Zero-knowledge arguments from secure multiparty computation

Jules Maire

Zero-Knowledge Arguments from Secure Multiparty Computation

PhD THESIS IN COMPUTER SCIENCE

Presented by

Jules Maire

To obtain the degree of

DOCTOR of SORBONNE UNIVERSITÉ

Supervised by

Damien Vergnaud

Publicly defended on October 11, 2024

Jury:

Carsten Baum	Associate Professor, Technical University of Denmark (reviewer)
Geoffroy Couteau	CNRS Researcher, Institut de Recherche en Informatique Fondamentale
Jean-Marc Couveignes	Professor, Université de Bordeaux (president)
Philippe Gaborit	Professor, Université de Limoges
Emmanuela Orsini	Assistant Professor, Bocconi University
David Pointcheval	Senior CNRS Researcher, Ecole Normale Supérieure (reviewer)
Adeline Roux-Langlois	CNRS Researcher, Université de Caen
Damien Vergnaud	Professor, Sorbonne Université

ABSTRACT

This thesis aims to study zero-knowledge arguments, a cryptographic primitive that allows to prove a statement while yielding nothing beyond its truth (we may call it proof instead of argument depending on the security model). Specifically, we focus on a family of arguments whose construction is based on secure multiparty computation. It is well-known that, given any functionality, there exists a secure multiparty protocol computing it. Let us take a generic one-way function f, and a secure multiparty protocol computing f, then it has been shown seventeen years ago that we can build a zero-knowledge argument for the $\mathcal{NP}$-problem of finding a pre-image of f. This construction was considered only theoretical until a few years ago, and this thesis contributes to the emergence of new techniques as well as efficient applications.

As an appetizer, we develop simple zero-knowledge protocols that significantly improve the state-of-the-art communication complexity for some well-known problems. Our first substantial contribution, with a desire to share small elements over large fields, is the introduction of a sharing over the integers that is securely embedded in our protocols with some artificial abortion. Applications are manifold, eventually in the post-quantum regime. In the line with our sharing over the integers, we propose a cryptographic string commitment scheme based on subset sum problems. In particular, it enables efficient arguments for circuit satisfiability. Then, we present a proof construction employing conversion between additive and multiplicative secret sharings, leading to efficient proofs of linear and multiplicative relations. The applications are again manifold when designing arguments and digital signatures. Finally, leaving aside protocols conception, we explore cryptography foundations with multi-prover zero-knowledge proofs, a framework for distributing the prover's computation of interactive zero-knowledge proofs. To capture the full interest of this dispatching, we add to the literature a fundamental result for threshold zero-knowledge proofs for generic $\mathcal{NP}$-statement.

ACKNOWLEDGEMENTS

In the 4th-5th century AD, in Book XI of his *Confessions*, Saint Augustin wrote the following: "What then is time? If no one asks me, I know; if I want to answer this question, I don't know". Although far removed from Augustine's reflections on the theme of time, I draw an analogy with what I experienced during my thesis. These three years of research have made me aware of one primordial thing: scientific research is a perpetual questioning of one's knowledge, and *certainty without reflection and questioning is only illusion.* In this respect for having challenged me and made me conscious of this adage, I would like to thank the researchers I came across during my studies at EPFL: Philippe Michel, Clément Hongler, at CISPA: Antoine Joux, and during my thesis: Damien Vergnaud.

These last years have been a journey I'm grateful to have shared with all the Almasty team as well as friends associated to the team, with all the cryptographers I have ever met from near or far, and with all my friends and family. A special mention to Damien Vergnaud for his kindness and always sound advice. Discussions with you were always very rewarding, and your many innovative ideas and wide-ranging knowledge have been enriching in many ways.

CONTENTS

1. CRYPTOGRAPHY

The Dictionnaire de l'Académie française defines cryptography as *l'art d'écrire en langage codé, secret, chiffré*, or in english, *the art of writing in coded, secret or encrypted language*. Although this definition was satisfactory for a long time, it does not reflect the core of modern cryptography. Since the end of the 20th century, cryptography has become a more rigorous science relying on a richer theory, encompassing many other aspects than this definition. Especially, researchers started to establish mathematical definitions of security for cryptographic schemes, then to design schemes hoping that they will meet the security models previous defined, and finally to ask whether the designs meet the security definitions. While designing algorithm performing specific tasks in a context where users running the algorithm trustworthy could be relatively easy, security models may require the design of schemes in an untrusted adversarial environment. This brings us to the necessity of the notion of *proofs* lying at the heart of cryptography: if we do not trust an adversary (and so her statement), it is natural to ask for a proof that the statement is true. However, proofs in mathematics are inherently reproducible in the sense that once one has verified a proof, one gets the ability to present the same proof to others and convince them to the statement. In other words, one learns more than just the validity of the proof and of the truthiness of the statement. Hence, we should look at a major conceptual expansion of the notion of proof.

1.1 Zero-Knowledge Proofs

Many cryptographic applications need to handle with a class of proofs that do reveal nothing else than the validity of the statement being proven. This stronger notion may seem counterintuitive at first glance, and many people were skeptical when Goldwasser, Micali, and Rackoff in [GMR89] offered an innovative idea by introducing randomization and interaction to achieve this goal. They viewed a proof as an interactive protocol between two parties, a prover and a verifier, who communicate with each other. In this context, proofs can be probabilistic (*i.e.*, both parties can flip coins) and interactive (*i.e.*, the verifier may ask questions to the prover). More precisely, the questions from the verifier are random challenges, and in order to convince the verifier of the truthfulness of the statement, the prover sends respective responses. At the end, the verifier should be convinced (with high probability) that the statement is true. Amazingly, it is possible to guarantee that the verifier learns essentially nothing else from the interaction. This leads to a major class of proofs and a powerful tool in the construction of cryptographic protocols: *zero-knowledge proof* systems (ZKP). On the way to a formal definition, a zero-knowledge proof should satisfy:

(i) correctness: if the statement is true, and the prover knows a proof of it, they will succeed in convincing the verifier;

(ii) soundness: if the statement is false, no prover can convince the verifier of the truth of the statement, except with small probability;

(iii) zero-knowledge: the interaction yields nothing beyond the fact that the statement is true.

From a security perspective, the zero-knowledge property aims to protect the prover against the verifier: their private input should not leak in any way. While the soundness protects the verifier against a malicious prover who does not know a valid proof.

ZKP have found numerous applications in cryptography, notably for privacy-preserving schemes or to enforce honest behaviour of users during protocols. Moreover, we may use the term of zero-knowledge *argument* (ZKA) when the security model assumes that the (potentially malicious) prover is computationally bounded, leading to a weaker notion of ZKP.

A central primitive in cryptography are commitment schemes [Blu82]. It is a cryptographic protocol that enables one party to commit to a value without revealing it, while ensuring that this value cannot be modified. In constructing sophisticated cryptographic protocols, it can be necessary to prove properties of a committed message without revealing anything else than the validity of these properties. This is usually achieved through the use of ZKP.

1.2 $\mathcal{P}$ versus $\mathcal{NP}$ Problem

In theoretical computer science, the complexity class $\mathcal{NP}$ refers to the problems for which solutions can be verified efficiently. In other words, $\mathcal{NP}$ can be seen as the set of all statements that can be proven efficiently (without considering

how hard it might be to find the proof). This makes $\mathcal{NP}$ an object of deep philosophical interest, as the famous and fundamental question of whether $\mathcal{NP}$ is contained in $\mathcal{P}$ essentially amounts to asking whether proofs are computationally harder to find than they are to verify.

Thus, the class of problems $\mathcal{NP}$ is of utmost importance for this thesis, since ZKP/ZKA should be built on top of these problems. Moreover, an answer to the question "$\mathcal{P}$ vs. $\mathcal{NP}$" would have a tremendous impact, because if $\mathcal{NP}$ collapsed to $\mathcal{P}$, then this would prove that one-way functions do not exist (a function that is easy to compute on every input, but hard to invert given the image of a random input). That in turn would imply that almost no secure cryptographic primitives could then be *proven secure* (according to the chosen definitions of security), although it does not mean that all these schemes would be broken in practice.

Furthermore, Goldreich, Micali and Wigderson [GMW91] proved that every $\mathcal{NP}$ problem has a ZKP (assuming secure bit commitment).

1.3 Provable Security

Although a cryptographic scheme may have some security properties (*e.g.* privacy, authenticity, etc.), we have not yet discussed how to prove that such a scheme achieves them. Traditionally, the most common method to gain confidence in the security of a scheme was to look for attacks and conjecture it is secure if no attacks were found that undermined its security. However, this approach has a significant limitation: we can never be sure that an attack does not exist. Consequently, the security can only be considered heuristic at best, as the possibility of an undiscovered attack cannot be ruled out.

Another method for proving the security of a cryptographic scheme, known as *provable security*, consists of linking the scheme's security with the security of its underlying primitives or computational problems. To accomplish this, one must first define the adversary's capabilities and the security goals that the scheme must achieve. E.g., in a ZKA, the malicious prover (who plays the role of the adversary here) is assumed to be computationally bounded by the security model. Then, a reduction must be provided, demonstrating how an adversary that compromises the security goals of the scheme can be transformed into an adversary against the security goals of the underlying primitives or computational problems on which the scheme relies. Hence, one direct benefit of provable security is that it eliminates the need to search for specific attacks against a scheme (as long as we trust that the underlying primitives are secure or the computational problems are difficult)

Post-quantum cryptography. In 1994, Shor [Sho94] introduced a quantum algorithm that could break some computational problems on which the security of most of the schemes of the time were based (via security reduction). More precisely, his new attacks could break cryptosystems based on the hardness of factoring large integers or solving discrete logarithm problems. This has emphasized the need for new cryptographic systems, leading to the emergence of a new field, known as *post-quantum cryptography*, which focuses on creating cryptographic algorithms that are proven secure against quantum (and classical) computers.

1.4 Secure Multiparty Computation

For many cryptographic applications as well as for theoretical aspects, we may consider a set of parties who want to perform a collaborative computation using their private inputs. As security guarantees, it seems natural to require the computation to satisfy *correctness* of the output and *privacy* of the inputs. This conjoint computation is termed *secure multiparty computation* (MPC). If they could all agree on a trusted third party, parties could delegate the computation to this party: everyone would send their inputs to the trusted party and then receive back the result. However, the existence of such a third party is often unrealistic. MPC aims to provide a protocol that the parties can execute themselves without needing a trusted intermediary, and a first approach was introduced by Yao [Yao86] with garbled circuits.

A plethora of security models can be considered in which we want to prove the security of our MPC protocols, *i.e.*, in which we wish our protocols to be correct and private. Let us focus on a security model tolerating the presence of an internal adversary (not necessarily computationally bounded) who can corrupt a subset of the parties, and learn their private inputs and outputs. This adversary may either be *passive* (it has only access to their view of the protocol execution), or *active* (it can make them deviate from the protocol). Therefore, the security notion we aim to achieve is against an adversary, who corrupts a subset of the parties, gains access to their view, and may control them (in the active setting). This adversary should learn nothing about the other parties' inputs (*i.e.* honest parties) beyond what can be inferred from the corrupted parties' inputs and outputs.

MPC-in-the-Head. In 2007, Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS07] demonstrated that MPC secure in the passive setting is sufficient for constructing zero-knowledge protocols. This theoretical paradigm, deemed *MPC-in-the-Head* (MPCitH), has received considerable practical attention recently since it offers an elegant approach to constructing efficient and succinct ZKP with good security properties. In particular, it has been used to propose a series of innovative signature schemes with alleged post-quantum security.

We previously mentioned that every $\mathcal{NP}$ problem has a ZKP. Essentially, MPCitH offers a framework for building such a ZKP, by leveraging MPC techniques. In a nutshell, let us consider some one-way function f (it defines an $\mathcal{NP}$-language). Then the MPCitH paradigm can be summary as follows:

(i) The prover mentally shares its secret witness (for the corresponding $\mathcal{NP}$-statement) and emulates a passively secure MPC protocol that computes f. Then the prover independently commits each party's view and sends these commitments to the verifier.

(ii) The verifier challenges the prover to reveal the views of a strict subset of parties.

(iii) Once the verifier receives these parties' views, they can check that the partial execution of the MPC protocol for the revealed parties is consistent (as well as the consistency of the commitments), and thus can accept or reject the proof accordingly.

The zero-knowledge property of the obtained ZKP relies on the privacy of the MPC protocol. The completeness and the soundness of the ZKP are based on the correctness of the MPC protocol. In particular for the soundness, a malicious prover (that does not know a valid witness) would need to deceive at least one party to cheat, which the verifier is likely to detect if the MPC is correct.

Therefore, the security of the obtained ZKP only relies on the hardness of the $\mathcal{NP}$-problem (assuming a secure commitment scheme and pseudo-random generator). Hence, a post-quantum secure $\mathcal{NP}$-problem implies a post-quantum secure ZKP.

1.5 Organization of this Thesis

In Chapter 2, we introduce the technical background with all the cryptographic building blocks met along this thesis. In Chapter 3, we put forward the state-of-the-art of the MPC-in-the-Head paradigm with recent optimizations and flourished variants. Especially, we bring to the fore two simple contributions [MV23b] as appetizer and by way of educative example. The following chapters contain the heart of the matter. Chapter 4 deals with sharing over the integers for the MPCitH setting with many applications [FMRV22, MV23a]. Chapter 5 addresses a conversion sharing technique for the MPCitH paradigm, again with efficient protocols for different hard problems [MV23b]. Finally, Chapter 6 proposes a new paradigm for building generic threshold proofs.

1.6 Our Results

Zero-Knowledge Protocols with Sharing over the Integers [FMRV22]

In Chapter 4, we propose zero-knowledge arguments for the modular *subset sum problem*. Given a set of integers, this problem asks whether a subset adds up to a target integer t modulo a given integer q. This NP-complete problem is considered since the 1980s as an interesting alternative in cryptography to hardness assumptions based on number theory and it is in particular believed to provide post-quantum security. Previous combinatorial approaches, notably one due to Shamir, yield arguments with cubic communication complexity (in the security parameter). More recent methods, based on the *MPC-in-the-Head* technique, also produce arguments with cubic communication complexity.

We improve this approach by using a secret-sharing over small integers (rather than modulo q) to reduce the size of the arguments and remove the prime modulus restriction. Since this sharing may reveal information on the secret subset, we introduce the idea of *rejection* to the MPC-in-the-head paradigm. Special care has to be taken to balance completeness and soundness and preserve zero-knowledge of our arguments. We combine this idea with two techniques to prove that the secret vector (which selects the subset) is well-formed of binary coordinates. Our new techniques have the significant advantage to result in arguments of size *independent* of the modulus q.

Our new protocols for the subset sum problem achieve an asymptotic improvement by producing arguments of quadratic size (against cubic size for previous proposals). This improvement is also practical: for a 256-bit modulus q, the best variant of our protocols yields 13KB arguments while previous proposals gave 1180KB arguments, for the best general protocol, and 122KB, for the best protocol restricted to prime modulus. Our techniques can also be applied to vectorial variants of the subset sum problem and in particular the *inhomogeneous short integer solution* (ISIS) problem for which they provide an efficient alternative to state-of-the-art protocols when the underlying ring is not small and NTT-friendly. We also show the application of our protocol to build efficient zero-knowledge arguments of plaintext and/or key knowledge in the context of *fully-homomorphic encryption*. When applied to the TFHE scheme, the obtained arguments are more than 20 times smaller than those obtained with previous protocols. Eventually, we use our technique to construct an efficient digital signature scheme based on a pseudo-random function due to Boneh-Halevi-Howgrave-Graham.

Commitments with Efficient ZK Arguments [MV23a]

A second part of the Chapter 4 presents a cryptographic string commitment scheme that is computationally hiding and binding based on (modular) subset sum problems (hence that is believed to provide post-quantum security). Using techniques developed along Chapter 4, this simple commitment scheme enables an efficient zero-knowledge proof of knowledge for committed values as well as proofs showing Boolean relations amongst the committed bits. In particular, one can prove that committed bits $m_0, m_1, \ldots, m_\ell$ satisfy $m_0 = C(m_1, \ldots, m_\ell)$ for any Boolean circuit C (without revealing any information on those bits). The proof system achieves good communication and computational complexity since for a security parameter λ, the protocol's communication complexity is $\tilde{O}(|C|\lambda + \lambda^2)$ (compared to $\tilde{O}(|C|\lambda^2)$ for the best code-based protocol due to Jain, Krenn, Pietrzak and Tentes).

Zero-Knowledge Arguments via Sharing Conversion [MV23b]

Chapter 5 deals with a novel technique within the MPC-in-the-Head framework, aiming to design efficient zero-knowledge protocols and digital signature schemes. The technique allows for the simultaneous use of additive and multiplicative sharings of secret information, enabling efficient proofs of linear and multiplicative relations. The applications of our technique are manifold. It is first applied to construct zero-knowledge arguments for Double Discrete Logarithms (DDLP) (which turns out to be less efficient than our other ZKP for the DDLP presented in Chapter 3). The resulting protocol achieves improved communication complexity without compromising efficiency compared to the state-of-the-art. We also propose a new zero-knowledge argument of knowledge for the Permuted Kernel Problem. Eventually, we suggest a short (candidate) post-quantum digital signature scheme constructed from a new one-way function based on simple polynomials known as fewnomials. This scheme offers simplicity and ease of implementation.

Threshold Zero-Knowledge Proofs

In Chapter 6, we explore multi-prover zero-knowledge proofs that distribute the prover's computation in interactive zero-knowledge protocols. To capture the full interest of this dispatching, we focus on threshold proofs, where any large enough subset of provers is able to conjointly convince the verifier. Our work falls within the scope of auditable MPC but stands out compared to previous works by being robust, adaptable to threshold settings, post-quantum secure, and generic. Hence, it helps bridge the gap in threshold cryptography by providing an additional tool for verifiable computation with shared inputs. Concretely, our approach is based on MPC and allows any sufficiently large subset of provers to produce a valid proof, ensuring robust security in the active setting. We then propose a black-box construction that emulates MPC protocols within the provers' minds to verify computations. To address practical concerns, we develop a non-signaling variant for low-depth arithmetic circuits. Its versatility and simplicity offers a new approach for effective, post-quantum threshold protocols, leveraging recent advancements in the MPC-in-the-Head world.

1.7 Our Other Contribution

Secure Multi-Party Linear Algebra with Perfect Correctness [MV24]

This works presents new secure multi-party computation protocols for linear algebra over a finite field, which improve the state-of-the-art in terms of security. We look at the case of *unconditional security with perfect correctness*, *i.e.*, information-theoretic security without errors. We notably propose an expected constant-round protocol for solving systems of m linear equations in n variables over $\mathbb{F}_q$ with expected complexity $O(k(n^{2.5} + m^{2.5} + n^2 m^{0.5}))$ where $k > m(m+n)+1$ (complexity is measured in terms of the number of secure multiplications required). The previous proposals were not error-free: known protocols can indeed fail and thus reveal information with probability $\Omega(\text{poly}(m)/q)$. Our protocols are simple and rely on existing computer-algebra techniques, notably the Preparata-Sarwate algorithm, a simple but poorly known "baby-step giant-step" method for computing the characteristic polynomial of a matrix, and techniques due to Mulmuley for error-free linear algebra in positive characteristic.

PERSONAL PUBLICATIONS

[FMRV22] T. Feneuil, J. Maire, M. Rivain, and D. Vergnaud. Zero-knowledge protocols for the subset sum problem from mpc-in-the-head with rejection. In S. Agrawal and D. Lin, eds, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, vol. 13792 of *Lecture Notes in Computer Science*, p. 371–402. Springer, 2022.

[MV23a] J. Maire and D. Vergnaud. Commitments with efficient zero-knowledge arguments from subset sum problems. In G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, eds, *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*, vol. 14344 of *Lecture Notes in Computer Science*, p. 189–208. Springer, 2023.

[MV23b] J. Maire and D. Vergnaud. Efficient zero-knowledge arguments and digital signatures via sharing conversion in the head. In G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, eds, *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*, vol. 14344 of *Lecture Notes in Computer Science*, p. 435–454. Springer, 2023.

[MV24] J. Maire and D. Vergnaud. Secure multi-party linear algebra with perfect correctness. *IACR Commun. Cryptol.*, 1(1):29, 2024.

2. PRELIMINARIES

2.1 Notations

Let $\mathbb{F}_q$ be the finite field with q elements for a prime power q, and let $\mathbb{F}$ denotes any finite field. For the sake of simplicity, we denote the set of integers $\{m, \dots, n\}$ by $[m, n]$ with $m < n$. The transpose operator is written as $\cdot^T$. All logarithms are in base 2. Given $m, n \in \mathbb{N}$, we may write $m = \text{poly}(n)$ when there exists some $c \in \mathbb{N}$ such that $m = O(n^c)$. Pr denotes a probability function.

For computationally secure algorithms encountered in this thesis, λ denotes the security parameter and is given to them as input in the unary form 1^λ. Algorithms may be written with a special font A, and A^{RO} refers to an algorithm with black-box access to a random oracle RO, and is called a *random-oracle algorithm.* Unless otherwise stated, algorithms are randomized, and PPT stands for "probabilistic polynomial-time" in the security parameter. Random sampling from a finite set X according to the uniform distribution is denoted by $x \xleftarrow{\$} X$. The symbol $\xleftarrow{\$}$ is also used for assignments from randomized algorithms, and the symbol $\leftarrow$ is used for assignments from deterministic algorithms and calculations.

We might denote integer vectors in bold print. Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x}\|\boldsymbol{y}$ denotes their concatenation, and if they have the same length, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes their inner-product, and $\boldsymbol{x} \circ \boldsymbol{y}$ denotes the component-wise product.

