :\\ (C, \rho) = \mathsf{Uni.Com}(\{P_j\}) \ \wedge \ \forall i \in [s] b_i \in \{0,1\} \ \wedge \ \exists\, T \in \mathbb{Z}_q[X] \text{ s.t.}\\ P' = f(P_j) + \sum_{i=1}^{s} b_i P_i^* - TR \ \}\end{aligned}$$

In order to use our SNARK on this relation, we can do the following. Given a function $\widehat{f} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ define the function $\widehat{f}'$ that takes $n + 2s$ inputs such that

$$\widehat{f}'(x_1, \ldots, x_n, o_1, \ldots, o_s, b_1, \ldots, b_s) = \widehat{f}(x_1, \ldots, x_n) + \sum_{i=1}^{s} b_i \cdot o_i.$$

Then we use our SNARK for the following relation

$$\begin{aligned}\mathcal{R}'_f \coloneqq \{(u = (C', P'); w = (\{P_j^*\}_{j=1}^n, \{P_i^*\}_{i=1}^s, \{b_i\}_{i=1}^s, \rho')) :\\ (C', \rho') = \mathsf{Uni.Com}(\{P_j\}, \{P_i^*\}, \{b_i\}) \ \wedge \ \forall i \in [s] b_i \in \{0,1\} \ \wedge \ \exists\, T \in \mathbb{Z}_q[X] \text{ s.t.}\\ P' = f'(P_j, \{P_i^*\}, \{b_i\}) - TR \ \}\end{aligned}$$

that can be seen as matching the format $\mathcal{R}_{f'}$, for the function f' and a larger set of inputs, of relations supported by our SNARK. One change however is that the commitment C' cannot be created by $\mathsf{ProbGen}$ as it contains elements that depend on a specific computation. We solve this problem by using the homomorphic properties of our commitment: namely we assume that at key generation a commitment $(C^*, \rho^*) = \mathsf{Uni.Com}(\{P_i^*\})$ is created and made public, and that the prover creates a similar commitment $(C_b, \rho_b) = \mathsf{Uni.Com}(\{b_i\})$ to the random coefficients. Then C' can be obtained as $C \cdot C^* \cdot C_b$ and its opening is $\rho' = \rho + \rho^* + \rho_b$.

A more precise description of the protocol is given below.

$\mathsf{KeyGen}(1^\lambda, \widehat{f}) \to (PK_f, SK_f)$:

- Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(\lambda)$ to generate the key pair for HE.
- Run $\mathsf{crs} \leftarrow \mathsf{Gen}(\mathcal{R}_{f'}, \lambda)$ to generate the common reference string of our SNARK for the relation $\mathcal{R}_{f'}$.
- For $i = 1$ to s: $P_i^* \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, 0)$ and compute a commitment $(C^*, \rho^*) = \mathsf{Uni.Com}(\{P_i^*\})$.
- Set $PK_f = (\mathsf{pk}, \{P_i^*\}_{i=1}^s, C^*, \rho^*, \mathsf{crs}, \widehat{f})$ and $SK_P = (\mathsf{sk}, \mathsf{crs})$.

$\mathsf{ProbGen}_{PK_f}(\mathbf{x}) \to (\sigma_x, \tau_x)$: this is the same as in the previous section.

$\mathsf{Compute}_{PK_f}(\sigma_x) \to \sigma_y$: parsing $\sigma_x = (C, \{P_i\}, \rho)$, do the following:

- Sample $b_1, \ldots, b_s \leftarrow\!\!\$\, \{0,1\}$ uniformly at random, and compute a commitment $(C_b, \rho_b) = \mathsf{Uni.Com}(\{b_i\})$ (thinking of each b_i as a degree-0 polynomial).
- Compute the result ciphertext $P' \leftarrow f(\{P_i\}) + \sum_{i=1}^s b_i P_i^*$.
- Run $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, (C \cdot C^* \cdot C_b, P'), (\{P_i\}, \{P_i^*\}, \{b_i\}, \rho + \rho^* + \rho_b))$.
- Define $\sigma_y = (P', C_b, \pi)$

$\mathsf{Verify}_{PK_f}(\tau_x, \sigma_y) \to acc$: output $b \leftarrow \mathsf{Ver}(\mathsf{crs}, (C \cdot C^* \cdot C_b, P), \pi)$.

$\mathsf{Decode}_{SK_f}(\tau_x, \sigma_y) \to y$: Decrypt $y = \mathsf{HE.Dec}(\mathsf{sk}, P')$.

Theorem 7.7.1. *If* HE *is semantically secure and circuit private, and* Π *is knowledge sound and zero-knowledge, then the VC described above is correct, secure, private and context-hiding.*

Sketch. The proof of the result is rather simple. Below we provide a proof sketch. First , notice that based on the correctness of our SNARK and that of HE, we obtain correctness of our protocol.

The security follows from the knowledge soundness of the SNARK. The only detail to mention is that we also rely on the correctness of the HE scheme in order to make sure that, for honestly generated ciphertexts $\{P_i\}$ of $\{x_i\}$, and $\{P_i^*\}$ for 0, and for binary coefficients $\{b_i\}$, the ciphertext $P' \leftarrow f(\{P_i\}) + \sum_{i=1}^s b_i P_i^*$ decrypts to $\widehat{f}(x)$.

Finally, we can prove context-hiding via a simple hybrid argument based on the privacy property of the HE scheme and the zero-knowledge of our SNARK. We define the $\mathcal{VC}$ simulators as follows. S_1 proceeds exactly as KeyGen except that it runs the SNARK simulator $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{crs}}(\mathcal{R}_{f'}, \lambda)$ instead of Gen, and set its trapdoor to be td. $S_2(\mathsf{td}, SK_f, y)$ first sets $\tau_x' = C$ where C is created as a commitment to some dummy input. Next, it creates C_b as another commitment to a dummy value, and computes P' as an encryption of y using $\mathsf{HE.S}(\mathsf{pk}, d, y)$ (where d is information on the depth of f), and finally it invokes the SNARK simulator $\pi \leftarrow \mathsf{Sim}(\mathsf{crs}, (C \cdot C^* \cdot C_b, P'))$. Then S_2 outputs τ_x' and $\sigma_y' = (P', C_b, \pi)$.

The indistinguishability of the keys is immediate from the zero-knowledge of the SNARK. For the second property, we can define an hybrid simulator S' that, with knowledge of σ_x, runs as S_2 but creates P' as in $\mathsf{Compute}$. It is easy to see that the output of S' is indistinguishable from that of S_2 by the property of $\mathsf{HE.Hide}$, also by the hiding of the commitment and by the zero-knowledge of the SNARK we obtain that the values (τ_x', σ_y') generated by S' are indistinguishable from the ones generated using $\mathsf{ProbGen}$ and $\mathsf{Compute}$.

□

Chapter 8

Conclusion

The contributions detailed in this thesis focus on the design and the analysis of SNARK systems, and target their applications to secure delegated computation. We presented along this manuscript several results that deal with different aspects of SNARK protocols. First, we have surveyed the area of SNARKs (cf. Chapter 3), and gave the state of art and some necessary background for better understanding our results. In Chapter 4, we introduced a new framework that can accommodate a SNARK construction secure against quantum attacks. Then, in Chapter 5 we studied in more detail the problematic of extraction in the security definition of SNARKs. We gave a new formalisation of the knowledge soundness property in presence of adversaries with Oracle access, called O-SNARK and showed some positive and negative results for this new notion. In the following chapter, as a follow-up of the previous, we gave some use cases of our new notion of O-SNARK, in the area of homomorphic authentication protocols.

Finally, in Chapter 7 we proposed a novel solution to secure computations even over encrypted data, meaning that in an outsourced computation scenario we further ask for the privacy of the input.

Our results aimed at improving SNARKs both from a theoretical and a practical point of view, but there are still questions that remain open. Here we will state some of them.

Open Questions

Post-Quantum SNARK. In our first contribution, the Post-Quantum SNARK construction, we are dedicated to provididing real-world solutions for post-quantum zk-SNARKs.