2.2 Cryptographic Primitives

Security criteria. A cryptographic scheme is said to be *information-theoretic* secure, if it is secure against a computationally unbounded adversary (in terms of time and storage resources). While a scheme whose security depends on the computational capabilities of the adversary, and that can be broken by a computationally unbounded adversary, is called *computationally* secure.

Definition 1 (Negligible function)**.** *A function $\nu(\cdot)$ is negligible if it is asymptotically smaller than the inverse of any polynomial: for every constant $c \in \mathbb{R}_{>0}$ and all sufficiently large $n \in \mathbb{N}$, we have $\nu(n) \leq 1/n^c$.*

2.2.1 Cryptographic hash functions

Cryptographic hash functions are a major building block for primitives like zero-knowledge proofs, digital signatures, message authentication code, etc.

A hash algorithm is any function that maps a message of arbitrary length to a fixed-length message digest. A *cryptographic hash function $\mathcal{H}$* is a hash algorithm that should be *collision resistance:* It is computationally infeasible to find two different inputs $x \neq x'$ such that $\mathcal{H}(x) = \mathcal{H}(x')$.

Definition 2 (Collision-resistant hash functions)**.** *Let $m, \ell, \mu : \mathbb{N} \to \mathbb{N}$ be some functions. A family of functions $\{\mathcal{H}_k : \{0,1\}^{m(k)} \to \{0,1\}^{\ell(k)}\}_{k \in \{0,1\}^{\mu(\lambda)}}$ for a security parameter λ, is a family of collision-resistant hash functions if $|m(k)| > |\ell(k)|$, every $\mathcal{H}_k$ can be computed within polynomial time given k, but there exists a negligible function ν such that, for any PPT algorithm A, we have*

$$\Pr[x_1 \neq x_2, \mathcal{H}_k(x_1) = \mathcal{H}_k(x_2) \mid (x_1, x_2) \leftarrow \mathcal{A}(k, 1^\lambda), k \leftarrow \{0,1\}^{\ell(\lambda)}] < \nu(\lambda).$$

2.2.2 Pseudo-random generator

A *pseudo-random generator* (PRG) is an algorithm that takes as input a *seed*, which generates a bitstring containing more bits than the seed, and whose distribution is indistinguishable from the distribution of a random string. This indistinguishable condition amounts to showing that there exists no efficient algorithm which can distinguish the PRG output from true randomness, as soon as the seed is randomly selected.

Definition 3 (Indistinguishable distributions)**.** *Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0,1]$ be two functions, with ϵ a negligible function. Two distributions $\{D_\lambda\}_\lambda$ and $\{\tilde{D}_\lambda\}_\lambda$ are deemed (t, ϵ)-indistinguishable if, for any algorithm A running in time at most $t(\lambda)$, we have*

$$|\Pr[\mathcal{A}(1^\lambda, x) = 1 \mid x \xleftarrow{\$} D_\lambda] - \Pr[\mathcal{A}(1^\lambda, x) = 1 \mid x \xleftarrow{\$} \tilde{D}_\lambda]| \leq \epsilon(\lambda).$$

Definition 4 (Pseudo-random generator)**.** *Given an additional function $\ell : \mathbb{N} \to \mathbb{N}$, a (ℓ, t, ϵ)-*pseudo-random generator *is a deterministic algorithm G that, for all $\lambda \in \mathbb{N}$, on input a bit-string $x \in \{0,1\}^\lambda$ outputs $G(x) \in \{0,1\}^{\ell(\lambda)}$ such that*

*(i) $\ell(\lambda) > \lambda$ (*expansion*);*

*(ii) the distributions $\{G(x) \mid x \xleftarrow{\$} \{0,1\}^\lambda\}$ and $\{r \mid r \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}\}$ are (t, ϵ)-indistinguishable (*pseudorandomness*).*

Randomness in cryptography is fundamental because it aims to hide secrets that have no reason to be random. Moreover, it ensures, for example, that cryptographic keys are truly random. Without true randomness, cryptographic systems may become vulnerable in an adversarial environment. However, surprising as it may seem, producing true randomness is very expensive. Indeed, it is impossible to generate truly random numbers from deterministic machine like computers, except if we rely on physical phenomena, hence producing such a large quantity of randomness is costly. A solution would be to rely on PRG to generate such a large quantity of random numbers using a computer, since it is well-known for a long time that PRG is equivalent to one-way functions [HILL99], and once we have produced a seed using true randomness, we can call to the PRG with this seed.

2.2.3 Commitment schemes

A commitment scheme [Blu82] is a fundamental primitive in cryptography, since it enables a committer party to lock a value inside a metaphorical sealed letter, ensuring that no one can distinguish the actual value (hiding property), while simultaneously preventing this committer from opening the commitment in more than one way (binding property).

Definition 5 (Commitment scheme)**.** *A commitment scheme is a triple of algorithms (Setup, Com, Ver) such that:*

- *Setup$(1^\lambda) \to pp$. On input λ, the* setup *PPT algorithm outputs the public parameters pp containing a description of the message space $\mathcal{M}$.*
- *Com$(pp, m) \to (c, aux)$. On input pp and $m \in \mathcal{M}$, the* commit *PPT algorithm outputs a commitment-opening pair (c, aux).*
- *Ver$(pp, m, c, aux) \to b \in \{0,1\}$. On input pp, $m \in \mathcal{M}$ and (c, aux), the* verifying *(or* opening*) deterministic polynomial-time algorithm outputs a bit $b \in \{0,1\}$.*

Moreover, it satisfies the following correctness property: for all $\lambda \in \mathbb{N}$,

$$\Pr[\textit{Ver}(pp, m, c, aux) = 1 \mid pp \xleftarrow{\$} \textit{Setup}(1^\lambda), m \xleftarrow{\$} \mathcal{M}, (c, aux) \xleftarrow{\$} \textit{Com}(pp, m)] = 1.$$

Eventually, the public parameter input pp will be made implicit in the calls to Com, and Ver. There are two security notions underlying a commitment scheme.

Definition 6. *Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0,1]$ be two functions, with ϵ a negligible function. A commitment scheme (Setup, Com, Ver) is said:*

- (t, ϵ)-computationally hiding *if, for any two messages m_1, m_2, the distributions $\{c \mid c \xleftarrow{\$} \mathrm{Com}(m_1)\}$ and $\{c \mid c \xleftarrow{\$} \mathrm{Com}(m_2)\}$ are (t, ϵ)-indistinguishable, i.e., if for all two-phases algorithm $A = (A_1, A_2)$, we have for all $\lambda \in \mathbb{N}$:*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} pp \xleftarrow{\$} \textit{Setup}(1^\lambda), (m_0, m_1, s) \xleftarrow{\$} A_1(pp), b \xleftarrow{\$} \{0,1\} \\ (c, aux) \xleftarrow{\$} \textit{Com}(pp, m_b), b' \xleftarrow{\$} A_2(c, s) \end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda)$$

  when A runs in time at most $t(\lambda)$ in this probabilistic computational game.

- (t, ϵ)-computationally binding *if for all PPT algorithm A, we have for all $\lambda \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \\ \textit{Ver}(pp, m_1, c, aux_1) = 1 \\ \textit{Ver}(pp, m_2, c, aux_2) = 1 \end{array} \;\middle|\; \begin{array}{l} pp \xleftarrow{\$} \textit{Setup}(1^\lambda), \\ (m_1, m_2, aux_1, aux_2, c) \xleftarrow{\$} A(1^\lambda, pp)\} \end{array}\right] \leq \epsilon(\lambda)$$

  when A runs in time at most $t(\lambda)$ in this probabilistic computational game.

In constructing sophisticated cryptographic protocols, it can be necessary to prove some property of a committed message without revealing anything more than the property itself. This is usually achieved through the use of zero-knowledge proofs of knowledge [GMR89]. This *commit-and-prove* paradigm [Kil89, CLOS02] is used in many areas of applied cryptography (anonymous credentials, electronic voting, etc.).

2.3 Zero-Knowledge Proofs

Previous cryptographic primitives met in this thesis were non-interactive algorithms, but can be used as building blocks for more complex interactive protocols. Informally, the goal of an interactive proof is to convince someone that a certain statement is true. The ability to prove such statements is very useful in many cryptographic applications. These proofs were introduced in a series of papers in the 1980s, culminating in the seminal work of Goldwasser, Micali and Rackoff [GMR89] that also defined the notion of *zero-knowledge proofs*, proofs where the verifier entity essentially learns nothing else than the validity of the proof.

Languages. A language $\mathcal{L}$ for a binary relation $\mathcal{R}$ is a set of words in $\{0,1\}^*$ defined as:

$$\mathcal{L} = \{x \in \{0,1\}^* \mid \exists w \in \{0,1\}^*, \mathcal{R}(x,w) = 1\}.$$

Given $x \in \{0,1\}^*$, "$x \in \mathcal{L}$" is called a *statement*, and if $\mathcal{R}(x,w) = 1$ (we may equivalently write $(x,w) \in \mathcal{R}$), then w is said to be a *witness* for this statement.

Complexity class $\mathcal{NP}$. The class $\mathcal{NP}$ is defined by all the statements for the truth of which a short proof exists, and can be verified in reasonable time. In more specific terms, the proof size and the verification running time have to be bounded by a polynomial in the size of the statement.

We could define the notion of interactive proofs for languages not in $\mathcal{NP}$, but then the result from [GMW91] −demonstrating that every $\mathcal{NP}$-problem has a zero-knowledge proof− does not hold anymore.

Definition 7 (Interactive proofs of knowledge)**.** *Let $\mathcal{L}$ be any $\mathcal{NP}$-language with binary relation $\mathcal{R}$. Let us consider two algorithms, a prover P and a verifier V, such that P is probabilistic but not computationally bounded, while V is a PPT algorithm. Both algorithms are given a common input statement x, and P is given an additional witness w for "$x \in \mathcal{L}$". The two algorithms exchange poly$(|x|)$ messages until V outputs a bit b ($b = 1$ to accept P's claim and $b = 0$ to reject). This sequence of messages and the answer b is referred to as a* transcript *and is denoted as $\langle P(x,w), V(x)\rangle$.*

Let us consider two functions $\epsilon, \alpha : \mathbb{N} \to [0,1]$, then $\langle P(x,w), V(x)\rangle$ is an interactive proof of knowledge *(PoK) for the statement "$x \in \mathcal{L}$" with* soundness error ϵ *and* α-completeness *if:*

- α-completeness*: given $(x,w) \in \mathcal{R}$, then for all $\lambda \in \mathbb{N}$,*

  $$\Pr[\langle P(x,w), V(x)\rangle = 1] \geq 1 - \alpha(\lambda)\ .$$

  In other words, for a true statement, P succeeds in convincing V except with probability α. When $\alpha = 1$, the proof achieves perfect completeness.

- ϵ-(special) soundness*: for all computationally unbounded algorithm $\tilde{P}$ such that for all $\lambda \in \mathbb{N}$ and all $x \in \{0,1\}^*$,*

  $$\tilde{\epsilon}(\lambda) := \Pr[\langle \tilde{P}(x), \mathcal{V}(x)\rangle = 1] > \epsilon(\lambda)\ ,$$

  there exists a PPT algorithm E (called extractor*) which, given rewindable black-box access to $\tilde{P}$ outputs a witness w such that $(x,w) \in \mathcal{R}$ in time poly$(\lambda, (\tilde{\epsilon} - \epsilon)^{-1})$ with probability at least $1/2$.*

Remark 1.

1. *The soundness property ensures that the algorithm $\tilde{P}$ without knowledge of a valid witness, cannot convince V with probability greater than ϵ (this includes cases where the statement is false). Otherwise, the existence of E implies that $\tilde{P}$ can use it to compute a valid witness w for $x \in \mathcal{L}$. In particular, the extractor E is assumed to be PPT in the formalism where a request to an oracle (here to $\tilde{P}$) takes a constant time.*

2. *We could provide a definition without extractor for the soundness, resulting in a* proof of membership*, where at the end of the interactions, V is only convinced of the existence of witness for the statement "$x \in \mathcal{L}$". Whereas the formalism of the soundness with an efficient extractor capable of recovering the witness yields to a stronger notion of soundness known as either* special-soundness *or* knowledge-soundness*, and additionally demonstrates that P knows the witness (it can be necessary for applications where authentication is required).*

3. *If P was assumed computationally bounded (therefore PPT), it would result in the weaker notion of interactive* argument of knowledge*.*

4. *The number of exchange messages is in poly$(|x|)$ to get an efficient proof.*

5. *In the soundness property, we assume that E has a rewindable oracle access to $\tilde{P}$, and this may lead to proofs that are not straight-line extractable and not tight.*

These PoK can have the additional property of *zero-knowledge*, captured by requiring the existence of a simulator algorithm that does not take the secret witness for the statement as input and that can produce a transcript for the interactive protocol that is closely distributed to a real transcript, by having a black-box access to the verifier.

Definition 8 (Zero-knowledge proof of knowledge). *Let $\mathcal{L}$ be any $\mathcal{NP}$-language with binary relation $\mathcal{R}$, and let $t : \mathbb{N} \to \mathbb{N}$, $\zeta : \mathbb{N} \to [0,1]$ be two functions. Consider a PoK between P and V for the statement "$x \in \mathcal{L}$", with the same notations and assumptions as in Definition 7. Then, $\langle P(x,w), V(x)\rangle$ is a* zero-knowledge proof of knowledge *(ZKPoK) for the statement "$x \in \mathcal{L}$" with* (t,ζ)-zero-knowledge *if:*

- (t,ζ)-zero-knowledge*: for every PPT algorithm $\tilde{V}$, there exists a PPT algorithm Sim (called* simulator*) which, given the input statement x and rewindable black-box access to $\tilde{V}$, outputs a simulated transcript whose distribution is (t,ζ)-indistinguishable from $\langle P(x,w), \tilde{V}(x)\rangle$.*

Remark 2.
1. *To achieve a targeted soundness error, we will perform parallel executions of the protocol. Such parallel repetitions do not preserve (general) zero-knowledge and the resulting scheme only achieves zero-knowledge property for a genuine verifier. The protocol is then deemed* special honest-verifier zero-knowledge*. In that case, Sim is given random challenges instead of a rewindable black-box access to the verifier (since a genuine verifier is supposed to choose their challenges randomly in the corresponding set of challenges). Hence, for the time being, one only considers special honest-verifier zero-knowledge proofs.*
2. *We may denote by ZKP any zero-knowledge proof of membership (not necessarily a proof of knowledge), and by ZKA any zero-knowledge argument of membership.*

2.3.1 Splitting lemma

In our security proofs for the soundness, we shall make use of the following lemma from [PS00]:

Lemma 1 (Splitting lemma). *Let X and Y be two finite sets, and let $A \subseteq X \times Y$ such that*

$$\Pr\left[(x,y) \in A \mid (x,y) \xleftarrow{\$} X \times Y\right] \geq \varepsilon\,.$$

For any $\alpha \in [0,1)$, let

$$B = \left\{(x,y) \in X \times Y \;\middle|\; \Pr\left[(x,y') \in A \mid y' \xleftarrow{\$} Y\right] \geq (1-\alpha)\varepsilon\right\}.$$

Then one has:

(i) $\Pr\left[(x,y) \in B \mid (x,y) \xleftarrow{\$} X \times Y\right] \geq \alpha\varepsilon$

(ii) $\Pr\left[(x,y) \in B \mid (x,y) \xleftarrow{\$} A\right] \geq \alpha\,.$

2.3.2 Non-interactive proofs

So far, we have encountered only interactive proofs, but there are several methods to convert an interactive proof into a non-interactive one. One widely used technique is the Fiat-Shamir heuristic [FS87], which transforms Σ-protocols (a class of zero-knowledge protocols with public-coin properties) into non-interactive zero-knowledge proofs (NIZK). This transformation is particularly useful for creating digital signatures in the random oracle model.

The key idea behind the Fiat-Shamir heuristic is to replace the interactive verifier's random challenges with a deterministic process: instead of the verifier issuing random challenges, the prover generates these challenges by hashing certain public values from the protocol. This hash function serves as a "random oracle", simulating the randomness needed for the challenge. The prover computes the hash using public information from the interaction, which replaces the need for a back-and-forth exchange with the verifier. By doing so, the proof becomes non-interactive, as the verifier no longer needs to actively participate in generating the challenge. This technique has wide applications, particularly in constructing digital signatures, as it allows proofs and verifications to be conducted in a single, non-interactive step, reducing complexity and making the process more efficient for real-world use.

2.4 Secure Multiparty Computation

Multiparty computation considers the scenario where a set of parties wish to perform a collaborative computation of some function. The aim of *secure multiparty computation* (MPC) is to allow parties to execute such distributed computing with some security guarantees. Applications range from privacy-preserving schemes to threshold cryptography, and more. Consequently, MPC has been a highly researched topic since its introduction in 1986 by Yao for the two-party case [Yao82], and by Goldreich, Micali and Wigderson for the multiparty case [GMW87]. Several paradigms have been developed through the years for designing MPC protocols. *Garbled circuits* allow the parties to build an encrypted version of the circuit that can be computed only once, thus requiring few communication rounds but typically demanding high bandwidth. *Homomorphic encryption* techniques, which are especially suitable for highly distributed computations, although they can impose a substantial computational burden. Finally, a very popular method—and the technique that we will focus on in this thesis—is *linear secret-sharing schemes*. This approach allows parties to maintain distributed computation throughout the process until the final output is achieved.

2.4.1 Secret sharing schemes

A central building block in distributed computation are secret sharing schemes. It allows a dealer to distribute a secret to n users in such a way that any large enough set of users can jointly reconstruct the secret from their shares, whereas any small subset of users can't derive any information on the secret. We call this scheme linear if any linear combination of valid shares results in a valid share of the linear combination of the respective secrets. We focus on information-theoretic secure schemes.

Definition 9 (Linear secret sharing schemes)**.** *A linear secret-sharing scheme (t, n)-LSSS consists of three polynomial-time algorithms:*

- *LSSS.Setup outputs the system parameters pp including descriptions of: a finite field $\mathbb{F}$, two vector spaces $\mathbb{V}_1$ (secret space) and $\mathbb{V}_2$ (share space) over $\mathbb{F}$, the number of users n, and the privacy/reconstruction threshold t.*
- *LSSS.Share : $\mathbb{V}_1 \times R \to \mathbb{V}_2^n$. On input the secret $s \in \mathbb{V}_1$ and some randomness from a randomness space $R \subset \{0,1\}^*$, outputs n shares $(\llbracket s \rrbracket_i)_{i\in[1,n]} := \llbracket s \rrbracket \in \mathbb{V}_2^n$. We may informally write LSSS.Share(s) by keeping silent on the randomness.*
- *LSSS.Reconstruct : $\mathbb{V}_2^{|I|} \to \mathbb{V}_1$. On input a set of shares $(\llbracket s \rrbracket_i)_{i\in I} \in \mathbb{V}_2^{|I|}$ (with a set of users $I \subseteq [1, n]$) and a description of the set I, outputs a secret $s \in \mathbb{V}_1$ or $\perp$ denoting failure.*

Moreover, an LSSS should satisfy the following properties:

(i) t-reconstructability: For any $pp \leftarrow$ LSSS.Setup, any secret $s \in \mathbb{V}_1$, any sharing $(\llbracket s \rrbracket_i)_{i\in[1,n]} \leftarrow$ LSSS.Share(s), and any subset $I \subseteq [1, n]$ with $|I| \geq t + 1$, LSSS.Reconstruct$((\llbracket s \rrbracket_i)_{i\in I}) = s$.

(ii) t-privacy: For any $pp \leftarrow$ LSSS.Setup, any secret $s \in \mathbb{V}_1$, any $(\llbracket s \rrbracket_i)_{i\in[1,n]} \leftarrow$ LSSS.Share(s), and any subset $I \subset [1, n]$ with $|I| \leq t$, there exists a simulator Sim such that the distributions of the output of Sim(I) and $(\llbracket s \rrbracket_i)_{i\in I}$ generated by a real execution of LSSS.Share(s) are identical.

(iii) $\mathbb{F}$-linearity: For every $s_1, s_2 \in \mathbb{V}_1, \alpha \in \mathbb{F}$, the two random variables defined by

$$\text{LSSS.Share}(s_1 + \alpha s_2) \text{ and } \text{LSSS.Share}(s_1) + \alpha\,\text{LSSS.Share}(s_2)$$

are equal in distribution over the randomness space R.

The previous definition is the tight version of a more general class of schemes called secret sharing ramp schemes, when there is a non-trivial gap between the privacy and the reconstruction threshold.

We present two common LSSS that this thesis simultaneously brings into play.
A *modular additive (resp. multiplicative)* $(n-1, n)$*-LSSS* of a field element $s \in \mathbb{F}$ (resp. $s \in \mathbb{F}^\times$) is a vector $\llbracket s \rrbracket = (\llbracket s \rrbracket_1, \ldots, \llbracket s \rrbracket_N) \in \mathbb{F}^n$ (resp. $\langle s \rangle = (\langle s \rangle_1, \ldots, \langle s \rangle_n) \in \mathbb{F}^{\times n}$) such that

$$\mathsf{LSSS.Reconstruct}(\{\llbracket s \rrbracket_i\}_{i=1}^n) = \sum_{i=1}^n \llbracket s \rrbracket_i = s$$

(resp. $\mathsf{LSSS.Reconstruct}(\{\llbracket s \rrbracket_i\}_{i=1}^n) = \prod_{i=1}^n \langle s \rangle_i = s$).
A *Shamir secret sharing* of $s \in \mathbb{F}$ of degree $t < n$ is a (t, n)-LSSS such that $\llbracket s \rrbracket = (\llbracket s \rrbracket_i := P(\alpha_i))_{i\in[1,n]}$, where $\{\alpha_i\}_{i\in[1,n]} \subset \mathbb{F}^n$ are public distinct non-zero points, and $P(x) \in \mathbb{F}[x]$ is a degree t polynomial with constant term $P(0) = s$. Without loss of generality, we can take $\{\alpha_i\}_{i\in[1,n]} = [1, n]$ for the rest of this thesis. Note that we get a constraint on the field size $|\mathbb{F}| > n$.