Our work combines some known techniques that are further adapted to a "post-quantum" setting. We construct a SNARK scheme that relies only on lattice-based assumptions which are claimed to hold even in the presence of quantum computers. This combination and adaptation to lattices requires technical and conceptual non-trivial work and allows us to build the first designated-verifier zk-SNARK on lattices with shorter CRS and weaker assumptions than all other existing schemes.

Still, publicly-verifiable SNARKs have a wider range of applications, as they are a core primitive in blockchains, anonymous credentials and verifiable computation in general. Therefore, we believe that it is a problem of great interest to find out new efficient constructions of public-verifiable SNARKs, that are secure against quantum adversaries.

This leads to our first open question:

Question 8.1. *Is it possible to instantiate efficient publicly-verifiable zk-SNARKS, from post-quantum assumptions?*

We point out that a publicly-verifiable SNARK construction that is quantum-secure is already known in the random oracle model (using the Fiat-Shamir heuristic). However, even the best optimized version recently proposed, called STARK – which can be instantiated with a post-quantum collision-resistant hash function – does not seem to scale well for even moderately complex computations.

This suggests an alternative open question, which is more of theoretical interest, as it calls for a better understanding of the mechanism necessary in our framework to allow for the verification to be done publicly:

Question 8.2. *Is there a lattice equivalent of a bilinear map that enables quadratic root check in a public way?*

It seems difficult to achieve public verifiability without some bilinear pairing map. However, the discovery of such a map would constitute a major breakthrough in cryptography.

Even if not publicly-verifiable, one of the main contributions of this chapter is the prototype implementation of a new lattice-based zk-SNARK. Compared with the state of the art, this is far from being practical, but it is still satisfiable for a first attempt to make SNARKs quantum-resilient. One natural question would be to improve the efficiency of this implementation to make it comparable with a pairing-based system.

Question 8.3. *Is it possible to get better efficiency for post-quantum SNARK to make them an alternative in practice?*

Our approach to post-quantum zk-SNARKs might require further optimization to come to daily use. Nevertheless, it is already a significant step to the progress that zk-SNARKs have made in the past years. Despite its limitations, we believe that this result raises many interesting questions both in the lattice community as well as for the SNARK's designers.

O-SNARKs. In Chapter 5 we study the extractability property necessary to define knowledge soundness of a SNARK scheme.

In line with recent work [BCPR14, BP15] on the feasibility of extraction in the presence of auxiliary input, our results indicate that additional care must be taken when considering extraction *in the presence of oracles*.

We show that extraction in this setting becomes impossible by only assuming collision-resistance hash functions. We try to give some conditions on the oracles in order to overcome this impossibility and we study some class of signature schemes and their signing oracles. Furthermore, our work establishes a framework that eases the analysis and the use of SNARK extractors in all those security experiments where these are given access to an oracle.

The main question we try to answer is:

Question 8.4. *Are there oracle families for which classical SNARKs are O-SNARKs?*

Unfortunately, our negative result shows that any classical SNARK is not an O-SNARK for *any* oracle. We constructed a counterexample in order to prove this impossibility and this example can be easily extended to other oracles, so our impossibility will hold for those oracles too.

We would like to be able to classify better the oracle families for which our counterexample cannot be applied.

We have a positive answer in the random oracle model, by the use of *hash and sign techniques*. Adaptiveness is definitely a main inconvenient in our black-box scenario (indeed, without adaptive access we have non-adaptive O-SNARKs from SNARKs) together with the fact that the oracle has a secret state. The malign information that the oracle can provide to the adversary can be removed by randomization in the random oracle model, but is harder to remove in standard model and we don't know any technique in this sense; this is an open problem to explore.

On the other side, we remark that our counterexamples are of artificial nature and do not give evidence of extraction's impossibility in the presence of "natural" oracles. This leads us to anther open question:

Question 8.5. *Can we find more natural examples to prove the impossibility of extraction in the standard model?*

Indeed a downside of our counterexamples is that they are involved and artificial. Therefore, we would like to understand under what properties (or restrictions) a "natural" oracle can be safely used with a SNARK.

Another open question that follows from our attempt of circumventing the negative example is:

Question 8.6. *Can we characterize the classes of signature schemes for which O-SNARKs exist?*

We show that we can overcome the impossibility by limiting the message space of the signature or the adaptive power of the adversary, but it would be interesting to find wider classes of signatures for which the signing oracles behave well for the extraction.

Besides its theoretical interest, the latter question also has a practical motivation since there are several efficient SNARK constructions proposed in the last years that one might like to use in larger authentification protocols. This line of work's goal is improving the understanding of a technical aspect of SNARKs, cleaning up flaws in prior work and giving a framework for extraction in the presence of oracles.

Applications of O-SNARK. This chapter is mostly a collection of applications of O-SNARKs focused on authentification protocols.

As the previous chapter deals with ways to circumvent the impossibility results for the case of signing oracles, the ultimate goal is obtaining secure instantiations of the homomorphic, functional and aggregated signatures constructions under well-specified assumptions.

Homomorphic signatures from SNARKs were mentioned as "folklore" in several works. None of these works ever formalized a proof of this construction. More and more such schemes are bursting in the literature recently, and hopefully, we now have the adequate building block to use for their construction.

SNARK with Data Privacy. In this chapter, we also address the other crucial challenge introduced by delegated computation: data confidentiality. This new dual problem can be formulated as:

Question 8.7. *Can we get practical schemes that achieve both privacy and integrity in delegated computations?*

We try to answer this question in the last part of the manuscript. We are motivated by the scenario where we wish to prove computations over encrypted data such that this preserves the privacy of the inputs and the outputs against the prover.

Moreover, in this scenario, one must be able to efficiently verify the correctness of the result despite not knowing (anymore) the inputs of the delegated computation.

We propose a new protocol for verifiable computation on encrypted data that improves on the state-of-the-art solutions in multiple aspects discussed in the chapter.

This novel construction attempts to lower down the prover overhead (including the size of the proof) by exploiting the properties of FHE schemes based on ring-LWE and by building tailored SNARKs for polynomial evaluation.

We overpass the previous works from the point of view of simplicity and efficiency, but there is still room of improvement.

Some of the possible follow-ups of this line of work can be: finding weaker (knowledge) assumptions to rely on, or further search for assumptions that provide quantum-security, extend the computation to something more expressive than just arithmetic functions and explore the possibility of removing the random oracle model requirement, by conserving the zero-knowledge property.

Overall, we believe that this is a promising starting point in achieving efficient solutions that address both the problems of data privacy and security of delegated computations.

Bibliography

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May 2010.

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. LNCS, pages 3–33. Springer, Heidelberg, December 2017.

[ACFP14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. http://eprint.iacr.org/2014/1018.

[ACNP16] Michel Abdalla, Mario Cornejo, Anca Nitulescu, and David Pointcheval. Robust password-protected secret sharing. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 61–79. Springer, Heidelberg, September 2016.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136. Springer, Heidelberg, February 2007.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[AFK87] Martín Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle (extended abstract). In Alfred Aho, editor, *19th ACM STOC*, pages 195–203. ACM Press, May 1987.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Heidelberg, March 2009.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 17*, pages 2087–2104. ACM Press, 2017.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *STOC*, pages 99–108. ACM, 1996.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, May 2017.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Heidelberg, August 2014.

[APS15a] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. http://eprint.iacr.org/2015/046.

[APS15b] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13. IEEE Computer Society Press, October 1992.

[AV77] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison, editors, *STOC*, pages 30–41. ACM, 1977.

[Bab85] László Babai. Trading group theory for randomness. In *17th ACM STOC*, pages 421–429. ACM Press, May 1985.

[Ban95] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices inr n. *Discrete & Computational Geometry*, 13(2):217–231, 1995.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 669–699. Springer, 2018.

[BBFR15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. In *2015 IEEE Symposium on Security and Privacy*, pages 271–286. IEEE Computer Society Press, May 2015.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, October 1988.