LSSS and error correcting codes. A linear code of length $n+1$ and dimension $t+1$ is a linear vector subspace $\mathcal{C} \subseteq \mathbb{F}^{n+1}$ of dimension $t + 1$. Each vector in $\mathcal{C}$ is called a *code word* of $\mathcal{C}$. The distance between two codewords is the Hamming distance d_H between them, and the distance d of a linear code $\mathcal{C}$ is defines as $d = \min_{a,b\in\mathcal{C},a\neq b} d_H(a, b)$. A linear code of length $n + 1$, dimension $t + 1$, and distance d is called a $[n + 1, t + 1, d]$-code.

The BCH codes [BR60] is a class of linear codes with a lower bound on the error detection ratio. Among those codes, the MDS codes (for *maximum distance separable*) are optimal in the number of redundancies required for a fixed error detection ratio. In other words, they achieve the singleton bound $d = n - t + 1$. McEliece and Sarwate [MS81] were the first to observe a connection between Shamir's secret-sharing scheme and a class of codes named Reed-Solomon code. It is well known that generalization holds: (t, n)-LSSS are equivalent to $[n + 1, t + 1, n - t + 1]$ MDS codes.

Theorem 1 ([MS81])**.** *Let us consider some (t, n)-LSSS and let $(\llbracket s \rrbracket_1, \ldots, \llbracket s \rrbracket_n) \leftarrow$ LSSS.Share(s) be some sharing of $s \in \mathbb{F}$. Then, for any subset of at least $t + 1 + 2e$ shares with at most e of these values being incorrect, the secret s can be recovered correctly.*

2.4.2 Secure multiparty computation

Secure multiparty computation enables a group of said n potentially distrustful users $\mathsf{P}_1, \ldots, \mathsf{P}_n$, to jointly compute an n-ary functionality f of their public and private inputs. All this while preserving (1) the *privacy* ensuring that a user learns nothing about the other user's private input, and (2) the *correctness* of the computation, *i.e.*, every user computes an output that is distributed according to f. Users may communicate with each other, and any constraints on these communications are captured within these three models:

- *General model:* Users can communicate with each other via synchronous communication over secure point-to-point channels, and over an authenticated broadcast channel;
- *Broadcasting model:* Users can only synchronously communicate via an authenticated broadcast channel;
- *Isolated model:* Users are not allowed to communicate with each other.

When communication takes place in the protocol, computation occurs in different *rounds*, during which each user can send one flow of values to each other user (it seizes the broadcast). Subsequently, the definition of an MPC protocol can be formulated, relying on its *next-message function*.

Definition 10 (MPC protocol)**.** *Let $(n, r) \in \mathbb{N}^2$ with $r = poly(\lambda)$, and let f be a n-ary functionality. A r-round n-party MPC protocol Π_f that computes f in the* general model *is a tuple of PPT algorithms $\Pi_f = (\Pi_f^1, \ldots, \Pi_f^r, Output_f)$, such that for each $i \in [1, n]$:*

- *$\Pi_f^1(1^\lambda, i, x, w_i, \rho_i)$ returns the messages broadcast/sent by the i-th user to other users during the first round, on public input x, private input w_i, and random tape ρ_i.*
- *For each $j \in [2, r]$, $\Pi_f^j(1^\lambda, i, x, w_i, \rho_i, (m_i^1, \ldots, m_i^{j-1}))$ returns the messages broadcast/sent by the i-th user to other users during the round j, on public input x, private input w_i, random tape ρ_i, and received messages $(m_i^1, \ldots, m_i^{j-1})$ from the previous rounds.*
- *$Output_f(1^\lambda, i, x, w_i, \rho_i, (m_i^1, \ldots, m_i^r))$ returns the output of the i-th user.*

Moreover, we define the view of the i-th user by $View_i := \{x, w_i, \rho_i, (m_i^1, \ldots, m_i^r)\}$ (the sent messages can be re-computed via these informations).

Remark 3. *Definition 10 has been presented in the general model, but it can be directly converted in the broadcasting model (where the exchanged messages are only broadcasts). Protocols in the isolated model have no round, thus $\Pi_f = Output_f(1^\lambda, i, x, w_i, \rho_i)$ straightly returns the output of the i-th user.*

Security model. We examine these protocols within two security models, contingent upon the context. We tolerate the presence of adversarial behavior (that may be throughout an adversary algorithm Adv) that may compromise the computation as well as the privacy of the inputs. Hence, once the model accepts some adversarial behavior, the security properties should hold: the computation must remain correct and the users' input must remain uncover. The *passive* model refers to a scenario where the users are assumed to follow the protocol (and therefore cannot obstruct the computation), but might attempt to gather information about other users' inputs by observing the computation. On the contrary, the *active* model tolerates the presence of a malicious adversary who may corrupt a subset of the users (called *corrupted* users as opposed to *honest* users). Corrupted users may behave arbitrarily by deviating from the protocol, sending incorrect information, or colluding with other corrupted users to execute their attacks. For the sake of simplicity, the adversary is assumed to be *static*, the set of corrupted users is determined before the protocol execution begins. This is in contrast to the *adaptive/dynamic* scenario, where the adversary can choose to corrupt throughout the protocol execution. Moreover, we stress that the adversary is *rushing*, *i.e.*, during each round of communication it can see the messages sent by the honest users before it determines the messages sent by the corrupted users.

Security definitions in the passive model.

In the passive model, the security is split between correctness and privacy, as defined below.

Definition 11 (Perfect/statistic correctness)**.** *We say that an n-party MPC protocol Π_f computes an n-ary functionality f with statistical correctness if, for all tuple of (public and private) inputs, the probability that the output of some user is different from the output of f is negligible in λ, where the probability is over the independent choices of the random tapes of the users. If the probability is 0, the correctness is said perfect.*

Definition 12 (Perfect/statistic/computational t-privacy)**.** *Let us fix $(t, n) \in \mathbb{N}^2$ with $t < n$, let f be an n-ary functionality, and let Π be an n-party MPC protocol. We say that Π is t-private for f if there exists a PPT algorithm Sim such that for every index subset of corrupted users $I \subset [1, n]$ of cardinality at most t, every public input x, and private inputs $(w_1, \ldots, w_n)$, the joint view $View_I(x, w_1, \ldots, w_n)$ of users in I has a distribution that is perfectly/statistically/computationally close to the distribution of $Sim(x, I, \{w_i\}_{i \in I}, f_I(x, w_1, \ldots, w_n))$, where f_I is the joint computation of users in I.*

Security definitions in the active model.

The active setting comes in two variants, *with abort* or with *robust security*. We focus on the second variant. This informally means that corrupted users should not be able to prevent honest users from realizing the protocol and getting valid outputs, whatever the behavior of the corrupted users. This is captured with the *robustness* property, a stronger notion of correctness.

Definition 13 (Perfect/statistic t-robustness)**.** *Let us fix $(t, n) \in \mathbb{N}^2$ with $t < n$. An n-party MPC protocol Π_f computes an n-ary functionality f with perfect (resp. statistic) t-robustness if:*

(i) Π_f is perfectly (resp. statistically) correct in the passive model according to Definition 11;

(ii) For any computationally unbounded adversary corrupting a set of at most t users, for any public input x and private inputs $w_1, \ldots, w_n$, the probability that the output of Π_f for an honest user in the active model is inconsistent with their output of Π_f over inputs $x, w_1, \ldots, w_n$ in the passive model is 0 (resp. negligible in λ), where the probability is over the random tapes of the users.

The privacy property is formalized with the *real world-ideal world* paradigm by comparing a real protocol execution to an ideal model, *i.e.*, a protocol is said to be private if a real execution can be simulated in the ideal model where the users just send their inputs to some trusted party computing the function and receive back outputs. That is, for every real model adversary algorithm Adv, there exists an ideal model adversary algorithm Sim that has oracle access to Adv and externally interacts with the trusted party, such that the distribution of a real execution of the protocol with Adv is close to the distribution of an ideal execution with Sim. For the next definition, we need to introduce two random variables. Let Π be a n-party MPC protocol, Adv an arbitrary algorithm with auxiliary input $z \in \{0,1\}^*$, and $I \subset [1, n]$ the set of corrupted users controlled by Adv. We denote by $\text{REAL}_{\Pi,\mathsf{Adv}(z),I}(x_1, \ldots, x_n)$ the random variable consisting of the view of Adv and the outputs of the honest users following a real execution of Π, where x_i is the input of P_i for $i \in [1, n]$. We denote by $\text{IDEAL}_{F,\mathsf{Sim}(z),I}(x_1, \ldots, x_n)$ the random variable consisting of the outputs of the ideal adversary Sim (controlling the corrupted users in I) and of the honest users after an ideal execution with a trusted party computing the functionality F (*i.e.* users send inputs to the trusted party computing F and receive back outputs) upon inputs $x_1, \ldots, x_n$ for the users and auxiliary input z for Sim. We stress that the communication between the trusted party and users is over an ideal private channel. The next security definition bears the same name as in passive security, however the context will clarify which definition should be used throughout the remainder of the thesis.

Definition 14 (Perfect/statistic/computational t-privacy)**.** *Let $(t, n) \in \mathbb{N}^2$ with $t < n \in \mathbb{N}$, let F be an n-ary functionality, and let Π be an n-party protocol. We say that Π is t-private for F if, for every probabilistic adversary Adv in the real model, there exists a probabilistic ideal adversary Sim having oracle access to Adv, and running in polynomial time in the running time of Adv, such that for every $I \subset [1, n]$ of cardinality at most t, public input x, private inputs $(w_1, \ldots, w_n)$, and auxiliary input z, the distribution of $\text{IDEAL}_{F,\mathsf{Sim}(z),I}(x, w_1, \ldots, w_n)$ is perfectly/statistically/computationally close to the distribution of $\text{REAL}_{\Pi,\mathsf{Adv}(z),I}(x, w_1, \ldots, w_n)$, where the probability is over the random tapes of the users, Sim, and Adv.*

The condition on the running time of the ideal adversary Sim is set to guarantee that information-theoretic security implies computational security.

Reactive functionalities.

Many protocols in this thesis call for modular subprotocols, making the analysis of the overall security complex, as standard security definitions typically apply to functionalities with a single stage of computation. To formalize these security requirements for modular composition (*i.e.* functionalities that invoke subfunctionalities), our MPC protocols must ensure that an adversary corrupting a subset of users cannot achieve more than they could by attacking an idealized protocol where subfunctionalities are computed by an external trusted party. In this ideal model, users send their inputs to an incorruptible trusted party, who computes the subfunctionality's output for them. More formally, let Π_f be a protocol for securely computing a functionality f that calls a subprotocol Π_g for computing a functionality g. We rely on the composition theorem [Gol04] which expedites a modular security analysis: we start by proving the security of Π_g, and then proving the security of Π_f in a model allowing a trusted party to ideally compute g (instead of the users running the real subprotocol Π_g). This model is called the *g-hybrid model* (because it involves both a real protocol execution and an ideal trusted party computing g). As an example, it is fairly common to prove the security of a protocol involving commitments in a *commitment-hybrid* model.

3. THE MPC-IN-THE-HEAD PARADIGM

This chapter introduces the original MPC-in-the-Head (MPCitH) paradigm, its developments, optimizations and variants. It is illustrated with our first contributions.

In 2007, Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS07] demonstrated that passive secure multiparty computation is sufficient for constructing zero-knowledge protocols. This theoretical paradigm, deemed *MPC-in-the-Head*, was at first stood in the realm of theoretical cryptography (with a focus on the asymptotic performance for any problem in $\mathcal{NP}$), but it was subsequently demonstrated to be also of practical relevance [GMO16, KKW18], and has found numerous applications (*e.g.* [BN20a, GHM+22]).

An MPCitH proof is built on an MPC protocol which relies on some secret-sharing scheme. In the seminal paper [IKOS07], the modular additive secret sharing was considered, and it remained so until very recently. Then a series of works brought diversity, [FJR23] shared the secret permutation as a composition of permutations over the symmetric group, [FMRV22] used an additive secret sharing over the integers, and [MV23b] the modular multiplicative secret sharing. Finally, the use of threshold linear secret sharing schemes has been proposed in such a way that it can be considered as a reunification effort, and is named *Threshold Computation-in-the-Head* (TCitH) [FR23a]. A concurrent work introduced a rather different paradigm based on Verifiable Oblivious Linear Evaluation named *VOLE-in-the-Head* (VOLEitH) [BBdSG+23], and which turns out to be a special case of TCitH.

3.1 The Framework

Let $x \in \mathcal{L}(\mathcal{R})$ be an $\mathcal{NP}$ statement with witness w, and let f be a binary functionality such that

$$f(x,w) = \begin{cases} 1 & \text{if } (x,w) \in \mathcal{R} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Theorem 2 (MPCitH [IKOS07])**.** *Let f be the functionality defined by equation (3.1), and let Π_f be an N-party MPC protocol realizing f in the passive setting (with $N \geq 3$), with perfect/statistic correctness and perfect/statistic/computational $(N-1)$-privacy. Then we can build a zero-knowledge proof of knowledge for the statement "$x \in \mathcal{L}$", achieving perfect/statistic completeness, perfect/statistic/computational zero-knowledge, and $(1/N + \delta(\lambda))$-soundness in the commitment-hybrid model, where $\delta(\cdot)$ is some negligible function.*

Originally introduced for the modular additive secret sharing, the MPCitH framework can be generalized to any (T,N)-LSSS, yielding to the next theorem, whose result collapses to Theorem 2 when $T = N-1$.

Theorem 3 (TCitH [FR23a])**.** *Let f be the functionality defined by equation (3.1), and let Π_f be an $T+1$-party MPC protocol realizing f in the passive setting (with $N \geq 3$), with perfect/statistic correctness and perfect/statistic/computational T-privacy for some $T < N$. Then we can build a zero-knowledge proof of knowledge for "$x \in \mathcal{L}$", achieving perfect/statistic completeness, perfect/statistic/computational zero-knowledge, and $\left(1/\binom{N}{T} + \delta(\lambda)\right)$-soundness in the commitment-hybrid model, where $\delta(\cdot)$ is some negligible function.*

3.1.1 Construction of the interactive proof

We consider a prover P and a verifier V engaging a 2-party interactive protocol with the goal of convincing V that P knows a valid witness w for $x \in \mathcal{L}$.

1. P first generates a random (T,N)-LSSS of w. Assume that Π_f, realizing the functionality f defined by equation (3.1), is a $(T+1)$-party MPC protocol. Then P chooses a subset $A \subset [1,N]$ of size $|A| = T+1$, and mentally simulates the computation of Π_f in the broadcasting model 3 for parties in A. P sends commitments of each party's view in the protocol (including input share, secret random tape, and broadcast values).

2. V randomly selects a subset $B \subset [1,N]$ of the parties of size $|B| = T$ and requests P to reveal their views.

3. P, on top of revealing these views, also sends the broadcacts of some party in $A \setminus B$.

Upon receiving them, V checks that these views are consistent with the MPC process as well as valid openings of the commitments. At first sight, presented as such, this interactive protocol seems to have 3-rounds. However, in most of

the cases, additional rounds will take place with extra challenges from V to be convinced. These challenges are given to the emulated parties, and for this purpose, let us detail the computation of Π_f in the broadcasting model, assuming it has r rounds. At each round $j \in [1, r]$ in $\Pi_f = (\Pi_f^1, \ldots, \Pi_f^r, \mathsf{output}_f)$, parties take as input: the public $\mathcal{NP}$-statement x, the private sharing of w, the random tape, broadcasts from the first $(j-1)$-th rounds, and the challenges received so far by P. Then, parties perform three types of actions during each round (on the inputs that we have mentioned):

(i) Receiving randomness: The parties may receive the same random value(s). This simulates additional challenges sent from V to P.

(ii) Receiving hint: The parties may receive some sharing generated by P (required for the MPC emulation, usually for proving multiplicative relations).

(iii) Computation in the broadcasting model: The parties may locally compute a linear function with some sharings as input, where the function depends on challenges and previous broadcast values. Then they may broadcast their shares and recover the function evaluation.

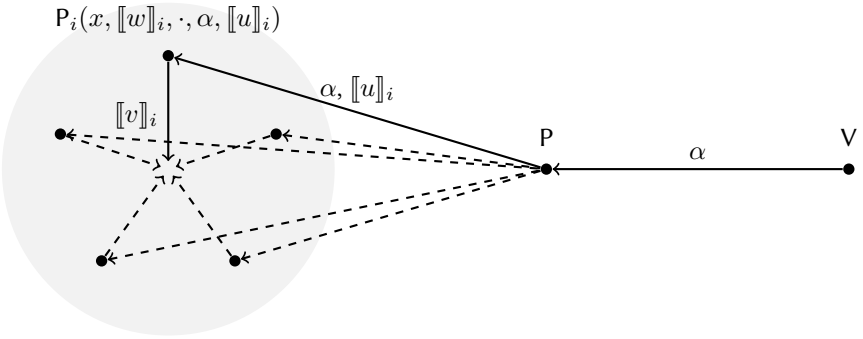


Fig. 3.1: Informal overview of party P_i computation during one round: the challenge α corresponds to type (i) action, the new sharing $[\![u]\!]$ to (ii), and $[\![v]\!]_i = g(\cdot, \alpha, [\![u]\!]_i)$ to (iii), where g is a linear function. The $\cdot$ may be substituted by previous broadcasts, and challenges.

Once all the parties have broadcasted their computation (*i.e.* $[\![v]\!]$), they can recover the broadcast value v. At this stage, on top of $[\![w]\!]$, $[\![u]\!]$ must be added to the content to be committed by P. If P wants to reveal all the parties' view except for P_{i^*} (when $T = N-1$), they should send, on top of $\{[\![w]\!], [\![u]\!]\}_{i \neq i^*}$, the broadcast value $[\![v]\!]_{i^*}$ from P_{i^*}. Hence, V is able to partially execute the MPC for all the parties $\{\mathsf{P}_i\}_{i \neq i^*}$, and V can get the output v.

3.1.2 Security and complexity analysis

Security analysis. The *completeness* of the ZKP holds thanks to the correctness of Π_f: if P knows a valid witness w for $x \in \mathcal{L}$ and honestly simulates Π_f, then by the perfect (resp. statistic) correctness of Π_f, V accepts the proof (resp. except with negligible probability). The *zero-knowledge* holds thanks to the privacy of Π_f: since the views of only T parties are disclosed, this does not reveal any information about the secret w, as long as Π_f is T-private and w is shared with some (T, N)-LSSS. The *soundness* also holds thanks to the correctness of Π_f: a malicious prover $\tilde{\mathsf{P}}$ who does not know a valid witness and who simulates Π_f, may still convince V in two different ways:

(i) Either $\tilde{\mathsf{P}}$ cheats during the computation such that the outputs of Π_f is now a sharing of 1 (recall that f outputs a bit). In that case, the best strategy of such an adversary is to cheat for exactly $N - T$ parties. Indeed, they have to cheat for at least $N - T$ parties, and if they cheat on more than $N - T$ parties, V shall always identify the cheat (since V asks for the opening of T parties). Hence, a malicious prover must undoubtedly cheat on exactly $N - T$ parties, and the only way for V to be convinced is to ask for the opening of the exact T parties which have been honestly emulated. The probability of this event to happen is $1/\binom{N}{T}$.

(ii) Or the outputs of the MPC protocol is always a sharing of 0 except with small probability which is the false positive probability of Π_f, *i.e.* $p := \Pr[\Pi_f = 1 \mid (x, w) \notin \mathcal{R}]$.

The overall probability that a malicious prover manages to convince a verifier, *i.e.* the soundness error, is then

$$\epsilon_{serr} = \frac{1}{\binom{N}{T}} + \left(1 - \frac{1}{\binom{N}{T}}\right) p \,.$$

When $T = N - 1$, the soundness error collapses to $\epsilon_{serr} = \frac{1}{N} + \left(1 - \frac{1}{N}\right) p$.

Complexity analysis. The soundness error ϵ of one repetition of the protocol may not be sufficient to reach the target soundness of $2^{-\lambda}$. Hence, one can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. The computational complexity of P corresponds to the emulation of $\tau(T+1)$ parties and the complexity of V to τT parties. The communication complexity is the size of the proof transcript which is composed of all the commitments, and the opening of views. Essentially, an MPCitH-like proof can be decomposed into a symmetric part (randomness derivation, commitment computation), and an MPC modeling part that depends on the $\mathcal{NP}$-problem considered and on how one arithmetizes it. We discuss below optimizations for both parts.

3.2 Symmetric Optimizations

Let us focus on additive or multiplicative secret sharing schemes (thus $T = N - 1$), although there have been some optimizations for threshold LSSS (see [FR23a]).