[BCC+14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580, 2014. http://eprint.iacr.org/2014/580.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In

Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why "Fiat-Shamir for proofs" lacks a proof. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 182–201. Springer, Heidelberg, March 2013.

[BDP00] Joan Boyar, Ivan Damgård, and René Peralta. Short non-interactive cryptographic proofs. *Journal of Cryptology*, 13(4):449–472, 2000.

[Ben16] Fabrice Ben Hamouda–Guichoux. *Diverse modules and zero-knowledge*. Theses, PSL Research University, July 2016.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Heidelberg, May 2011.

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 1–16. Springer, Heidelberg, March 2011.

[BFKW08] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. Cryptology ePrint Archive, Report 2008/316, 2008. http://eprint.iacr.org/2008/316.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.

[BGLS02] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. Cryptology ePrint Archive, Report 2002/175, 2002. http://eprint.iacr.org/2002/175.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BHZ87] Ravi B. Boppana, Johan Hastad, and Stathis Zachos. Does co-np have short interactive proofs? *Information Processing Letters*, 25(2):127 – 132, 1987.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal snargs via linear multi-prover interactive proofs. Cryptology ePrint Archive, Report 2018/133, 2018. https://eprint.iacr.org/2018/133.

[BLS11] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. On the correct use of the negation map in the pollard rho method. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 128–146. Springer, Heidelberg, March 2011.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *44th FOCS*, pages 384–393. IEEE Computer Society Press, October 2003.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd FOCS*, pages 112–117. IEEE Computer Society Press, November 1982.

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. http://eprint.iacr.org/2013/703.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 236–261. Springer, Heidelberg, November / December 2015.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In Shafi Goldwasser, editor, *ITCS 2012*, pages 350–366. ACM, January 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12. ACM, January 2014.

[CCH+18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-shamir from simpler assumptions. *IACR Cryptology ePrint Archive*, 2018:1004, 2018.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. Cryptology ePrint Archive, Report 2018/131, 2018. https://eprint.iacr.org/2018/131.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 595–613. Springer, Heidelberg, March 2009.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic MACs for arithmetic circuits. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 336–352. Springer, Heidelberg, May 2013.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society Press, May 2015.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 371–389. Springer, Heidelberg, August 2014.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In *ASIACRYPT 2017, Part I*, LNCS, pages 377–408. Springer, Heidelberg, December 2017.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

[CL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *CiE*, volume 5028 of *Lecture Notes in Computer Science*, pages 175–185. Springer, 2008.

[CLW18] Ran Canetti, Alex Lombardi, and Daniel Wichs. Non-interactive zero knowledge and correlation intractability from circular-secure fhe. *IACR Cryptology ePrint Archive*, 2018:1248, 2018.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In Shafi Goldwasser, editor, *ITCS 2012*, pages 90–112. ACM, January 2012.

[Cou18] Geoffroy Couteau. *Zero-Knowledge Proofs for Secure Computation*. Theses, PSL research University, October 2018.

[CR72] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *STOC*, pages 73–80. ACM, 1972.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

[Dam93] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 341–355. Springer, Heidelberg, May 1993.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Heidelberg, May 2000.

[dB88] Bert den Boer. Diffie-hillman is as strong as discrete log for certain primes. In Shafi Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 530–539. Springer, 1988.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 54–74. Springer, Heidelberg, March 2012.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 361–381. Springer, Heidelberg, February 2010.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015.

[DMP90] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 269–282. Springer, Heidelberg, August 1990.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47(10):777–780, May 1935.

[Fei73] H. Feistel. *Cryptography and Computer Privacy*. Scientific American, 1973.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 844–855. ACM Press, November 2014.

[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*, pages 308–317. IEEE Computer Society Press, October 1990.

[FN16] Dario Fiore and Anca Nitulescu. On the (in)security of SNARKs in the presence of oracles. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 108–138. Springer, Heidelberg, October / November 2016.

[FNP18] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In *in review*, 2018.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge snarks. In *IACR International Workshop on Public Key Cryptography*, pages 315–347. Springer, 2018.