3.2.1 GGM trees

To produce randomness for each party, P could generate one seed by party and then use some PRG (see Subsection 2.2.2). A typical way to additively share a value $w \in \mathbb{F}$ is, given a seed seed_i for each party P_i, to derive $[\![w]\!]_i \xleftarrow{\$} \mathrm{PRG}(\mathsf{seed}_i)$ in $\mathbb{F}$, and then to fix the sharing with some auxiliary value $\Delta w \in \mathbb{F}$ such that $w = \sum_{i=1}^{N} [\![w]\!]_i + \Delta w$. By doing so, for opening all-but-one parties, P has to send $N - 1$ seeds. This can be optimized by using some tree-structure when generating seeds [KKW18], this last work relying on a standard construction [GGM84], usually named the GGM construction. The idea is to use a tree PRG algorithm (named TreePRG) in which one uses a PRG with $\ell(\lambda) = 2\lambda$ (see Definition 4) to expand a λ-bit root seed (usually labeled mseed for master seed) into N λ-bit subseeds in a structured fashion as follows (in practice we take N to be a power-of-two, let us say $N = 2^t$). Let us consider a complete binary tree of depth t, with root labeled mseed, in which the right/left child of each node is labeled with the λ most/least significant bits of the output of the PRG applied to the root label. The subseeds $(\mathsf{seed}_i)_{i\in[1,N]}$ are defined as the labels of the N leaves of the tree. To reveal $N-1$ subseeds $(\mathsf{seed}_i)_{i\in[1,N]\setminus\{i^*\}}$, one reveals the siblings of all the nodes in the path from the punctured seed (seed_{i^*}) to the tree root as illustrated in Figure 3.2. These siblings are also called the co-path of the punctured seed. This co-path allows reconstructing $(\mathsf{seed}_i)_{i\neq i^*}$ while still hiding seed_{i^*}. Thus, the communication cost scales with $\log_2 N$. This can be easily generalized if we wish to reveal all the subseeds but a small subset $A \subset [1, N]$.

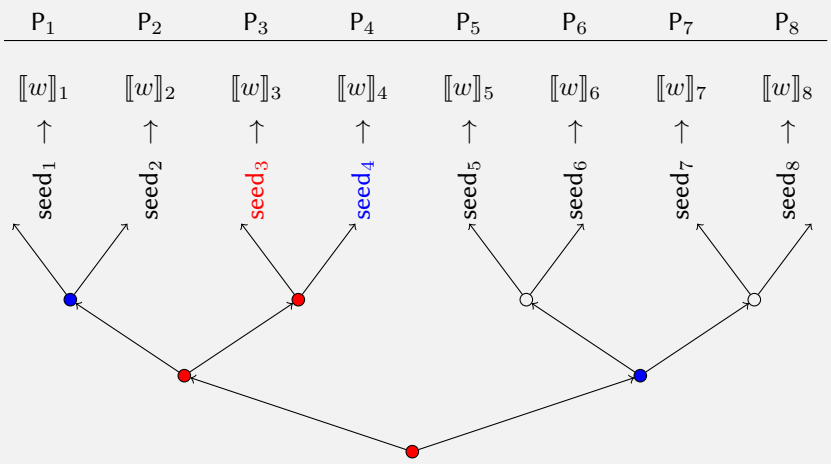


Fig. 3.2: Blue co-path of the punctured seed_3 with $(N = 8, i^* = 3)$.

3.2.2 Parallel optimization of GGM trees

We have seen that to shorten the communication, shares and random coins used in the protocol are generated using a TreePRG: P randomly and uniformly chooses a master seed mseed and constructs a tree of depth $\log N$ by expanding mseed into N subseeds as explained in the previous subsection.

For the interactive protocol to achieve the target soundness error, we repeat the protocol τ times such that $(\epsilon_{serr})^\tau < 2^{-\lambda}$. Following the previous GGM tree construction, P generates one GGM tree for each repetition, yielding to τ trees, each with N leaves. Instead of generating these τ independent GGM trees (in parallel), which cost at most $\tau\lambda\log_2(N)$ bits in the proof size when revealing the τ co-path of the τ punctured seeds, Baum et al. [BBM+24] derive a large GGM tree with τN leaves, which reduces with high probability the number of nodes that need to be sent. Let us consider that

the first τ leaves of the tree correspond to the seeds of each first emulated party of the τ repetitions, the next leaves correspond to the seeds of each second emulated party, and so on, hence following an interleaved fashion. Precisely, the i-th seed of the e-th repetition is associated to the $(eN + i)$-th leaf of the large GGM tree. Then, as explained in [BBM+24], "opening all but τ leaves of the big tree is more efficient than opening all but one leaf in each of the τ smaller trees, because with high probability some of the active paths (*i.e.* co-path) in the tree will merge relatively close to the leaves, which reduces the number of internal nodes that need to be revealed." However, the number of nodes to send depends on the distribution of the challenges for the opening views, thus it causes variability in the proof size. To get around the problem, the authors consider some threshold for the maximal number of nodes to send above which they abort the protocol. This optimization aims to save a few hundreds bits.

Merging the commitments A simple but saving optimization consists of merging some commitments before sending them. Indeed, to commit to each party input, instead of sending N commitments $\mathrm{Com}_1, \ldots, \mathrm{Com}_N$ of 2λ bits each, we can merge them by hashing all these commitments and then send $\mathcal{H}_1(\mathrm{Com}_1, \ldots, \mathrm{Com}_N)$. This implies to send the commitment Com_{i^*} to the unopened party with index i^* so that the hash value can be recomputed and checked by V. Let h_j be the hash value sent during the j-th iteration with $j \in [1, \tau]$, where τ is the number of parallel repetitions of the protocol. Instead of sending τ hash values $h_1, \ldots, h_\tau$, P can merge them together to send a single hash $h = \mathcal{H}_2(h_1, \ldots, h_\tau)$. This yields to a total of $(2 + \tau 2)\lambda$ bits (2λ bits for h and $\tau 2\lambda$ bits for the commitments Com_{i^*} to the unopened party), instead of naively $\tau N 2\lambda$ bits.

3.3 MPC Modeling Optimizations

3.3.1 Hypercube optimization

In [MGH+23], Aguilar-Melchor, Gama, Howe, Hülsing, Joseh, and Yue developed a geometrical approach for the MPC emulation phase. This optimization relies on the commutativity of the laws in finite fields. In the traditional approach of MPCitH, P simulates N parties (for each of the τ repetitions of the protocol). By this hypercubing approach, this number of parties can be reduced to $1 + \log_2 N$, with the same soundness error. Essentially, instead of simulating one MPC protocol with N parties, the prover emulates $\log_2 N$ MPC protocols with 2 parties.

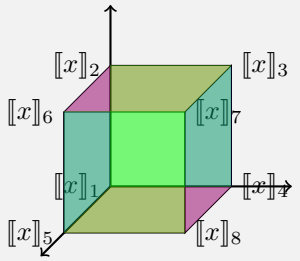


Fig. 3.3: Hypercube overview for $x = \sum_{j=1}^{8} [\![x]\!]_j$

In a nutshell, given $N = k^d$ parties, one can represent the N shares into a hypercube of dimension d and length k as in Figure 3.3. By fixing one dimension, one can regroup the N shares into a disjoint union of k subsets. By applying the chosen group law on the shares from the same subset, one gets k new hyper-shares such that the group law applied on these k hyper-shares equals the group law applied on the N original shares (thanks to commutativity). Hence, we can compute the MPC protocol Π_f realizing Equation (3.1) with these k hyper-shares as input. This reasoning can be extended to each other dimension (and it has to be extended to reach a soundness of $1/N$). It leads to d MPC protocols with k parties, thus to a reduction from N to kd parties to simulate. It can even be reduced to $k + (d-1)(\log_2 N - 1)$ parties (since each MPC protocol shall output the same result, once one protocol is computed, all but one parties is enough to determine the evaluation of the last party in the other protocols). By taking $(k = 2, d = \log_2 N)$, we get a reduction to $\log_2 N + 1$ parties.

Concretely, given $[\![x]\!] \in \mathbb{F}^N$, P derives the two hyper-shares for the ℓ-th MPC protocol (with $\ell \in [1, \log_2 N]$) as follows:

$$[\![x]\!]_{\ell,0} := \sum_{1 \le i \le N, (i_2)_\ell = 0} [\![x]\!]_i, \; [\![x]\!]_{\ell,1} := \sum_{1 \le i \le N, (i_2)_\ell = 1} [\![x]\!]_i;$$

where $(i_2)_\ell$ denotes the ℓ-th bit of the 2-basis representation of the element i.

Most of the time, this optimization makes the MPC protocols emulation less costly in computation and thus allows us to take a larger number of parties (and get smaller proof sizes). Although the computational gain is attenuated by the number of repetitions since the total number of parties to emulate is $\tau(1 + \log_2 N) \approx \lambda(1 + 1/\log_2 N)$, where τ is the number of repetitions of the protocol to get a negligible soundness, it is tempting to increase the number of parties

N to get an improvement in terms of proof size while keeping the computational cost of the proof reasonable. This strategy works, but has its limitations: the GGM tree to generate at the beginning of the protocol, as well as the shares preparation may become very costly (this cost is independent of the hypercubing optimization).

The MPCitH framework used in conjunction with a linear secret sharing scheme makes all linear operations free to prove in terms of elements to send to V. Multiplicative relationships, on the other hand, are the most cumbersome part. Hence we're focusing on it, detailing two techniques in the next two subsections.

3.3.2 Batch product verification

Let us consider the following context: given a triple of sharings $([\![x]\!], [\![y]\!], [\![z]\!])$ with $x, y, z \in \mathbb{F}$, we would like to check that $xy = z$. To this aim, we can use a solution suggested in [LN17, BN20a] by "sacrificing" another triple $([\![a]\!], [\![b]\!], [\![c]\!])$ that satisfies $ab = c$. The second triple can be used a single time (to preserve the zero-knowledge property), hence the "sacrifice". Recently Kales and Zaverucha [KZ22] have adapted and optimized this method to build an efficient MPC protocol which check simultaneously many products by sacrificing a dot-product. Specifically, given n triples $([\![x_j]\!], [\![y_j]\!], [\![z_j]\!])$ and a tuple $(([\![a_j]\!])_{j\in[1,n]}, [\![c]\!])$, their protocol verifies that $\langle a, y\rangle = c$ and $z_j = x_j y_j$ for all $j \in [1, n]$, without revealing any information on (x, y, z). The protocol runs as follows:

1. The parties get a random $\varepsilon \in \mathbb{F}^n$ from the verifier;
2. The parties locally sets $[\![\alpha_j]\!] = \varepsilon_j [\![x_j]\!] + [\![a_j]\!]$ for all $j \in [1, n]$;
3. The parties open α by broadcasting their shares;
4. The parties locally sets $[\![v]\!] = \langle \alpha, [\![y]\!]\rangle - [\![c]\!] - \langle \varepsilon, [\![z]\!]\rangle$;
5. The parties open v by broadcasting their shares;
6. The parties accept iff $v = 0$.

Protocol 1: Batch product verification

If $([\![x_j]\!], [\![y_j]\!], [\![z_j]\!])_{j\in[1,n]}$ contains an incorrect multiplication triple (*i.e.* there exists a j_0 such that $x_{j_0} y_{j_0} \neq z_{j_0}$) or if $((\![a_j]\!])_{j\in[1,n]}, [\![c]\!])$ does not satisfy the relation $\langle a, y\rangle = c$, then [KZ22] shows the parties accept with a probability at most $|\mathbb{F}|^{-1}$.

Complexity analysis. Concretely, the prover can randomly sample the shares $[\![a_j]\!] \xleftarrow{\$} \mathbb{F}^N$ for each $j \in [1, n]$, and then define $a_j = \sum_{i=1}^{N} [\![a_j]\!]_i$ for each $j \in [1, n]$. Since a can be chosen randomly, we do not need of an auxiliary value to fix the sharing. By generating a random sharing of c, the sole auxilliary value is Δc such that $\langle a, y\rangle - \sum_{i=1}^{N} [\![c]\!]_i = \Delta c$. During the Protocol 1, two values are brodcasted, $\alpha \in \mathbb{F}^n$ and $v \in \mathbb{F}$. We stress that the prover does not need to send $[\![v]\!]_{i^*}$ because the verifier knows that v must be zero and will deduce $[\![v]\!]_{i^*} = -\Delta v - \sum_{i\neq i^*} [\![v]\!]_i$. However, the prover has to send $[\![\alpha]\!]_{i^*}$. Therefore, this technique requires to send $n + 1$ field elements.

3.3.3 Cut-and-choose strategy

Beullens introduced the well-known *cut-and-choose* technique from zero-knowledge proofs into the MPCitH paradigm. This new proof-style was called MPCitH with *helper* [BdSG20]. This approach adds a trusted third party (called the helper) to the MPC protocol which runs a preprocessing phase. To then remove the helper, one uses a cut-and-choose strategy. Let us assume that the prover wishes to prove that a generated sharing indeed encodes a binary vector. At the beginning of the protocol, the prover generates M sharings $\{[\![r^{[\ell]}]\!]\}_{\ell\in[1,M]}$ for M binary vectors $\{r^{[\ell]}\}_{\ell\in[1,M]}$ (in practice these values are pseudo-randomly derived from some seeds). Then the prover commits to these values. The verifier asks to open all the sharings $\{[\![r^{[\ell]}]\!]\}_{\ell\in[1,M]}$ except one and checks that they correspond to binary vectors. Hence, the verifier will trust that the unopened sharing also encodes a binary vector with a soundness error of $1/M$. Combined with the MPCitH approach, the obtained zero-knowledge protocol when revealing all-but-one parties (assuming for the moment that the false positive probability of the MPC protocol is 0) has then a soundness error of

$$\max\left\{\frac{1}{M}, \frac{1}{N}\right\}.$$

Let ϵ be the soundness of one repetition of the protocol. To achieve a targeted soundness error of $2^{-\lambda}$, we can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. Like in [KKW18], instead of performing τ independent cut-and-choose phases each resulting in trusting one binary sharing $[\![r]\!]$ among M, we can perform a global cut-and-choose

phase resulting in τ trusted sharings among a larger M. The idea is that V asks to reveal $M - \tau$ out of M master seeds. The remaining τ executions of the pre-processing phase are used to emulate τ independent instances of the MPC protocol. When opening all but one seed, a wrong sharing (*i.e.* a sharing of a non-binary vector) will not be detected with probability

$$\frac{1}{N} + \left(1 - \frac{1}{N}\right) p, \tag{3.2}$$

where p is the false positive probability of the MPC protocol. If a cheating prover produces $M - k \leq \tau$ wrong sharings, they will not be detected during the first phase (when revealing $M - \tau$ master seeds) with probability

$$\binom{k}{M-\tau}\binom{M}{M-\tau}^{-1}.$$

This leads to the soundness error

$$\epsilon = \max_{M-\tau \leq k \leq M} \left\{ \frac{\binom{k}{M-\tau}\left(\frac{1}{N} + (1 - \frac{1}{N})p\right)^{k-M+\tau}}{\binom{M}{M-\tau}} \right\}.$$

This optimization reduces the communication cost of the cut-and-choose from $\tau\lambda \log_2 M$ bits to $\tau\lambda \log_2 \frac{M}{\tau}$ bits.

3.4 Examples and First Contributions

As an appetizer and to illustrate this chapter, we present two simple results with alternative approaches.

3.4.1 RSA-in-the-Head [MV23b]

The goal of this subsection is to build our first zero-knowledge argument, especially a ZKA of knowledge of an RSA plaintext for a small public exponent that significantly improves the state-of-the-art communication complexity. The scheme is very simple but seems to have been overlooked. Assume P wants to prove the knowledge of an RSA plaintext for a public exponent e, *i.e.* $x^e = y \bmod n$ where n is some RSA modulus. Then we could imagine that P shares x multiplicatively as $x = \prod_{j=1}^{N} \langle x \rangle_j \Delta x \bmod n$ and runs the following MPC protocol:

1. The parties locally compute $\langle x \rangle^e$;
2. The parties broadcast $\langle x \rangle^e$ and recomputes x^e;
3. The parties output 1 if $x^e = y$, and 0 otherwise.

Protocol 2: MPC protocol for RSA

Now, we can describe the zero-knowledge argument of knowledge protocol for the RSA plaintext where Protocol 2 is plugged into the red part of the following Protocol 3. A cryptographic hash function $\mathcal{H}_1$, a PRG and a TreePRG are used in this protocol.

Prover P (x, e, y, n) s.t. $x^e = y \bmod n$		Verifier V (e, y, n)
$\mathsf{mseed} \xleftarrow{\$} \{0,1\}^\lambda$ $(\mathsf{seed}_i, \rho_i)_{i\in[1,N]} \leftarrow \mathrm{TreePRG}(\mathsf{mseed})$ For each $i \in [1, N]$: $\quad \langle x\rangle_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$ $\quad \mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$ $\Delta x = x / \prod_i \langle x\rangle_i$ The parties compute $\Pi(\langle x\rangle, \Delta x)$ $h = \mathcal{H}_1(\mathsf{com}_1, \ldots, \mathsf{com}_N, \Delta x, \{\langle x\rangle_i^e\}_{i\in[1,N]})$	$\xrightarrow{h}$	
		$ch = i^* \xleftarrow{\$} [1, N]$
	$\xleftarrow{i^*}$	
	$\xrightarrow{\text{co-path to } \mathsf{seed}_{i^*},\ \Delta x,\ \mathsf{com}_{i^*}}$	
		For all $i \neq i^*$ $\quad \langle x\rangle_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$ $\quad \mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$ Compute $\langle x\rangle_{i^*}^e = y / \left(\prod_{i\neq i^*} \langle x\rangle_i^e (\Delta x)^e\right)$ Check h Return 1

Protocol 3: Identification scheme of an RSA plaintext

The soundness of Protocol 3 is $1/N$ since the MPC Protocol 2 is perfectly correct. When repeating the protocol τ times (in parallel) to reach a soundness error $\epsilon^\tau \leq 2^{-\lambda}$, we can use a second hash function $\mathcal{H}_2$ to merge the hash value h of each repetition, leading to one hash value for the τ repetitions. Then, the communication cost of the overall protocol is (in bits):

$$\underbrace{2\lambda}_{\mathcal{H}_2} + \tau\Big(\underbrace{2\log_2 n}_{\Delta x} + \underbrace{\lambda \log_2 N}_{\text{co-path}} + \underbrace{2\lambda}_{\mathsf{com}_{i^*}}\Big).$$

This simple construction improves the communication complexity of the seminal protocol from Guillou and Quisquater [GQ90] for the public exponent $e = 3$ from around 20.4KB to 6.6KB for a 2048-bit modulus n and has similar efficiency. The communication complexity could be made even smaller by increasing N (but at the cost of an increased computational complexity).

Remark 4. *Interestingly, the hypercube optimization [MGH+23] may be computationally more costly for some particular cases. For our ZKA for RSA with public exponent e, once $\langle x\rangle$ has been derived, they are around $N \log_2 e$ multiplications to perform to compute each share exponentiation (plus $N-1$ multiplications at the end for recomputing x^e). Whereas with the hypercubing technique with $(d, k) = (\log_2 N, 2)$, there are approximately $N \log_2 N$ multiplications for building 2 hypershares for each of the $\log_2 N$ MPC protocols, and approximately $2 \log_2 e \log_2 N$ multiplications for the overall number of exponentiation (plus $\log_2 N$ multiplications for recomputing x^e in each MPC protocols). Although the hypercubing technique is assymptotically better, for a typical case of $N = 2^8$ parties, as long as $e \leq 2^8$, the hypercubing approach is computationally more costly, and for bigger values of e, the Guillou-Quisquater protocol is then more efficient than our.*

3.4.2 DDLP-in-the-Head [MV23b]

We present a second ZKA, now for the Double Discrete Logarithm Problem (DDLP), a problem which has found numerous applications in cryptography [CS97, ASM10, CG07, CGM16, BTV20].

Double Discrete Logarithm Problem (DDLP).

Let $\mathbb{G}$ be a cyclic group of prime order q with some generator $g \in \mathbb{G}$, and let $h \in \mathbb{F}_q^*$ of prime order p with $p|(q-1)$. Given $(y, g, h) \in \mathbb{G} \setminus \{1_\mathbb{G}\} \times \mathbb{G} \times \mathbb{F}_q^*$, the *DDLP* asks to find some $x \in \mathbb{F}_p^\times$ such that $y = g^{h^x}$.

Given the public statement $\{y, g, h\}$, P wants to prove the knowledge of x such that $y = g^{h^x}$.

In Chapter 5, we will present another ZKA for this problem, but it turns out to be less efficient than this forward-backward construction (since a cut-and-choose has to be produced). Our ZKA is based on a recent idea of Joux [Jou23], a forward-backward technique achieving arguments about 6KB long. This improves the communication complexity of Stadler's protocol [Sta96] by about 75% (for the same security guarantees and overall efficiency). We start by sharing x additively. Then the prover P commits to the values

$$y_i := \left(\left(g^{h^{[\![x]\!]_1}}\right)^{h^{[\![x]\!]_2}}\right)^{\cdots}{}^{h^{[\![x]\!]_i}} \quad \text{for } i \in [1, N].$$

The correctness of this approach relies on the fact that $y_N = y$. The verifier V sends a challenge $i^* \xleftarrow{\$} [1, N]$. The prover P answers by sending the seeds $\{\mathsf{seed}_i\}_{i \neq i^*}$ (i.e. opens all the shares of x except the i^*th) to V. This last can recompute all the committed values by a forward-backward technique: they iteratively compute y_i as

$$y_i = y_{i-1}^{h^{[\![x]\!]_i}} \text{ if } 1 \leq i \leq i^* - 1 \text{ and } y_i = y_{i+1}^{-h^{[\![x]\!]_{i+1}}} \text{ if } i^* \leq i \leq N - 1,$$

with $y_0 = g$ and $y_N = y$.

In particular, V can recompute every y_i (to check the commitments) and they can check the consistency of the subsequences $\{y_1, \ldots, y_{i^*-1}\}$ and $\{y_{i^*}, \ldots, y_N = y\}$ thanks to the revealed seeds. But V can not check the whole sequence $\{y_1, \ldots, y_N\}$ since $[\![x]\!]_{i^*}$ is missing. Hence, a malicious prover has to cheat on $[\![x]\!]_{i^*}$ to manage convincing V. Therefore, the soundness error is again $1/N$. This yields the following MPC protocol 4, where a party P_i uses the output of P_{i-1} to compute their output (for $i \in [2, N]$). We notice that the following MPC model is not a classical broadcasting protocol because it has asynchronous broadcasts.

Input: $y \neq 1_{\mathbb{G}}$ in a cyclic group $\mathbb{G}$ of prime order q, $h \in \mathbb{F}_q^*$ of prime order p with $p|(q-1)$, and an additive sharing of $x \in \mathbb{F}_p^\times$.

1. Party P_1 computes $y_1 = g^{h^{[\![x]\!]_1}}$ and broadcasts it;
2. For each $i \in [2, N]$: Party P_i computes $y_i = y_{i-1}^{h^{[\![x]\!]_i}}$ and broadcasts it;
3. If $y_N = y$ parties output 1, otherwise 0.

Protocol 4: MPC protocol with the forward-backward technique for the DDLP

We could plug Protocol 4 into the red part of Protocol 3 (with some slight modifications, essentially substituting the multiplicative secret sharing by the additive one).

4. ZERO-KNOWLEDGE PROTOCOLS FOR THE SUBSET SUM PROBLEM FROM MPCITH WITH REJECTION

4.1 Introduction

The *(modular) subset sum* problem is to find, given integers $w_1, \ldots, w_n$, t and q, a subset of the w_i's that sum to t modulo q, *i.e.* to find bits $x_1, \ldots, x_n \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} x_i w_i = t \bmod q. \tag{4.1}$$

It was shown to be $\mathcal{NP}$-complete (in its natural decision variant) in 1972 by Karp [Kar72] and was considered in cryptography as an interesting alternative to hardness assumptions based on number theory. Due to its simplicity, it was notably used in the 1980s, following [MH78], for the construction of several public-key encryption schemes. Most of these proposals (if not all) were swiftly broken using lattice-based techniques (see [Odl90]), but the problem itself remains intractable for appropriate parameters and is even believed to be so for quantum computers. For instance, when the so-called density $d = n/\log_2(q)$ of the subset sum instance is close to 1 (*i.e.* $q \simeq 2^n$), the fastest known (classical and quantum) algorithms have complexity $2^{O(n)}$ (see [BBSS20] and references therein) and one can reach an alleged security level of λ bits with $n = \Theta(\lambda)$. A plethora of cryptographic constructions have been proposed whose security relies on the hardness of the subset sum problem: a celebrated pseudo-random generators [IN96], bit commitments [IN96], public-key encryption [AD97, LPS10], etc.

4.1.1 Prior works

Given integers $w_1, \ldots, w_n$, t and q, an elegant zero-knowledge proof system due to Shamir [Sha86] (see also [BGKW90, Sim91, Blo09][BGKW90]) allows a prover to convince a verifier that they knows $x_1, \ldots, x_n \in \{0, 1\}$ such that the relation (4.1) holds. The proof system is combinatorial in nature and it requires $\Theta(\lambda)$ rounds of communication to achieve soundness error $2^{-\lambda}$ where each round requires $\Theta(n^2)$ bits of communication. For an alleged security level of λ bits, the overall communication complexity of Shamir's proof system is thus of $\Theta(\lambda^3)$. In [LNSW13], Ling, Nguyen, Stehlé, and Wang proposed a proof of knowledge of a solution for the infinity norm *inhomogeneous small integer solution* (ISIS) problem which is a vectorial variant of the subset sum problem. It is based on Stern's zero-knowledge proof of knowledge for the *syndrome decoding* problem [Ste94b] and is also combinatorial. It thus requires a large number of rounds of communication and when specialized to the subset sum problem it also yields proofs with $\Theta(\lambda^3)$-bit communication complexity for an alleged security level of λ bits.

In [BD10], Bendlin and Damgård were the first to use the MPCitH paradigm in lattice-based cryptography. They proposed a zero-knowledge proof of knowledge of the plaintext contained in a given ciphertext from Regev's cryptosystem [Reg05] (and a variant they proposed). More recently, Baum and Nof [BN20a] proposed an efficient zero-knowledge argument of knowledge of the *short integer solution* (SIS) problem (incorporating the *sacrificing* principle in the MPCitH paradigm). Beullens also recently proposed such arguments obtained from sigma protocols *with helper* [Beu20]. When applied to the subset sum problem itself, all (variants of) these protocols yield proofs with $\Theta(\lambda^3)$-bit communication complexity for an alleged security level of λ bits.

There exist numerous other protocols for (vectorial variants of) the subset sum problem from lattice-based cryptography. Until recently, they all introduce some slack in the proof, *i.e.* there is a difference between the language used for completeness and the language that the soundness guarantees (see, *e.g.* [BDLN16] for a generic argument of knowledge of a pre-image for *homomorphic one-way functions over integer vectors*). In particular, the witness that can be extracted from a proof is larger than the one that an honest prover uses (and in the subset sum problem, the extractor will not output a binary vector). This slack forces to use larger parameters for the underlying cryptosystem and induces some loss in efficiency. Conversely, we shall only consider exact arguments for the subset-sum problem in the present paper. Finally, new exact arguments were proposed recently [BLS19, ENS20, LNS21] but they require to use a modulus q of a special form (namely a prime number as in [BN20a, Beu20] but with additional arithmetic constraints to make it "NTT-friendly").

4.1.2 Contributions

The MPCitH paradigm introduced in Chapter 3 involves a prover who wants to convince a verifier that they know a secret witness for some public statement. For the subset sum problem, the statement is $(w, q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and a witness is a binary string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ satisfying (4.1). Thus, it is thus natural to perform some secret-sharing of x in $\mathbb{Z}_q$ in such a way that the shares of any unauthorized set of parties should reveal no information about the secret. This approach has the major disadvantage that sharing a single bit requires several elements of $\mathbb{Z}_q$ each of size $\Theta(\lambda)$ bits.

We adapt this paradigm using a secret sharing scheme formed directly over the integers. This approach was already used in cryptography (*e.g.* for multi-party computation modulo a shared secret modulus [CGH00]). To additively share a secret t in a given interval $[-T, T]$ for $T \in \mathbb{N}$, among $n \geq 2$ parties, a dealer may pick uniformly at random $t_1, \ldots, t_n \in [-T2^\rho, T2^\rho]$ under the constraint that $t = t_1 + \cdots + t_n$ (over the integers), for some parameter ρ. However, given $(n-1)$ shares, $t_2, \ldots, t_n$ for instance, the value $t_1 = t - (t_2 + \cdots t_n)$ is not randomly distributed in $[-T2^\rho, T2^\rho]$ and this may reveal information on the secret t. It is thus necessary to sample the shares in an interval sufficiently large in such a way that their distributions for distinct secrets are statistically indistinguishable. For a security level λ, this requires $\rho = \Omega(\lambda)$ and thus the additive sharing of bits involves shares of size $\Omega(\lambda)$. To overcome this limitation and use additive secret sharing over *small* integers, we will rely on *rejection*. The computation being actually simulated by the prover, they can abort the protocol whenever the sharing leaks information on the secret vector $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. In some cases, the prover cannot respond to the challenge from the verifier and must abort the protocol. A similar idea was used for lattice-based signatures by Lyubashevsky [Lyu08, Lyu09] but using different methods.

Our technique also allows overcoming the second disadvantage of the previous tentatives to use the MPCitH paradigm for lattice-based problems. Indeed, using our additive secret sharing over the integers, we can prove the knowledge of some integer vector $x = (x_1, \ldots, x_n)$ satisfying relation (4.1) (for any q) and further prove that $x_i \in \{0,1\}$ for $i \in [1, n]$. This is achieved by simulating a (single) non-linear operation modulo some arbitrary prime number q' (independent from q and much smaller than q). We also introduce another technique to prove that the solution $x = (x_1, \ldots, x_n)$ indeed lies in $\{0,1\}^n$ using some masking and a *cut-and-choose* strategy. Both methods yield zero-knowledge proofs with $\Theta(\lambda^2)$-bit communication complexity for an alleged security level of λ bits. This improvement is not only of theoretical interest since for $q \simeq 2^{256}$, our protocol can produce proof of size 13KB where Shamir's protocol [Sha86] (updated with modern tips) produces proof of size 1186KB and [LNSW13] produces proofs of size 2350KB.

Our protocols are particularly efficient for the subset sum problem where the modulus q is large. However, we show that our method has applications in other contexts in cryptography. We show that it can be used for the (binary) ISIS problem in lattice-based cryptography and that the resulting protocols are competitive with state-of-the-art protocols for this problem. We also present applications of our techniques to the context of *fully-homomorphic encryption* (FHE). Specifically, adaptations of our protocols provide efficient zero-knowledge arguments of plaintext and/or key knowledge for the so-called *Torus Fully Homomorphic Encryption* (TFHE) scheme from [CGGI20]. Eventually, we use our technique to construct an efficient digital signature scheme based on a pseudo-random function due to Boneh, Halevi, and Howgrave-Graham [BHH01].

4.2 General Idea

We consider an instance $(w, t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ of the subset sum problem (SSP) and denote x one solution. We have $x \in \{0,1\}^n$ and $\sum_{j=1}^n x_j w_j = t \bmod q$.

We want to use the MPCitH paradigm to build a zero-knowledge protocol that proves the knowledge of a solution for the instance (w, t). To proceed, we need to build an MPC protocol with passive security whose parties take as inputs shares of the secret x, and possibly shares of other data, and which computation can only succeed if x is a valid solution of the SSP instance. As a first ingredient, we need a method to share the secret x between the different parties.

4.2.1 The naive approach

The SSP instance is defined on $\mathbb{Z}_q$, so a natural sharing of x would be defined as:

$$\begin{cases} [\![x]\!]_i \xleftarrow{\$} (\mathbb{Z}_q)^n \text{ for all } i \in [1, N], \\ \Delta x \leftarrow x - \sum_{i=1}^N [\![x]\!]_i \bmod q \end{cases}.$$

In the MPCitH paradigm, the communication cost of a sharing is the cost to send the auxiliary values, *i.e.* the vector Δx. Here, the natural sharing of x costs

$$n \log_2(q) \text{ bits.}$$

If we take $n = 256$ and $q = 2^{256}$, the cost is about 2^{16} bits $= 8$ KB. To achieve a soundness error of 2^{-128} with $N = 256$ parties, we need to repeat the protocol at least 16 times, so the communication cost of the protocol would be already more than 128 KB for the sole sharing of x. Asymptotically, the parameters for the subset sum problem are chosen such that $n = \Theta(\lambda)$ and $\log_2 q = \Theta(\lambda)$, the communication cost of this sharing is thus about $\Theta(\lambda^2)$ bytes per

protocol repetition. Since we need to repeat the protocol about $\Theta(\lambda)$ times to achieve a $2^{-\lambda}$ soundness error the global communication cost is then of at least $\Theta(\lambda^3)$ (for the sharing only).

We present hereafter an alternative strategy for the sharing of x, which achieves better practical and asymptotic communication costs.

4.2.2 *Sharing on the integers and opening with abort*

We propose another way to share the secret x to achieve lower communication. We know that x is a binary vector (*i.e.* $x \in \{0,1\}^n$), so instead of the natural sharing, we suggest to use a sharing defined on the integers, that is

$$\begin{cases} [\![x]\!]_i \xleftarrow{\$} [0, A-1]^n \text{ for all } i \in [1, N], \\ \Delta x \leftarrow x - \sum_{i=1}^{N} [\![x]\!]_i. \end{cases}$$

However, this sharing leaks information about the secret x. The distribution Δx_j is not the same depending on whether $x_j = 0$ or $x_j = 1$ as illustrated on Figure 4.1. To solve this issue, the prover must abort the protocol in some cases.

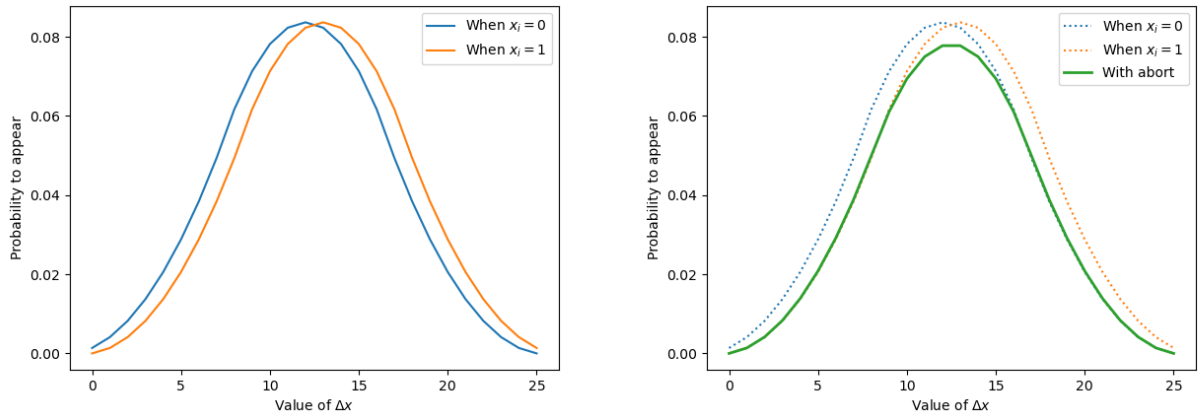


Fig. 4.1: Probability mass function of Δx_j when $x_j = 0$ and when $x_j = 1$ (on the left) and of Δx_j with abort (on the right), for $N = 3$ and $A = 9$.

To see how this leakage can be effectively exploited to (partly) recover x, let us recall that at the end of the protocol, the verifier shall ask the prover to open the views of all parties except one. Let us denote i^* the index of the unopened party. It means the verifier will have access to

$$\{[\![x]\!]_i\}_{i \neq i^*} \text{ and } \Delta x \,.$$

For the sake of simplicity, let us first consider the case $n = 1$, *i.e.* $x \in \{0,1\}$ and $[\![x]\!]$ is the sharing of a single integer. With the opened values, the verifier can compute

$$x - [\![x]\!]_{i^*} \text{ as } \Delta x + \sum_{i \neq i^*} [\![x]\!]_i \,.$$

Now let us denote $Y = x - [\![x]\!]_{i^*}$ the underlying random variable over the uniform random sampling of $[\![x]\!]_{i^*}$. We have

$$\Pr(Y = -A+1) = \begin{cases} \frac{1}{A} & \text{if } x = 0 \\ 0 & \text{if } x = 1 \end{cases} \quad \text{and} \quad \Pr(Y = 1) = \begin{cases} 0 & \text{if } x = 0 \\ \frac{1}{A} & \text{if } x = 1 \end{cases}$$

while

$$\Pr(Y = y) = \frac{1}{A} \quad \text{for every } y \in [-A+2, 0] \,.$$

So by observing $x - [\![x]\!]_{i^*} = -A+1$ one learns $(x, [\![x]\!]_{i^*}) = (0, -A+1)$. Similarly, by observing $x - [\![x]\!]_{i^*} = 1$ one learns $(x, [\![x]\!]_{i^*}) = (1, 0)$. To avoid this flaw, the prover must abort the protocol before revealing $\{[\![x]\!]_i\}_{i \neq i^*}$ and Δx whenever one of these two cases occurs. This notably implies that Δx must not be revealed before receiving the challenge i^*, but it should still be committed beforehand in order to ensure the soundness of the protocol. Doing so, we modify the distribution of the revealed auxiliary value which does not leak any information about x anymore as illustrated in Figure 4.1, and the probability to abort does not leak information about x since it is $1/A$ in the both cases ($x = 0$ and $x = 1$).

Let us now come back to the general case of $n \geq 1$. The prover applies the above abortion strategy for all the coordinates of x, namely

- if there exists $j \in [1, n]$ such that $x_j = 0$ and $[\![x_j]\!]_{i^*} = A - 1$, the prover aborts;
- if there exists $j \in [1, n]$ such that $x_j = 1$ and $[\![x_j]\!]_{i^*} = 0$, the prover aborts;
- otherwise the prover proceeds.

The probability to abort, which we call *rejection rate*, is

$$1 - \left(1 - \frac{1}{A}\right)^n \leq \frac{n}{A} .$$

We note that the rejection rate can be tightly approximated by the n/A upper bound when A is sufficiently large. In order to achieve a small (constant) rejection rate, we should hence choose A greater than n. Asymptotically, we then have $A = \Theta(n) = \Theta(\lambda)$, which represents an exponential improvement compared to $q = 2^{\Theta(\lambda)}$.

Let us now analyze the computation cost of our strategy for sharing x. In the absence of rejection, Δx_j belongs to $[-N(A-1)+1, 0]$, therefore sending the auxiliary value Δx would cost $n \log_2(N(A-1))$ bits. However, the prover can save communication by sending $x - [\![x]\!]_{i^*}$ instead, which is strictly equivalent in terms of revealed information by the relation $x - [\![x]\!]_{i^*} = \Delta x + \sum_{i \neq i^*} [\![x]\!]_i$. Since each coordinate of $x - [\![x]\!]_{i^*}$ is uniformly distributed over $[-A+2, 0]$, sending it only costs

$$n \log_2(A-1) \text{ bits.}$$

With $x - [\![x]\!]_{i^*}$, the verifier can recover Δx by computing $\Delta x = (x - [\![x]\!]_{i^*}) - \sum_{i \neq i^*} [\![x]\!]_i$. The cost of this sharing has the advantage of being independent of the modulus q on which the SSP instance is defined. The value of A will be chosen according to the desired trade-off between communication cost and rejection rate. If $n = 256$ and $A = 2^{16}$, we have a cost of 0.5 KB for a rejection rate of 0.0038, which is much better than the 8 KB of the naive approach.

Let us remark that adding an abort event does not impact the soundness of the protocol. A malicious prover can abort as many times she wants claiming that it would leak information, but an abortion does not help to convince the verifier. The soundness theorem will state that someone who does not know the secret can only answer with a probability smaller than the constant value called soundness error, and adding an abort event cannot increase this probability. The prover could sample a random party i' and give to i' a wrong share and she may indeed decide to abort if the verifier challenge is not i', but this does not change the fact that the probability for the prover to convince the verifier is the probability that the prover guesses the verifier challenge a priori.

Now that we have defined the sharing of x, we need to demonstrate two properties of the shared SSP instance through multi-party computation. The first one is the SSP relation which in the shared setting translates to

$$\sum_{j=1}^{n} [\![x_j]\!] w_j = [\![t]\!] \bmod q$$

for a sharing $[\![t]\!]$ of t. The linearity of this relation makes it easy to deal with: the share $[\![t]\!]_i$ can simply be computed as $[\![t]\!]_i := \sum_{j=1}^{n} [\![x_j]\!]_i w_j \bmod q$ and committed to the verifier by each party. The verifier can then check that the open parties have correctly computed their shares $[\![t]\!]_i$ and that the relation $\sum_{i=1}^{N} [\![t]\!]_i = t \bmod q$ well holds. The second property which must be demonstrated through multiparty computation is that the solution x corresponding to the sharing $[\![x]\!]$ is a binary vector. This is not a priori guaranteed to the verifier since the shares of the coordinate of x are defined over $[0, A-1]$ and the correctness of the linear relation does not imply that x is indeed binary. We present two different solutions to this issue in the following.

4.2.3 Binarity proof from batch product verification

Our first solution relies on standard MPCitH techniques to prove the relation

$$x \circ (x - \mathbf{1}) = \mathbf{0}$$

where $\circ$ denotes the coordinate-wise product, $\mathbf{0}$ and $\mathbf{1}$ are to be interpreted as the all-0 and all-1 vectors. To this aim, we can use the MPCitH batch product verification suggested in [LN17, BN20a] and recently improved in [KZ22] (see Subsection 3.3.2). However, we can do better than a straight application of those techniques.

The relation $x \circ (x - \mathbf{1}) = \mathbf{0}$ is defined in $\mathbb{Z}_q$ and the above techniques imply to send at least one field element per product, that is n elements from $\mathbb{Z}_q$. To save communication and since the sharing $[\![x]\!]$ is defined on the integers, we can work on a smaller field. We previously explained that the verifier receives $\{[\![x]\!]_i\}_{i \neq i^*}$ and Δx from the prover, so they can check that, for all $j \in [1, n]$,

$$-A + 2 \leq x_j - [\![x_j]\!]_{i^*} \leq 0 .$$

They further trusts $[\![x_j]\!]_{i^*} \in [0, A-1]$ (which is verified for the open parties). Thus the verifier can deduce that, for all $j \in [1, n]$,

$$-A + 2 \leq x_j \leq A - 1 . \tag{4.2}$$

Let q' be a prime such that $q' \geq A$. If the prover convinces the verifier that $x_j(x_j - 1) = 0 \bmod q'$, then the latter deduces that $x_j \in \{0, 1\}$ because

$$\begin{aligned} q'|x_j(x_j-1) &\Rightarrow (q'|x_j) \text{ or } (q'|x_j-1) \\ &\Rightarrow (x_j = 0) \text{ or } (x_j = 1) \quad \text{by (4.2)} \end{aligned}$$

The prover hence just needs to prove $x \circ (x - \mathbf{1}) = \mathbf{0} \bmod q'$ for some prime q' such that $q' \geq A$. To this purpose, we apply the batch product verification of [KZ22] as presented in Subsection 3.3.2. Given the protocol notations from Subsection 3.3.2, we identify $\mathbb{F} = \mathbb{Z}_{q'}$ (*i.e.* the computation is over $\mathbb{Z}_{q'}$), $y = \mathbf{1} - x$, and $z = \mathbf{0}$. The prover holds a sharing of x and of a (a uniformly random element in $(\mathbb{Z}_{q'})^n$), $c = \langle a, y \rangle$. They also give a random challenge $\varepsilon \in (\mathbb{Z}_{q'})^n$ from the verifier as inputs to the parties and runs the following MPC protocol:

1. the parties locally set $[\![\alpha]\!] = \varepsilon \circ [\![x]\!] + [\![a]\!]$;
2. the parties open $[\![\alpha]\!]$ to get α;
3. the parties locally set $[\![v]\!] = \langle \alpha, \mathbf{1} - [\![x]\!] \rangle - [\![c]\!]$;
4. the parties open $[\![v]\!]$ to get v and accept iff $v = 0$.