[Gaa96] Jostein Gaarder. *Sophie's World: A Novel about the History of Philosophy*. Berkley Books, München, 1996.

[Gal13] Steven D. Galbraith. Space-efficient variants of cryptosystems based on learning with errors. preprint, 2013. https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin,

editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205 – 214, 1998.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 230–240. Tsinghua University Press, January 2010.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier CS-Proofs. Cryptology ePrint Archive, Report 2011/456, 2011. http://eprint.iacr.org/2011/456.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. LNCS, pages 581–612. Springer, Heidelberg, 2017.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM Conference on Computer and Communications Security*, pages 556–573. ACM, 2018.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[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*, pages 174–187. IEEE Computer Society Press, October 1986.

[GMY82] Shafi Goldwasser, Silvio Micali, and Andrew Chi-Chih Yao. On signatures and authentication. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 211–215. Plenum Press, New York, USA, 1982.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[Gol95] Oded Goldreich. Foundations of cryptography (fragments of a book). Electronic Colloquium on Computational Complexity, 1995.

[Gol08] Oded Goldreich. Probabilistic proof systems: A primer. *Foundations and Trends® in Theoretical Computer Science*, 3(1):1–91, 2008.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432, 2007. http://eprint.iacr.org/2007/432.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 803–815. Springer, Heidelberg, July 2005.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155, 2007. http://eprint.iacr.org/2007/155.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[Gt12] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. http://gmplib.org/.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *computational complexity*, 11(1-2):1–53, 2002.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320. Springer, Heidelberg, December 2013.

[HAO15] Ryo Hiromasa, Masayuki Abe, and Tatsuaki Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 699–715. Springer, Heidelberg, March / April 2015.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HJP13] W. Hart, F. Johansson, and S. Pancratz. FLINT: Fast Library for Number Theory, 2013. Version 2.4.0, http://flintlib.org.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 3–34. Springer, Heidelberg, April 2015.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions. Cryptology ePrint Archive, Report 2018/385, 2018. https://eprint.iacr.org/2018/385.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[HPS08] J. Hoffstein, J. Pipher, and J.H. Silverman. *An Introduction to Mathematical Cryptography*. Undergraduate Texts in Mathematics. Springer, New York, NJ, USA, 2008.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, Heidelberg, August 1998.

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 575–594. Springer, Heidelberg, February 2007.

[Jab01] David P. Jablon. Password authentication using multiple servers. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 344–360. Springer, Heidelberg, April 2001.

[Kah67] David Kahn. *The Codebreakers*. The Macmillan Company, New York, 1967. xvi + 1164 pages.

[Ker83] A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, pages 161–191, 1883.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[KR01] Joe Kilian and Phillip Rogaway. How to protect des against exhaustive key search (an analysis of desx). *J. Cryptology*, 14(1):17–35, 2001.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. LNCS, pages 224–251. Springer, Heidelberg, 2017.

[KTX07] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 315–329. Springer, Heidelberg, April 2007.

[KW93] M. Karchmer and A. Wigderson. On span programs. In IEEE Computer Society Press, editor, *In Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, Gaithersburg, MD, USA, 1993. IEEE Computer Society Press.

[KW18] Sam Kim and David J. Wu. Multi-theorem preprocessing nizks from lattices. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 733–765. Springer, 2018.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC 2005*, volume 3650 of *LNCS*, pages 314–328. Springer, Heidelberg, September 2005.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013.

[Lip16] Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 185–206. Springer, Heidelberg, April 2016.

[LLNW18] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 700–732. Springer, 2018.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[LPR10a] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[LPR10b] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 58–76. Springer, Heidelberg, May 2014.

[Mau94] Ueli M. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete algorithms. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 271–281. Springer, 1994.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*, Vol. 42, No. 44.:114–116, 1978.