As described in Section 3.3.2, the batch product MPC verification produces false positives with probability $1/q'$. Thus the soundness error of the obtained zero-knowledge protocol is

$$\frac{1}{N} + \left(1 - \frac{1}{N}\right)\frac{1}{q'} .$$

Finally, the prover-to-verifier communication cost (in bits) for one repetition is

$$2(2\lambda) + \underbrace{n \log_2(A-1)}_{x - [\![x]\!]_{i^*}} + \underbrace{n \log_2(q')}_{\Delta\alpha} + \underbrace{\log_2(q')}_{\Delta c} + \lambda \log_2 N + 2\lambda ,$$

and the protocol has a rejection rate of $1 - (1 - \frac{1}{A})^n$.

4.2.4 Binarity proof from masking and cut-and-choose strategy

Our second solution to prove that $[\![x]\!]$ encodes a binary vector relies on a masking of x and a cut-and-choose strategy. The idea is to generate a random vector r from $\{0, 1\}^n$ and to apply the sharing described in Subsection 4.2.2 to r. In addition, the prover computes (and commits) $\tilde{x} := x \oplus r \in \{0, 1\}^n$ where $\oplus$ represents the XOR operation. Instead of giving the shares $[\![x]\!]$ of x as inputs of the MPC protocol, the idea is now to send the shares $[\![r]\!]$ of r. Then using $\tilde{x}$, the parties can locally deduce a sharing of x as

$$[\![x]\!] = (1 - \tilde{x}) \circ [\![r]\!] + \tilde{x} \circ (1 - [\![r]\!])$$

which is a linear relation in $[\![r]\!]$, and the verifier can further deduce the auxiliary value Δx from Δr as

$$\Delta x = (1 - \tilde{x}) \circ \Delta r + \tilde{x} \circ (1 - \Delta r) .$$

By replacing $[\![x]\!]$ with $[\![r]\!]$ the parties' input is made independent of the secret. The interest of doing so is to enable a cut-and-choose strategy to prove that $[\![r]\!]$ encodes a binary vector, which in turns implies that $x = \tilde{x} \oplus r$ is a binary vector. More precisely, at the beginning of the zero-knowledge protocol, the prover produces M binary vectors $r^{[\ell]}$ and their corresponding shares $[\![r^{[\ell]}]\!]$ (in practice these vectors and their sharings are pseudo-randomly derived from some seeds). Then the prover commits those sharings $[\![r^{[\ell]}]\!]$ as well as the corresponding masked vectors $\tilde{x}^{[\ell]} := x \oplus r^{[\ell]}$. Then the verifier asks to open all the sharings $r^{[\ell]}$ except one and checks that they correspond to binary vectors. The verifier will hence trust that the unopened sharing encodes also a binary vector with a soundness error of $1/M$. We stress that all the values $\tilde{x}^{[\ell]}$ for which $r^{[\ell]}$ is opened must remain hidden (otherwise x could be readily recovered). The obtained zero-knowledge protocol (for one repetition) has a soundness error of

$$\max\left\{\frac{1}{M}, \frac{1}{N}\right\} ,$$

a rejection rate of $1 - (1 - \frac{1}{A})^n$ and a prover-to-verifier communication cost (in bits) of

$$2(2\lambda) + \underbrace{\lambda \log_2 M}_{\text{Cost of C\&C}} + \underbrace{n \log_2(A-1)}_{r - [\![r]\!]_{i^*}} + \underbrace{n}_{\tilde{x}} + \lambda \log_2 N + 2\lambda .$$

4.2.5 Asymptotic Analysis

We analyze hereafter the asymptotic complexity of the two variants of our protocol. We show that for a security parameter λ both variants have an asymptotic communication cost of $\Theta(\lambda^2)$ and an asymptotic computation time of $\Theta(\lambda^4)$.

For the binarity proof based on masking and cut-and-choose, we assume $M = N$ (which is optimal for the communication cost given the soundness error). For the other parameters, let us recall that

- for a security parameter λ, one must take $n \approx \log_2 q = \Theta(\lambda)$,
- the prime q' can be chosen as the smallest prime greater than A, which implies $q' \approx A$.

For both variants, the asymptotic communication cost for one repetition of the protocol is then of

$$\Theta(\lambda \log_2 A + \lambda \log_2 N) .$$

Since each repetition has a soundness error of $\Theta(1/N)$, the protocol must be repeated $\tau = \Theta(\lambda/\log_2 N)$ times to reach a global soundness error of $2^{-\lambda}$. The probability that any of these τ repetitions aborts is given by

$$1 - \left(1 - \frac{1}{A}\right)^{n\tau} \approx \frac{n\tau}{A}$$

where the approximation is tight when A is sufficiently large. Thus for a small constant rejection probability, one must take $A = \Theta(n\tau) = \Theta(\lambda^2/\log_2 N)$. We have a communication cost for the τ iterations in

$$\Theta\left(\lambda^2 \frac{\log_2 A}{\log_2 N} + \lambda^2\right) = \Theta\left(\frac{\lambda^2}{\log_2 N} \log_2\left(\frac{\lambda^2}{\log_2 N}\right) + \lambda^2\right)$$

and we hence obtain a minimal asymptotic communication cost of $\Theta(\lambda^2)$ by taking $N = \Theta(\lambda)$.

The asymptotic computation time for one repetition of the protocol is of $\Theta(Nn \log_2 q \log_2 A)$, where the term $\log_2 q \log_2 A$ arises from the complexity of the multiplication between an element of $\mathbb{Z}_q$ and a value smaller than A. We hence get a computation time of $\Theta(\lambda^3 \log_2 \lambda)$ per repetition which makes $\Theta(\lambda^4)$ for τ repetitions.

4.3 Protocols and Security Proofs

In this section, we formally describe our two protocols and state their security. We further introduce a method to decrease the rejection rate.

4.3.1 Protocol with batch product verification

In Subsection 4.2.3, we proposed an MPC protocol proving that the sharing $[\![x]\!]$ encodes a binary vector. We then add the checking of the linear relation as described in Subsection 4.2.2 and we transform the MPC into a zero-knowledge protocol which proves the knowledge of a solution of an SSP instance. We give the formal description of our protocol in Protocol 5. The protocol makes use of a pseudo-random generator PRG, a tree-based pseudo-random generator $\mathrm{TreePRG}$, two collision-resistant hash functions Hash_i for $i \in \{1, 2\}$ and a commitment scheme $(\mathrm{Com}, \mathrm{Verif})$. In this description, the procedure Check returns 0 if the evaluated condition is false (*i.e.* the equality does not hold) and the execution continues otherwise.

To achieve a targeted soundness error $2^{-\lambda}$, we can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. As mentioned in Chapter 2, such parallel repetition does not preserve (general) zero-knowledge and the resulting scheme achieves *honest-verifier* zero-knowledge. Following the optimizations developed Chapter 3, instead of sending τ values for h and h', the prover can merge them together to send a single h and a single h'. Moreover, instead to sending the $N - 1$ seeds and commitment randomness of $(\mathsf{seed}_i, \rho_i)_{i \neq i^*}$ for each execution, we can instead send the co-path from $(\mathsf{seed}_{i^*}, \rho_{i^*})$ to the tree root, it costs at most $\lambda \log_2(N)$ bits (we need to reveal $\log_2(N)$ nodes of the tree) by execution (this could be optimized with [BBM+24]). The communication cost (in bits) of the protocol with τ repetitions is then

$$\textsc{Size} = 4\lambda + \tau\left[n(\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda\right]$$

while the soundness error and rejection rate scale as

$$\left(\frac{1}{N} + \left(1 - \frac{1}{N}\right)\frac{1}{q'}\right)^\tau \quad \text{and} \quad 1 - \left(1 - \frac{1}{A}\right)^{\tau n}$$

respectively. Let us stress that the obtained size is independent of the modulus q (and of the size of the integers $\{w_j\}, t$).

Prover P		Verifier V
$x \in \{0,1\}^n$		
$w \in \mathbb{Z}_q^n, t = \langle w, x \rangle$		w, t
$\mathsf{mseed} \xleftarrow{\$} \{0,1\}^\lambda$		
Compute parties' seeds		
$(\mathsf{seed}_1, \rho_1), \ldots, (\mathsf{seed}_N, \rho_N)$		
with $\mathrm{TreePRG}(\mathsf{mseed})$		
For each party $i \in [1, N]$:		
$[\![a]\!]_i, [\![x]\!]_i, [\![c]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$		$\triangleright\ a \in \mathbb{Z}_{q'}^n,\ c \in \mathbb{Z}_{q'},\ [\![x]\!]_i \in [0, A-1]^n$
$\mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$		
$\Delta x = x - \sum_i [\![x]\!]_i$		
$\Delta c = \langle a, x \rangle - \sum_i [\![c]\!]_i$		
$h = \mathcal{H}_1(\Delta x, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$		
	$\xrightarrow{h}$	
		$\varepsilon \xleftarrow{\$} \mathbb{Z}_{q'}^n$
	$\xleftarrow{\varepsilon}$	
The parties locally set		
- $[\![t]\!] = \langle w, [\![x]\!] \rangle$		$\triangleright\ t \in \mathbb{Z}_q$
- $[\![\alpha]\!] = \varepsilon \circ (1 - [\![x]\!]) + [\![a]\!]$		$\triangleright\ \alpha \in \mathbb{Z}_{q'}^n$ (computation in $\mathbb{Z}_{q'}$)
The parties open $[\![\alpha]\!]$ to get α.		
The parties locally set		
$[\![v]\!] = \langle \alpha, [\![x]\!] \rangle - [\![c]\!]$		$\triangleright\ v \in \mathbb{Z}_{q'}$ (computation in $\mathbb{Z}_{q'}$)
$h' = \mathcal{H}_2([\![t]\!], [\![\alpha]\!], [\![v]\!])$		
	$\xrightarrow{h'}$	
		$i^* \xleftarrow{\$} [1, N]$
	$\xleftarrow{i^*}$	
If there exists $j \in [1, n]$ such that:		
- either $[\![x_j]\!]_{i^*} = 0$ with $x_j = 1$		
- or $[\![x_j]\!]_{i^*} = A - 1$ with $x_j = 0$,		
then abort.		
$y = x - [\![x]\!]_{i^*}$		
	$(\mathsf{seed}_i, \rho_i)_{i \neq i^*},\ \mathsf{com}_{i^*},$ $y,\ \Delta c,\ [\![\alpha]\!]_{i^*}$ $\xrightarrow{\quad}$	
		For all $i \neq i^*$,
		$[\![a]\!]_i, [\![x]\!]_i, [\![c]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		$\Delta x = y - \sum_{i \neq i^*} [\![x]\!]_i$
		$\Delta \alpha = \varepsilon \circ (1 - \Delta x)$
		For all $i \neq i^*$,
		Rerun the party i as the prover
		and compute the commitment com_i.
		$\Delta t = \langle w, \Delta x \rangle$
		$\Delta v = \langle \alpha, \Delta x \rangle - \Delta c$
		$[\![t]\!]_{i^*} = t - \Delta t - \sum_{i \neq i^*} [\![t]\!]_i$
		$[\![v]\!]_{i^*} = -\Delta v - \sum_{i \neq i^*} [\![v]\!]_i$
		Check $h = \mathcal{H}_1(\Delta x, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$
		Check $h' = \mathcal{H}_2([\![t]\!], [\![\alpha]\!], [\![v]\!])$
		Return 1

Protocol 5: Zero-knowledge argument for Subset Sum Problem via MPC-in-the-Head with rejection, using batch product verification to prove binarity.

4.3.2 Security proofs for Protocol 5

The following theorems state the completeness, zero-knowledge and soundness of Protocol 5.

Theorem 4 (Completeness). *A prover P who knows a solution x to the subset sum instance $(w,t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and who follows the steps of Protocol 5 convinces the verifier V with probability*

$$\left(1 - \frac{1}{A}\right)^n .$$

Theorem 5 (Zero-Knowledge). *Let the* PRG *used in Protocol 5 be (t, ε_{PRG})-secure and the commitment scheme* Com *be (t, ε_{Com})-hiding. Then there exists an efficient simulator Sim that outputs a transcript which is $(t, \varepsilon_{PRG}+\varepsilon_{Com})$-indistinguishable from a real transcript of Protocol 5.*

Theorem 6 (Soundness). *Suppose that there is an efficient prover $\tilde{P}$ that, on input (w,t), convinces the honest verifier V on input (w,t) to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{P}(w,t), V(w,t)\rangle = 1] > \epsilon$$

for a soundness error ϵ equal to

$$\frac{1}{N} + \left(1 - \frac{1}{N}\right)\frac{1}{q'} .$$

Then, there exists an efficient probabilistic extraction algorithm E that, given rewindable black-box access to $\tilde{P}$, produces either a witness x such that $t = \langle w, x\rangle$ and $x \in \{0,1\}^n$, or a commitment collision, by making an average number of calls to $\tilde{P}$ which is upper bounded by

$$\frac{4}{\tilde{\epsilon} - \epsilon}\left(1 + \tilde{\epsilon}\frac{2\ln(2)}{\tilde{\epsilon} - \epsilon}\right) .$$

As a preliminary note before entering into these proofs, we make an aside on abort events. In what follows, the complementary of an event $\mathcal{E}$ is denoted $\neg\mathcal{E}$. For all the proofs in this section, we introduce the following events: for all $j \in [1,n]$,

- $\mathsf{A}_j^0 := \{x_j = 0, [\![x_j]\!]_{i^*} = A - 1\}$ which is the first case of abortion,
- $\mathsf{A}_j^1 := \{x_j = 1, [\![x_j]\!]_{i^*} = 0\}$ which is the second case of abortion,
- $\mathsf{A}_j := \mathsf{A}_j^0 \cup \mathsf{A}_j^1$.

Now let us denote abort the event when Protocol 5 aborts. By construction of the protocol, we have

$$\Pr[\mathsf{abort}] := \Pr[\bigcup_{j=1}^{n} \mathsf{A}_j].$$

Let X be a random variable modeling the secret vector x. For any $x \in \{0,1\}^n$, we have

$$\begin{aligned}
\Pr[\mathsf{abort} \mid X = x] &= \Pr[\bigcup_{j=1}^{n} \mathsf{A}_j \mid X = x] \\
&= 1 - \Pr[\bigcap_{j=1}^{n} (\neg\mathsf{A}_j^0 \cap \neg\mathsf{A}_j^1) \mid X = x] \\
&= 1 - \Pr[\bigcap_{j=1}^{n} \neg A_j^{x_j} \mid X = x] \qquad (4.3) \\
&= 1 - \Pr[\bigcap_{j=1}^{n} [\![x_j]\!]_{i^*} \neq (1 - x_j)\cdot(A-1)] \\
&= 1 - \prod_{j=1}^{n} \Pr[[\![x_j]\!]_{i^*} \neq (1 - x_j)\cdot(A-1)] \qquad (4.4) \\
&= 1 - \left(1 - \frac{1}{A}\right)^n .
\end{aligned}$$

The equality (4.3) comes from the fact that $\neg\mathsf{A}_j^{1-x_j}$ is true when $X_j = x_j$, and the equality (4.4) comes from the independency between the coordinates of the share $[\![x]\!]_{i^*}$. We get that the probability of the event abort is independent of X and satisfies:

$$\Pr[\mathsf{abort}] = 1 - \left(1 - \frac{1}{A}\right)^n . \qquad (4.5)$$

Proof. (Theorem 4) For any sampling of the random coins of P and V, if the computation described in the protocol is honestly performed and if there is *no abort*, all the checks of V pass. The completeness probability is hence of $1 - \Pr[\text{abort}]$, which from (4.5) implies the theorem statement. □

Proof. (Theorem 5) Before building the desired simulator (*i.e.* an algorithm that outputs transcripts that are indistinguishable from real transcripts without knowing the secret), let us first show the independence between the secret x and some events and values that can be observed from the transcript.

- The abortion event abort must be independent of the secret x, *i.e.*

$$\Pr[\text{abort}|x] = \Pr[\text{abort}],$$

  it ensures that the fact to abort does not leak any information. This independence has been demonstrated just below.

- When there is no abort, the transcript includes some values computed from the secret. While one can directly remark that the values of some elements are independent of the secret since they are masked by the uniform values of $[\![c]\!]_{i^*}$ and $[\![a]\!]_{i^*}$, it is less clear for $y := x - [\![x]\!]_{i^*}$. So let us explicit the probability distribution of y given that the protocol did not abort and given the shares $\{[\![x]\!]_i\}_{i\neq i^*}$. Let X and Y be random variables respectively modeling x and $y = x - [\![x]\!]_{i^*}$. For any $y \in [-A+2, 0]^n$ and $x \in \{0,1\}^n$, we have

$$\begin{aligned}
&\Pr[Y = y \mid X = x, \{[\![x]\!]_i\}_{i\neq i^*}, \neg\text{abort}] \\
&\qquad = \Pr[[\![x]\!]_{i^*} = y + x \mid \bigcap_{j=1}^{n} (\neg \mathsf{A}_j^{x_j})] \\
&\qquad = \Pr\left[[\![x]\!]_{i^*} = y + x \;\middle|\; \bigcap_{j=1}^{n} ([\![x_j]\!]_{i^*} \neq (1-x_j)(A-1))\right] \\
&\qquad = \prod_{j=1}^{n} \Pr\left[[\![x_j]\!]_{i^*} = y_j + x_j \;\middle|\; [\![x_j]\!]_{i^*} \neq (1-x_j)(A-1)\right] \\
&\qquad = \left(\frac{1}{A-1}\right)^n
\end{aligned}$$

  We deduce that the coordinates of $x - [\![x]\!]_{i^*}$ follow the uniform distribution in $[-A+2, 0]$ and that $y = x - [\![x]\!]_{i^*}$ (together with the occurrence of $\neg$abort and the shares $\{[\![x]\!]_i\}_{i\neq i^*}$) does not leak any information about the secret x.

Let us now describe the simulator Sim who has oracle access to some probabilistic-polynomial time $\tilde{\mathsf{V}}$, and works as follows (we keep the notation from Protocol 5):

Simulator Sim:

1. Sample a challenge $i^* \xleftarrow{\$} [1, N]$.
2. Sample mseed $\xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute parties' seeds $(\text{seed}_1, \rho_1), \ldots, (\text{seed}_N, \rho_N)$ with $\text{TreePRG}(\text{mseed})$.
4. For each party $i \in [1, N]\backslash\{i^*\}$: $[\![a]\!]_i, [\![x]\!]_i, [\![c]\!]_i \leftarrow \text{PRG}(\text{seed}_i)$ and $\text{com}_i = \text{Com}(\text{seed}_i; \rho_i)$
5. Sample $y \xleftarrow{\$} [-A+2, 0]^n$, $\Delta x = y - \sum_{i\neq i^*} [\![x]\!]_i$, and $\Delta c \xleftarrow{\$} \mathbb{Z}_{q'}$.
6. Sample a random commitment com_{i^*}.
7. Call $\tilde{\mathsf{V}}$ with the hash digest h of $\Delta x, \Delta c$ (and of the commitments of the seed and associated randomness of each party) and gets a challenge ε.
8. Sample $\alpha \xleftarrow{\$} \mathbb{Z}_{q'}^n$.
9. Simulate the computation of all the parties $i \neq i^*$ to get $\{[\![t]\!]_i, [\![\alpha]\!]_i, [\![v]\!]_i\}_{i\neq i^*}$ and $(\Delta t, \Delta\alpha, \Delta v)$.

10. Adapt the messages from and the outputs of the party i^*: $[\![\alpha]\!]_{i^*} = \alpha - \Delta\alpha - \sum_{i\neq i^*}[\![\alpha]\!]_i \bmod q'$, $[\![t]\!]_{i^*} = t - \Delta t - \sum_{i\neq i^*}[\![t]\!]_i \bmod q$, and $[\![v]\!]_{i^*} = 0 - \Delta v - \sum_{i\neq i^*}[\![v]\!]_i \bmod q'$
11. Call $\tilde{\mathsf{V}}$ with the hash digest h' of $[\![t]\!], [\![\alpha]\!], [\![v]\!]$ and gets a challenge $\tilde{i}^*$. If $\tilde{i}^* \neq i^*$, then Sim restarts the simulation from scratch.
12. Abort with probability
$$1 - \left(1 - \frac{1}{A}\right)^n .$$
13. Outputs the transcript
$$\left(h, h', (\mathsf{seed}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, y, \Delta c, [\![\alpha]\!]_{i^*}\right) .$$

When no abortion occurs, the output transcript is identically distributed to the genuine transcript except for the commitment of the party i^*. Distinguishing them means breaking the commitment hiding property or the PRG security. The above simulator Sim is a probabilistic polynomial-time algorithm since the challenge set $[1, N]$ (from which i^* is sampled) has a size that is polynomial in the security level.

□

Proof. (Theorem 6) For the sake of simplicity, we assume that the commitment scheme is perfectly binding. (If the commitment scheme was computationally binding we would have to deal with additional cases where the extractor would produce a commitment collision.)

For any set of successful transcripts corresponding to the same commitment, with at least two challenges for unopened party (i^*),

- either the revealed shares of $[\![x]\!]$ are not consistent, and then we find a hash collision (if the committed values are not the same, then the commitments cannot be the same since the commitment scheme is perfectly binding),
- or the openings are unique and hence the underlying witness $[\![x]\!]$ is uniquely defined.

This witness can be recovered from any two successful transcripts T_1 and T_2 corresponding to the same commitment and for which $i^*_{T_1} \neq i^*_{T_2}$. Let us call a witness $[\![x]\!]$ a *good witness* whenever

$$\langle w, x\rangle = t \quad \text{and} \quad x \circ (x-1) = 0$$

where $x := \sum_i [\![x]\!]_i$. Such a witness enables us to build a solution for the subset sum instance.

In what follows, we consider that the extractor only gets transcripts with consistent shares since otherwise, the extractor would find a hash collision.

We shall further denote by R_h the randomness of $\tilde{\mathsf{P}}$ which is used to generate the initial commitment $\textsc{Com} = h$, and we denote r_h a possible realization of R_h. Let us now describe the extractor procedure:

Extractor E:

1. Repeat $+\infty$ times:
2. Run $\tilde{\mathsf{P}}$ with honest Vto get transcript T_1
3. If T_1 is not a successful transcript, go to the next iteration
4. Do N_1 times:
5. Run $\tilde{\mathsf{P}}$ with honest Vand same r_h as T_1 to get transcript T_2
6. If T_2 is a successful transcript, $i^*_{T_1} \neq i^*_{T_2}$ and (T_1, T_2) reveals a good witness,
7. Return (T_1, T_2)

In what follows, we estimate the extraction complexity, *i.e.* how many time in average the extractor calls $\tilde{\mathsf{P}}$. Throughout the proof, we denote $\mathsf{succ}_{\tilde{\mathsf{P}}}$ the event that $\tilde{\mathsf{P}}$ succeeds in convincing V. By hypothesis, we have $\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}] = \tilde{\epsilon}$.

Let us fix an arbitrary value $\alpha \in (0, 1)$ such that $(1-\alpha)\tilde{\epsilon} > \epsilon$, it exists since $\tilde{\epsilon} > \epsilon$. Let r_h be a possible realization of R_h. We will say that r_h is *good* if it is such that

$$\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h] \geq (1-\alpha)\tilde{\epsilon} . \tag{4.6}$$

By the Splitting Lemma 1 (see Appendix 2.3.1) we have

$$\Pr[R_h \text{ good} \mid \mathsf{succ}_{\tilde{\mathsf{P}}}] \geq \alpha . \tag{4.7}$$

Let assume we sample a successful transcript T_1 as in the step 2 of the extractor E and let r_h be the underlying realization of R_h. Assume r_h is good. By definition, we have

$$\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h] \geq (1-\alpha)\tilde{\epsilon} > \epsilon > \frac{1}{N}$$

implying that there must exist a successful transcript T_2 with $i^*_{T_2} \neq i^*_{T_1}$. As explained above, this implies that there exists a unique and well-defined witness $[\![x]\!]$ corresponding to these transcripts (and to all the transcripts with same r_h).

We will show that if this witness is a *bad* witness (*i.e.* is not a good witness) then we have $\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h] \leq \epsilon$ meaning that r_h is *not* good. By contraposition, we get that if r_h is good, then the witness $[\![x]\!]$ is a good witness. So let us assume that the witness $[\![x]\!]$ in T_1 is a bad witness. This means that

$$\langle w, x\rangle \neq t \quad \text{or} \quad x \circ (x-1) \neq 0$$

where $x = \sum_i [\![x]\!]_i$. Let us denote FP the event that a geniune execution of the batch product checking outputs a false positive, *i.e.* outputs a zero vector v. We have

$$\Pr[\mathsf{FP}] \leq \frac{1}{q'}$$

according to Section 3.3.2.

Let us upper bound the probability that the inner loop finds a successful transcript:

$$\begin{aligned}\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h] &= \Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}, \mathsf{FP} \mid R_h = r_h] + \Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}, \neg\mathsf{FP} \mid R_h = r_h] \\ &\leq \frac{1}{q'} + (1 - \frac{1}{q'}) \Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h, \neg\mathsf{FP}]\end{aligned}$$

Having a successful transcript means that the sharings $[\![v]\!]$ and $[\![t]\!]$ in the first response of the prover must encode respectively a zero vector and t. But the event $\neg\mathsf{FP}$ when we have $x \cdot (x-1) \neq 0$ implies that a geniune execution outputs a *non-zero* vector v, and if $x \circ (x-1) = 0$, it implies that $[\![t]\!]$ does not correspond to the vector t (since the witness is bad). So to have a successful transcript, the prover must cheat for the simulation of at least one party. If the prover cheats for several parties, there is no way it can produce a successful transcript, while if the prover cheats for exactly one party (among the N parties), the probability to be successful is at most $1/N$. Thus, $\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h, \neg\mathsf{FP}] \leq 1/N$ and we have

$$\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}} \mid R_h = r_h] \leq p + (1-p)\frac{1}{N} = \epsilon,$$

meaning that r_h is *not* good. By contraposition, we get that if r_h is good, then $[\![x]\!]$ is a good witness.

Now, let us lower bound the probability that the ith iteration of the inner loop finds a successful transcript T_2 such that $i^*_{T_1} \neq i^*_{T_2}$ in the presence of a good R_h. We have

$$\begin{aligned}&\Pr[\mathsf{succ}^{T_2}_{\tilde{\mathsf{P}}} \cap (i^*_{T_1} \neq i^*_{T_2}) \mid R_h \text{ good}] \\ &\quad= \Pr[\mathsf{succ}^{T_2}_{\tilde{\mathsf{P}}} \mid R_h \text{ good}] - \Pr[\mathsf{succ}^{T_2}_{\tilde{\mathsf{P}}} \cap (i^*_{T_1} = i^*_{T_2}) \mid R_h \text{ good}] \\ &\quad\geq (1-\alpha)\tilde{\epsilon} - \Pr[i^*_{T_1} = i^*_{T_2} \mid R_h \text{ good}] \\ &\quad= (1-\alpha)\tilde{\epsilon} - \Pr[i^*_{T_1} = i^*_{T_2}] \\ &\quad= (1-\alpha)\tilde{\epsilon} - 1/N \\ &\quad\geq (1-\alpha)\tilde{\epsilon} - \epsilon\end{aligned}$$

Let define $p_0 := (1-\alpha)\tilde{\epsilon} - \epsilon$. By running $\tilde{\mathsf{P}}$ with the same r_h as for the good transcript N_1 times, we hence obtain a second non-colliding transcript T_2 with probability at least $1/2$ when

$$N_1 \approx \frac{\ln(2)}{\ln\left(\frac{1}{1-p_0}\right)} \leq \frac{\ln(2)}{p_0}\,. \tag{4.8}$$

Let C denotes the number of calls to $\tilde{\mathsf{P}}$ made by the extractor before finishing. While entering a new iteration:

- the extractor makes one call to $\tilde{\mathsf{P}}$ to obtain T_1,
- if T_1 is not successful, which occurs with probability $(1 - \Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}])$,
  - the extractor continues to the next iteration and makes an average of $\mathbb{E}[C]$ calls to $\tilde{\mathsf{P}}$,
- if T_1 is successful, which occurs with probability $\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}]$,

- if r_h is good which occurs with probability α, the extractor makes at most N_1 calls to $\tilde{\mathsf{P}}$ in the inner loop of E and output a pair (T_1, T_2) with probability $1/2$,
- otherwise the extractor makes N_1 calls to $\tilde{\mathsf{P}}$ in the inner loop of E without stopping, with probability at most $(1 - \frac{\alpha}{2})$.

The mean number of calls to $\tilde{\mathsf{P}}$ hence satisfies the following inequality:

$$\mathbb{E}[C] \leq 1 + \underbrace{(1 - \Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}])\mathbb{E}[C]}_{T_1 \text{ unsuccessful}} + \underbrace{\Pr[\mathsf{succ}_{\tilde{\mathsf{P}}}]\Big(N_1 + \big(1 - \frac{\alpha}{2}\big)\mathbb{E}[C]\Big)}_{T_1 \text{ successful}}$$

which gives

$$\begin{aligned}
\mathbb{E}[C] &\leq 1 + (1 - \tilde{\epsilon})\mathbb{E}[C] + \tilde{\epsilon}\Big(N_1 + \big(1 - \frac{\alpha}{2}\big)\mathbb{E}[C]\Big) \\
&\leq 1 + \tilde{\epsilon}N_1 + \mathbb{E}[C]\Big(1 - \frac{\tilde{\epsilon}\alpha}{2}\Big) \\
&\leq \frac{2}{\alpha\tilde{\epsilon}}(1 + \tilde{\epsilon}N_1) \\
&\leq \frac{2}{\alpha\tilde{\epsilon}}\left(1 + \tilde{\epsilon}\frac{\ln(2)}{(1-\alpha)\tilde{\epsilon} - \epsilon}\right).
\end{aligned}$$

To obtain an α-free formula, let us take α such that $(1 - \alpha)\tilde{\epsilon} = \frac{1}{2}(\tilde{\epsilon} + \epsilon)$. We have $\alpha = \frac{1}{2}\left(1 - \frac{\epsilon}{\tilde{\epsilon}}\right)$ and the average number of calls to $\tilde{\mathsf{P}}$ is upper bounded by

$$\frac{4}{\tilde{\epsilon} - \epsilon}\left(1 + \tilde{\epsilon}\frac{2\ln(2)}{\tilde{\epsilon} - \epsilon}\right)$$

which concludes the proof. □

4.3.3 Protocol with cut-and-choose strategy

As described in Subsection 4.2.4, we can also use a cut-and-choose strategy to prove that the vector $[\![x]\!]$ is binary. It is possible since we can remplace the input $[\![x]\!]$ of the MPC by a sharing $[\![r]\!]$ *independent* of the secret, where r is a mask uniformly sampled in $\{0,1\}^n$. To achieve a targeted soundness error $2^{-\lambda}$, we can perform τ parallel executions of the protocol such that $\epsilon^\tau \leq 2^{-\lambda}$. Like in [KKW18] and as developed in Chapter 3, instead of performing τ independent cut-and-choose phases each resulting in trusting one sharing $[\![r]\!]$ among M, we can perform a global cut-and-choose phase resulting in τ trusted sharings $[\![r]\!]$ among a larger M. We give the formal description of this zero-knowledge protocol in Protocol 6. The protocol makes use of a pseudo-random generator PRG, a tree-based pseudo-random generator $\mathrm{TreePRG}$, four collision-resistant hash functions Hash_i for $i \in \{1,2,3,4\}$ and a commitment scheme $(\mathrm{Com}, \mathrm{Verif})$. In this description, the procedure Check returns 0 if the evaluated condition is false (*i.e.* the equality does not hold) and the execution continues otherwise.

Prover P		Verifier V
$x \in \{0,1\}^n$		
$w \in \mathbb{Z}_q^n, t = \langle w, x \rangle$		w, t
$\mathsf{mseed}^{[0]} \xleftarrow{\$} \{0,1\}^\lambda$		
$(\mathsf{mseed}^{[e]})_{e\in[1,M]} \leftarrow \mathrm{TreePRG}(\mathsf{mseed}^{[0]})$		
For each $e \in [1, M]$:		
$r^{[e]} \leftarrow \mathrm{PRG}(\mathsf{mseed}^{[e]})$		$\triangleright\ r^{[e]} \in \{0,1\}^n$
$(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\in[1,N]} \leftarrow \mathrm{TreePRG}(\mathsf{mseed}^{[e]})$		
For each $i \in [1, N]$:		
$[\![r^{[e]}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i^{[e]})$		$\triangleright\ [\![r^{[e]}]\!]_i \in [0, A-1]^n$
$\mathsf{com}_i^{[e]} = \mathrm{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$		
$\Delta r^{[e]} = r^{[e]} - \sum_i [\![r^{[e]}]\!]_i$		
$h_e = \mathcal{H}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$		
$h = \mathcal{H}_2(h_1, \ldots, h_M)$		
	$\xrightarrow{\quad h \quad}$	
		$J \xleftarrow{\$} \{J \subset [1, M]\ ;\ \lvert J\rvert = \tau\}$
	$\xleftarrow{\quad J \quad}$	
For each $e \in J$:		
$\tilde{x}^{[e]} = x \oplus r^{[e]}$		$\triangleright\ \oplus$ is the XOR operation ($\tilde{x} \in \{0,1\}^n$)
The parties locally set		
$[\![x]\!]^{[e]} = (1 - \tilde{x}^{[e]}) \circ [\![r^{[e]}]\!]$		
$+\tilde{x}^{[e]} \circ (1 - [\![r^{[e]}]\!])$		
and they set $[\![t]\!]^{[e]} = \langle w, [\![x]\!]^{[e]} \rangle$.		
$h'_e = \mathcal{H}_3(\tilde{x}^{[e]}, [\![t]\!]^{[e]})$		
$h' = \mathcal{H}_4((h'_e)_{e\in J})$		
	$\xrightarrow{\quad h',\ (\mathsf{mseed}^{[e]})_{e\in[1,M]\setminus J} \quad}$	
		$L = \{\ell_e\}_{e\in J} \xleftarrow{\$} [1, N]^\tau$
	$\xleftarrow{\quad L \quad}$	
If there exists $(e, j) \in J \times [1, n]$ such that:		
- either $[\![r_j^{[e]}]\!]_{\ell_e} = 0$ with $r_j^{[e]} = 1$		
- or $[\![r_j^{[e]}]\!]_{\ell_e} = A - 1$ with $r_j^{[e]} = 0$,		
then abort.		
$y = r^{[e]} - [\![r^{[e]}]\!]_{\ell_e}$		
	$\left(\begin{array}{c}(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\neq\ell_e} \\ y, \tilde{x}^{[e]},\ \mathsf{com}_{\ell_e}^{[e]}\end{array}\right)_{e\in J}$	
		For each $e \notin J$:
		Compute h_e using $\mathsf{mseed}^{[e]}$
		For each $e \in J$:
		For all $i \neq \ell_e$
		$\mathsf{com}_i^{[e]} = \mathrm{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$
		Rerun the party i
		as the prover to get $[\![t^{[e]}]\!]_i$
		$\Delta r^{[e]} = y - \sum_{i\neq\ell_e} [\![r^{[e]}]\!]_i$
		$h_e = \mathcal{H}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$
		From $\Delta r^{[e]}$, deduce $\Delta t^{[e]}$.
		$[\![t]\!]^{[e]} = t - \Delta t^{[e]} - \sum_{i\neq\ell_e} [\![t]\!]_i^{[e]}$
		$h'_e = \mathcal{H}_3(\tilde{x}^{[e]}, [\![t]\!]^{[e]})$
		Check $h = \mathcal{H}_2(h_1, \ldots, h_M)$
		Check $h' = \mathcal{H}_4((h'_e)_{e\in J})$
		Return 1

Protocol 6: Zero-knowledge argument for Subset Sum Problem via MPC-in-the-Head with rejection, using cut-and-choose strategy to prove binarity.

Let us recall that the couples $(\mathsf{seed}_i, \rho_i)$ are sampled using a TreePRG, sending $(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\neq\ell_e}$ costs at most $\lambda \log_2(N)$ bits by iteration. The communication cost (in bits) of the protocol is then

$$\textsc{Size} = 4\lambda + \lambda\tau \log_2 \frac{M}{\tau} + \tau\left[n \log_2(A-1) + n + \lambda \log_2 N + 2\lambda\right].$$

Here again, the obtained size is independent of the modulus q (and of the size of the integers $\{w_j\}, t$).

4.3.4 Security proofs for Protocol 6

The following theorems state the completeness, zero-knowledge and soundness of Protocol 6.

Theorem 7 (Completeness). *A prover P who knows a solution x to the subset sum instance $(w,t) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and who follows the steps of Protocol 6 convinces the verifier V with probability*

$$\left(1 - \frac{1}{A}\right)^{\tau n}.$$

Theorem 8 (Honest-Verifier Zero-Knowledge). *Let the* PRG *used in Protocol 6 be (t, ε_{PRG})-secure and the commitment scheme* Com *be (t, ε_{Com})-hiding. Then, there exists an efficient simulator Sim which, given random challenges J and L outputs a transcript which is $(t, \tau\varepsilon_{PRG} + \tau\varepsilon_{Com})$-indistinguishable from a real transcript of Protocol 6.*

Theorem 9 (Soundness). *Suppose that there is an efficient prover $\tilde{P}$ that, on input (w,t), convinces the honest verifier V on input $(w,t))$ to accept with probability*

$$\tilde{\epsilon} := \Pr[\langle \tilde{P}(w,t), V(w,t)\rangle = 1] > \epsilon$$

for a soundness error ϵ equal to

$$\max_{M-\tau \le k \le M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau} N^{k-M+\tau}} \right\}.$$

Then, there exists an efficient probabilistic extraction algorithm E that, given rewindable black-box access to $\tilde{P}$, produces either a witness x such that $t = \langle w, x\rangle$ and $x \in \{0,1\}^n$, or a commitment collision, by making an average number of calls to $\tilde{P}$ which is upper bounded by

$$\frac{4}{\tilde{\epsilon} - \epsilon}\left(1 + \tilde{\epsilon}\frac{8M}{\tilde{\epsilon} - \epsilon}\right).$$

Let us denote abort the event when Protocol 6 aborts. By exactly the same reasoning as in Subsection 4.3.2 (but here the protocol aborts if any of the τ iterations aborts), we have

$$\Pr[\text{abort} \mid X = x] = 1 - \left(1 - \frac{1}{A}\right)^{n\tau}$$

and

$$\Pr[\text{abort}] = 1 - \left(1 - \frac{1}{A}\right)^{n\tau}. \tag{4.9}$$

Proof. (Theorem 7) For any sampling of the random coins of P and V, if the computation described in the protocol is honestly performed and if there is *no abort*, all the checks of V pass. The completeness probability is hence of $1 - \Pr[\text{abort}]$, which from (4.9) implies the theorem statement. □

Proof. (Theorem 8) Before building the desired simulator (*i.e.* an algorithm that outputs transcripts that are indistinguishable from real transcripts without knowing the secret), let us first show the independence between the secret x and some events and values that can be observed from the transcript.

- The abortion event abort must be independent of the secret x, *i.e.*

  $$\Pr[\text{abort}|x] = \Pr[\text{abort}],$$

  it ensures that the fact to abort does not leak any information. It has been demonstrated with equation (4.9).

- When there is no abort, the transcript includes some values computed from the secret. By the same reasoning than in the proof of Theorem 5, we get that the coordinates of $r^{[e]} - [\![r^{[e]}]\!]_{i^*}$ follow the uniform distribution in $[-A+2, 0]$ and that $y^{[e]} = r^{[e]} - [\![r^{[e]}]\!]_{i^*}$ (together with the occurrence of ¬abort and the shares $\{[\![r^{[e]}]\!]_i\}_{i \neq i^*}$) does not leak any information about the vector $r^{[e]}$. Thus since $r^{[e]}$ is uniformly sampled in $\{0,1\}^n$, the value of $\tilde{x}^{[e]}$ is independent of the secret x.

Let us now describe the simulator Sim who has oracle access to some probabilistic-polynomial time $\tilde{\mathsf{V}}$, and works as follows (we keep the notation from Protocol 6):

Simulator Sim:

1. Sample $J \xleftarrow{\$} \{J \subset [1,M]; |J| = \tau\}$ and $L = \{\ell_e\}_{e\in J} \xleftarrow{\$} [1,N]^\tau$ uniformly at random (as an honest verifier).
2. Sample $\mathsf{mseed}^{[0]} \xleftarrow{\$} \{0,1\}^\lambda$

3. $(\mathsf{mseed}^{[e]})_{e\in[1,M]} \leftarrow \mathrm{TreePRG}(\mathsf{mseed}^{[0]})$
4. For $e \in [1,M]\backslash J$, follow honestly the protocol since it does not need to know the secret and deduce h_e.
5. For $e \in J$,
   - Compute $(\mathsf{seed}_1^{[e]}, \rho_1^{[e]}), \ldots, (\mathsf{seed}_N^{[e]}, \rho_N^{[e]})$ with $\mathrm{TreePRG}(\mathsf{mseed}^{[e]})$.
   - For each party $i \in [1,N]\backslash\{\ell_e\}$: $[\![r^{[e]}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i^{[e]})$, $\mathsf{com}_i^{[e]} = \mathrm{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$
   - Sample $\tilde{x}^{[e]} \xleftarrow{\$} \{0,1\}^n$, $y^{[e]} \xleftarrow{\$} [-A+2, 0]^n$, and $\Delta r^{[e]} = y^{[e]} - \sum_{i\neq\ell_e} [\![r^{[e]}]\!]_i$.
   - Sample a random commitment $\mathsf{com}_{\ell_e}^{[e]}$.
   - Simulate the computation of all the parties $i \neq \ell_e$ to get $\{[\![t]\!]_i^{[e]}\}_{i\neq\ell_e}$ and $\Delta t^{[e]}$.
   - Adapt the outputs of the party ℓ_e: $[\![t]\!]_{\ell_e}^{[e]} = t^{[e]} - \Delta t^{[e]} - \sum_{i\neq\ell_e} [\![t]\!]_i^{[e]} \mod q$
   - Compute $h_e = \mathcal{H}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$, $h'_e = \mathcal{H}_3(\tilde{x}^{[e]}, [\![t]\!]^{[e]})$
6. Compute $h = \mathcal{H}_2(h_1, \ldots, h_M)$, $h' = \mathcal{H}_4((h'_e)_{e\in J})$
7. Abort with probability
$$1 - \left(1 - \frac{1}{A}\right)^{n\tau}.$$
8. Outputs the transcript
$$\left(h, h', (\mathsf{mseed}^{[e]})_{e\in[1,M]\backslash J}, ((\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\neq\ell_e}, \mathsf{com}_{\ell_e}^{[e]}, y^{[e]}, \tilde{x}^{[e]})_{e\in J}\right).$$

When no abortion occurs, the output transcript is identically distributed to the genuine transcript except for commitment of the party ℓ_e in each execution $e \in J$. Distinguishing them means breaking the commitment hiding property or the PRG security. □

Proof. (Theorem 9) Let us first how to extract the subset sum solution x from a few transcripts satisfying specific conditions. We will then show how to get such transcripts from rewindable black-box access to $\tilde{\mathsf{P}}$.