[Mer79] Ralph Charles Merkle. *Secrecy, authentication and public key systems.* PhD thesis, Stanford University, June 1979.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 2002.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[Mic10] Daniele Micciancio. A first glimpse of cryptography's holy grail. *Commun. ACM*, 53(3):96, 2010.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[Nao03a] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[Nao03b] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[Pei08] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. Cryptology ePrint Archive, Report 2008/481, 2008. http://eprint.iacr.org/2008/481.

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. http://eprint.iacr.org/2015/939.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

[Pol78] John M. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 422–439. Springer, Heidelberg, March 2012.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PVW07] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. Cryptology ePrint Archive, Report 2007/348, 2007. http://eprint.iacr.org/2007/348.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[RAD78] Ron Rivest, Len Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–177, 1978.

[Reg05a] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. pages 84–93, 2005.

[Reg05b] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. http://eprint.iacr.org/2010/137.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[Sha71] Daniel Shanks. Class number, a theory of factorization, and genera. In *Analytic Number Theory*, volume 20 of *Proceedings of Symposia in Pure Mathematics*, pages 415–440. American Mathematical Society, 1971.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[Sho02] Victor Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394. Springer, Heidelberg, December 2010.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and

David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, Heidelberg, May 2010.

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). In *STOC*, pages 196–203. ACM, 1976.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, May 2010.

[vdHBPW18] Jeroen van den Hoven, Martijn Blaauw, Wolter Pieters, and Martijn Warnier. Privacy and information technology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2018 edition, 2018.

[Vee17] Meilof Veeningen. Pinocchio-based adaptive zk-SNARKs and secure/correct adaptive function evaluation. In *AFRICACRYPT 17*, LNCS, pages 21–39. Springer, Heidelberg, 2017.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, Heidelberg, July 2005.

[Wee07] Hoeteck Wee. Lower bounds for non-interactive zero-knowledge. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 103–117. Springer, Heidelberg, February 2007.

[Wes67] Alan F. Westin. *Privacy and Freedom*. Atheneum, New York, 1967.

[WJB+17] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *ACM CCS 17*, pages 2071–2086. ACM Press, 2017.

[WTS+18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, 2018.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880. IEEE Computer Society Press, 2017.

RÉSUMÉ

Cette thèse est consacrée à une exploration des schémas de preuve de connaissance succincts, les SNARKs. S'inscrivant dans un contexte marqué par le développement du Cloud et des technologies Blockchain, les SNARKs sont des primitives cryptographiques permettant la vérification de l'intégrité des calculs.
Dans un modèle de type client-serveur, où un client à faible puissance de calcul délègue une tache à un serveur à forte puissance de calcul, les SNARKs lui permettent de verifier efficacement si le serveur a bien exécuté la tache demandée.
Notre attention se porte en particulier sur des sujets comme la sécurité post-quantique des SNARKs, la propriété d'extractabilité, qui fait du SNARK un outil si puissant dans des protocoles cryptographiques, la composition de ces preuves avec d'autres primitives cryptographiques et la construction d'un protocole cryptographique basé sur des preuves SNARKs qui garantit non seulement l'intégrité du résultat, mais aussi la confidentialité des données représentant l'entrée du calcul à vérifier.

MOTS CLÉS

SNARK, post-quantique, Cloud, intégrité des calculs

ABSTRACT

The contributions detailed in this thesis focus on the design and the analysis of Succinct non-interactive arguments of knowledge, known as SNARKs.
SNARKs enable a party with large computational resources to prove to a weaker party that a particular statement is true in an efficient way without further interaction and under a minimal communication requirement.
Our results deal with three different aspects of SNARK protocols: the post-quantum security of SNARKs, the composability of SNARKs with other cryptographic primitives and the confidentiality of the inputs in the computations verified by SNARKs.
First, we propose a new framework that allows the instantiation of a quantum-resilient SNARK scheme from lattice assumptions. We also study the notion of extractability that is part of the soundness definition for SNARKs. We remark some limitations of this definition and we address
this problem, by introducing and studying a new notion, O-SNARKs.
Finally, to achieve data privacy in delegated computation, we study the possibility of constructing SNARKs that enables verification of computations over encrypted data.

KEYWORDS

SNARK, post-quantum, Cloud, secure verifiable computation