Transcripts used for extraction. We assume that we can extract three transcripts

$$T_i = (\textsc{Com}^{(i)}, \textsc{Ch}_1^{(i)}, \textsc{Rsp}_1^{(i)}, \textsc{Ch}_2^{(i)}, \textsc{Rsp}_2^{(i)}) \quad \text{for } i \in \{1,2,3\}, \tag{4.10}$$

from $\tilde{\mathsf{P}}$, with $\textsc{Ch}_1^{(i)} := J^{(i)}$, $\textsc{Ch}_2^{(i)} := \{\ell_j^{(i)}\}_{j\in J^{(i)}}$, which satisfy:

1. $\textsc{Com}^{(1)} = \textsc{Com}^{(2)} = \textsc{Com}^{(3)} = h$,
2. there exists $j_0 \in (J^{(1)} \cap J^{(2)}) \setminus J^{(3)}$ s.t. $\ell_{j_0}^{(1)} \neq \ell_{j_0}^{(2)}$
3. T_1 and T_2 are success transcripts (*i.e.* which pass all the tests of V),
4. $\mathsf{seed}^{[j_0]}$ from $\textsc{Rsp}_1^{(3)}$ is consistent with the $(\sigma_i^{[j_0]}, s_i^{[j_0]})$ from T_1 and T_2.

Using these three transcripts, we next show that it is possible to extract a solution of the subset sum instance defined by w and t. We can assume that all the revealed shares are mutually consistent between the three transcripts because else we find a hash collision. So, we know all the shares for the iteration j_0 from T_1 and T_2.

Extraction of x from T_1, T_2 and T_3. For this part, we will only consider the variables of the form $(*)^{[j_0]}$, so we will omit the superscript for the sake of clarity. In the following, we will denote $\mathcal{V}_{T_i}$ the set of checked equations at the end of the transcript with T_i for $i \in \{1,2,3\}$.

Let us define $x' := \Delta x + \sum_{i=1}^N [\![x]\!]_i$. We simply return x' as a candidate solution for x. Thanks to the multi-party computation, we know

- The sharing $[\![t]\!]$ encodes t: $t = \Delta t + \sum_{i=1}^N [\![t]\!]_i$.

- We have $[\![t]\!] = \sum_{j=1}^{n} w_j [\![x_j]\!]$, *i.e.*

$$\begin{cases} \forall i \in [1, N], [\![t]\!]_i = \sum_{j=1}^{n} w_j [\![x_j]\!]_i, \\ \Delta t = \sum_{j=1}^{n} w_j \Delta x_j. \end{cases}$$

- We have $[\![x]\!] = (1 - \tilde{x}) \circ [\![r]\!] + \tilde{x} \circ (1 - [\![r]\!])$:

$$\begin{cases} \forall i \in [N], [\![x]\!]_i = (1 - \tilde{x}) \circ [\![r]\!]_i + \tilde{x} \circ (-[\![r]\!]_i), \\ \Delta x = (1 - \tilde{x}) \circ \Delta r + \tilde{x} \circ (1 - \Delta r). \end{cases}$$

So we deduce that

$$\begin{aligned} \sum_{j=1}^{n} w_j x'_j &= \sum_{j=1}^{n} w_j (\Delta x_j + \sum_{i=1}^{N} [\![x_j]\!]_i) \\ &= \sum_{j=1}^{n} w_j \Delta x_j + \sum_{i=1}^{N} \sum_{j=1}^{n} w_j [\![x_j]\!]_i \\ &= \Delta t + \sum_{i=1}^{N} [\![t]\!]_i = t \end{aligned}$$

and

$$\begin{aligned} x' &= \Delta x + \sum_{i=1}^{N} [\![x]\!]_i \\ &= ((1 - \tilde{x}) \circ \Delta r + \tilde{x} \circ (1 - \Delta r)) + \sum_{i=1}^{N} ((1 - \tilde{x}) \circ [\![r]\!]_i + \tilde{x} \circ (-[\![r]\!]_i)) \\ &= (1 - \tilde{x}) \circ (\Delta r + \sum_{i=1}^{N} [\![r]\!]_i) + \tilde{x} \circ (1 - \Delta r - \sum_{i=1}^{N} [\![r]\!]_i) \\ &= (1 - \tilde{x}) \circ r + \tilde{x} \circ (1 - r) \end{aligned}$$

where $r := \Delta r + \sum_{i=1}^{N}$.

From $\mathcal{V}_{T_3}$, we get that r is a binary vector. Since $\tilde{x}$ is binary by definition, thanks to the above relation, we deduce that the vector x' is a binary vector.

Since x' verifies $t = \sum_{j=1}^{n} w_j x'_j$ and is a binary vector, it is a solution of the subset sum instance (w, t).

Extraction of T_1, T_2 and T_3 from $\tilde{P}$. We can use exactly the same extractor E as defined in the Appendix E of [FJR23]. It defines an extractor which produces the wanted transcripts by making in average at most

$$\frac{4}{\tilde{\epsilon} - \epsilon} \left(1 + \tilde{\epsilon} \frac{8 \cdot M}{\tilde{\epsilon} - \epsilon}\right)$$

calls to $\tilde{\mathsf{P}}$, which concludes the proof. □

4.3.5 Decreasing the rejection rate

The two above protocols have a rejection rate around $\tau n / A$ which implies that we must take $A = \Theta(\tau n)$ to obtain a constant (small) rejection rate. In practice, this results in a significant increase in the communication cost. Let us for instance consider Protocol 5 with $(\tau, N, A) = (16, 280, 2^{13})$. For this setting, the proof size is about 15.6 KB for a rejection rate of 0.394. If we increased A to get a rejection rate below 0.003, we should take $A = 2^{21}$ and the proof size would be 23.6 KB.

A better strategy consists in allowing the prover to abort a few of the τ iterations. Let us assume that the verifier accepts the proof if the prover can answer to $\tau - \eta$ challenges among the τ iterations. This slightly increases the soundness error, but it can also significantly decrease the global rejection rate. If we denote p_{rej} the probability that an iteration aborts, then the global rejection rate of this strategy is given by

$$1 - \sum_{i=0}^{\eta} \binom{\tau}{i} (1 - p_{\text{rej}})^{\tau - i} p_{\text{rej}}^{i} \,. \tag{4.11}$$

At the same time, the soundness error for Protocol 5 becomes

$$\sum_{i=0}^{\eta} \binom{\tau}{i} (1-\epsilon)^i \epsilon^{\tau-i}$$

where $\epsilon = \frac{1}{N} + \left(1 - \frac{1}{N}\right)\frac{1}{q'}$ is the soundness error of a single iteration. Using this strategy with $\tau = 20$ and $\eta = 3$, the proof size is of 16.7 KB for a rejection rate of 0.003 (instead of 23.6 KB with the naive strategy).

The same strategy also applies to Protocol 6. The rejection rate is also given by Equation (4.11) while the soundness error becomes

$$\max_{M-\tau \le k \le M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau}} \sum_{i=0}^{\eta} \left[\binom{k-M+\tau}{i} \left(1 - \frac{1}{N}\right)^i \left(\frac{1}{N}\right)^{k-M+\tau-i} \right] \right\}.$$

In any case, the prover always answers to at most $\tau - \eta$ challenges of the verifier (even if the prover aborts less than η among the τ iterations) so that the communication cost is roughly that of $\tau - \eta$ iterations. Additionally, for each unanswered challenge, the prover must further send two hash digests to enable the verifier to recompute and check h and h'. Thus the new proof size (in bits) for Protocol 5 is

$$\textsc{Size}_\eta = 4\lambda + \eta 4\lambda + (\tau - \eta)\left[n(\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda\right] ,$$

while the new proof size (in bits) for Protocol 6 is

$$\textsc{Size}_\eta = 4\lambda + \eta 4\lambda + \lambda\tau \log_2 \frac{M}{\tau} + (\tau - \eta)\left[n \log_2(A-1) + n + \lambda \log_2 N + 2\lambda\right] .$$

We note that in practice, given a target security level and a target rejection probability, one needs to use a slightly increased τ (or N) to compensate for the loss in terms of soundness. While this shall slightly increase the proof size, the above approach (with $\eta > 0$) still provides better trade-offs than the original approach ($\eta = 0$).

4.4 Instantiations and Performances

4.4.1 Subset Sum instances

We recall in this section known techniques to solve the modular subset sum problem (SSP) defined by (4.1). It is well-known that the hardness of an SSP instance depends greatly on its *density* defined as $d = n/\log_2 q$. If the SSP instance is too sparse (e.g. $d < 1/n$) or too dense (*e.g.* $d > n/\log^2 n$) then the problem can be solved in polynomial time (see e.g. [CJL+92] and references therein). We shall therefore only consider SSP instances with density $d \simeq 1$ (*i.e.* $q \simeq 2^n$) which are arguably the hardest ones [IN96].

In this case, simple algorithms exist based on brute force enumeration at $O(2^n)$ time and constant space, or time-space tradeoff [HS74] with $O(2^{n/2})$ time and space complexities. The first non-trivial algorithm was published by Schroeppel and Shamir [SS81] with time complexity $O(2^{n/2})$ and space complexity $O(2^{n/4})$. Later, faster algorithms were proposed with similar time and space complexities, e.g. $\tilde{O}(2^{0.337n})$ by Howgrave-Graham and Joux [HJ10]and $\tilde{O}(2^{0.283n})$ by Bonnetain, Bricout, Schrottenloher and Shen [BBSS20]. The latter algorithms neglect the cost to access an exponential memory but even with this optimistic assumption, for $n = 256$, all known algorithms require at least a time complexity lower-bounded by 2^{128} operations or memory of size at least 2^{72} bits. There also exists a vast literature on quantum algorithms for solving the SSP (see [BBSS20] and references therein). The best (heuristic) quantum complexity from [BBSS20] has time complexity $\tilde{O}(2^{0.216n})$ and thus requires about 2^{64} quantum operations and quantum memory for $n = 256$. In the following, we, therefore, consider the efficiency of our protocols for $n = 256$.

4.4.2 Zero knowledge protocols

Let us consider the subset sum problem with $n = 256$. We propose in Table 4.1 several sets of parameters for our two protocols which target a security of 128 bits. We provide two kinds of instantiations to give the reader an idea of the obtained performance while changing the number of parties. The first ones correspond to instantiations with fast computation. The second ones correspond to instantiations that achieve smaller communication costs but slower computation. For each setting, we suggest two parameter sets: one achieving a rejection rate around 0.4 and the other one achieving a rejection rate between 0.001 and 0.004.

We provide in Table 4.1 the performance of the other zero-knowledge protocols proving the knowledge of an SSP solution. The only other protocol designed for the subset sum problem is Shamir's one [Sha86]. We can also compare these protocols with [LNSW13] which is an adaptation of Stern's protocol to the ISIS (inhomogeneous short integer solution) problem. The remaining articles in the literature about proofs for the ISIS problem are restricted to the case where the modulus q is prime. We add Beullens' protocol [Beu20] for ISIS with prime q to the comparison.

We provide in Appendix B the performances of the obtained signatures when applying the Fiat-Shamir transform [FS87] to our protocols.

Protocol	Parameters					Proof size	Rej. rate	Soundness err.
	τ	η	N	A	M			
Shamir [Sha86]	219	-	-	-	-	1186 KB	-	128 bits
[LNSW13]	219	-	-	-	-	2350 KB	-	128 bits
Beullens [Beu20]	14	-	1024	-	4040	122 KB	-	128 bits
Protocol 5 (batching)	26	0	32	2^{14}	-	25.7 KB	0.334	130 bits
Protocol 5 (batching)	31	3	32	2^{14}	-	27.9 KB	0.001	128 bits
Protocol 6 (C&C)	27	0	32	2^{14}	462	17.4 KB	0.344	128 bits
Protocol 6 (C&C)	33	3	32	2^{14}	470	19.6 KB	0.002	128 bits
Protocol 5 (batching)	17	0	256	2^{13}	-	16.6 KB	0.412	135 bits
Protocol 5 (batching)	21	3	256	2^{13}	-	17.7 KB	0.004	133 bits
Protocol 6 (C&C)	19	0	256	2^{13}	954	13.0 KB	0.448	128 bits
Protocol 6 (C&C)	24	3	256	2^{14}	952	15.4 KB	0.001	128 bits

Tab. 4.1: Comparison of state-of-the-art zero-knowledge protocols for proving the knowledge of an SSP instance (with $n = 256$ and $q \approx 2^{256}$).

4.4.3 Comparison with generic techniques

We compare our scheme with efficient generic techniques to prove the knowledge of an SSP solution. Among those techniques, we consider SNARKs (e.g. [Gro16a]), "compressed" proof systems such as *Bulletproofs* [BBB+18] and STARKs (*e.g.* [BBHR18]). For the sake of accuracy, we split the notation for the security level of the subset sum instance, denoted κ, and of the zero-knowledge argument, denoted λ. Adapting the analysis of Subsection 4.2.5 to this setting, we get a communication cost of $\Theta(\lambda^2 + \lambda\kappa)$ for our protocols.

The asymptotic size of [Gro16a] arguments is roughly[1] $\Omega(\lambda^3)$ which is asymptotically larger than ours, but for $\kappa = \lambda = 128$, these arguments will be shorter than ours (within the range of 700–800 bytes). Using *Bulletproofs* [BBB+18], one can obtain an asymptotic communication cost of $\Omega(\log(\kappa)(\lambda+\kappa))$, and about 600 bytes for $\kappa = \lambda = 128$. Although SNARKs and "compressed" proof systems give shorter arguments than ours for $\kappa = \lambda = 128$, they both require stronger and non post-quantum computational assumptions. In particular, [Gro16a] requires a trusted setup and a non-falsifiable assumption, while Bulletproofs rely on the algebraic group model in their non-interactive version [GOP+22]. In comparison, the security of our arguments only relies on weak post-quantum assumptions (PRG, collision-resistant hash functions).

Regarding STARKs [BBHR18], their security assumptions are similar to ours. When applying STARKs to the subset sum problem, one gets arguments of size $\Omega(\lambda^2 \log^2 \kappa)$, which is larger than ours.[2]

4.5 Further Applications

As illustrated on the subset sum problem, our technique of sharing over the integers with rejection is –more generally– instrumental to a context of a secret vector $s \in \mathbb{Z}_q^n$ with small coefficients. Since the communication cost of our protocols is independent of the size q of the ring $\mathbb{Z}_q$, the gain in communication is higher when the modulus q is high. But it does not need to have a modulus as high as in the subset sum problem to be interesting. In the three subsections, we present the performance of our schemes with the sharing over the integers on three other applications with moderate-size modulus:

- to prove the knowledge of a solution of an ISIS problem instance,
- to prove the knowledge of a secret key and plaintexts matching a set of FHE ciphertexts,
- to construct an efficient digital signature based on Boneh-Halevi-Howgrave-Graham pseudo-random function.

Another advantage of the sharing on the integers is that we can perform any operation on it with any modulus. We used this property in one of our protocols to check multiplication triples in a smaller field. This property can be also useful when we want to prove that the same secret vector verifies many relations using distinct modulus.

[1] This is due to sub-exponential attacks on the discrete logarithm in the target group which also impacts the size of elements of the second group of the bilinear structure.

[2] The λ^2 factor is obtained by λ for the hash digest size times λ for the number of evaluation points in the FRI protocol (which scales with the soundness error). The $\log^2 \kappa$ factor comes from the size κ of the program verifying the SSP instance.

4.5.1 Short Integer Solution Problem

Given a matrix $A \in \mathbb{Z}^{m \times n}$ and a vector $u \in \mathbb{Z}_m$, the inhomogenous short integer solution (ISIS) problem consists in finding a vector $s \in \mathbb{Z}^n$ with small coefficients such that

$$As = u \bmod q.$$

The Ling-Nguyen-Stehlé-Wang protocol [LNSW13], which is an adaptation of Stern's protocol, has been for a long time the only zero-knowledge exact protocol which proves the knowledge of a solution of an ISIS instance. Other protocols existed but they were only relaxed proofs, *i.e.* they prove the knowledge of an s' and c satisfying $As' = cu \bmod q$. These protocols can be useful in some contexts, but they are not suited to prove the exact statement.

Recently, new exact proofs [BLS19, ENS20, LNS21, BN20a, Beu20] have been published. However, all these new protocols require an assumption on the modulus q to work: some of them only require that q is a prime number when the others require that q is an NTT-friendly prime number. In the state of the art, the only protocol which works for any q (even when q is not a prime) is [LNSW13].

We can adapt our protocols of Section 4.3 to the case of the ISIS problem. The linear constraint "$As = u$" is free in communication as it was the case for "$t = \langle w, x \rangle$" for the subset sum problem (see Section 4.2.2). The hard part is to prove that the secret s satisfies $\|s\|_\infty \leq \beta$ for some bound β. To proceed, we decompose s as $k := \lceil \log_2(2\beta + 1) \rceil$ vectors $(s_0, \ldots, s_{k-1})$ of $\{0,1\}^n$ such that

$$s = \sum_{i=0}^{k-2} 2^i s_i + (2\beta - 2^{k-1} + 1) s_{k-1} - \beta \,. \tag{4.12}$$

If all vectors s_i belong to $\{0,1\}$, the above relation gives that $\|s\|_\infty \leq \beta$. So we just need to give the sharing $\{[\![s_i]\!]\}_{i \in [0,k-1]}$ to the MPC protocol instead of $[\![s]\!]$. The latter can then check that $\{[\![s_i]\!]\}_{i \in [0,k-1]}$ are binary vectors and that $A[\![s]\!]$ corresponds to u modulo q where $[\![s]\!]$ is recovered by linearity of Equation (4.12). The proof sizes of the resulting protocols are given by the formulae as before, we just need to consider that the length of the secret is nk (instead of n).

We compare our protocols with the state of the art in Table 4.2 on the two following ISIS problems:

1. $\|s\|_\infty \leq 1$, $m = 1024$, $n = 2048$, $q \approx 2^{32}$
2. Binary s, $m = 512$, $n = 4096$, $q \approx 2^{61}$

For both instances, we have $kn = 4096$. For our protocols, we choose the following parameters:

- Protocol 1 (batch product verification):

$$A = 2^{16},\ N = 128,\ q' \approx A,\ \tau = 23,\ \eta = 3.$$

- Protocol 2 (cut-and-choose strategy):

$$A = 2^{16},\ N = 256,\ q' \approx A,\ M = 952,\ \tau = 24,\ \eta = 3.$$

We can remark that our protocols have the same communication cost for both instances. It comes from the fact that their proof size is independent of the modulus q. Even when q is prime (and larger than 2^{32}), our Protocol 6 (with the cut-and-choose phase) has smaller communication cost than Beullens' protocol and this while taking less aggressive parameters towards size against speed (the parameters used in [Beu20] are $(\tau, M, N) = (14, 4040, 2^{10})$). We also observe that our protocols achieve proof sizes which are more than 10 times smaller than those of [LNSW13], the only previous protocol supporting any modulus q.

4.5.2 Fully Homomorphic Encryption

Our zero-knowledge protocols also find application to fully homomorphic encryption (FHE). We can indeed adapt our protocols to prove the knowledge of a secret key matching a set of FHE-encrypted plaintexts. We elaborate on this application hereafter for the particular case of TFHE (Torus FHE) [CGGI20] which is currently one of the FHE schemes with the best performances in practice.

For some $q \in \mathbb{N}$, let $\mathbb{T}_q = q^{-1}\mathbb{Z}/\mathbb{Z}$ be the discretized torus with q elements, i.e. the submodule of the real torus with representative $\{i/q \ ; \ i \in \mathbb{Z}_q\}$ [Joy21]. In practice, q is often chosen to be 2^{32} or 2^{64} in order to match the word-size and arithmetic operations of common CPUs. For this reason, we shall consider that q is a power of 2 in the following (although the described application can be easily generalized to any q). TFHE relies on so called TLWE (Torus Learning With Error) encryption. Let $p \mid q$ and $\delta = q/p$. The plaintext space is defined as $\mathbb{Z}_p$ while the key space is defined as

Protocol	Year	Any q	Instance 1		Instance 2	
			Proof Size	Rej. Rate	Proof Size	Rej. Rate
[LNSW13]	2013	✓	3600 KB	-	8988 KB	-
[BN20a]	2020	q prime	-	-	4077 KB	-
[Beu20]	2020	q prime	233 KB	-	444 KB	-
Our Protocol 1	2022	✓	291 KB	0.04	291 KB	0.04
Our Protocol 2	2022	✓	184 KB	0.05	184 KB	0.05
[BLS19]	2019	q prime + NTT	384 KB	0.92		
[ENS20]	2020	q prime + NTT	47 KB	0.95		
[LNS21]	2021	q prime + NTT	33.3 KB	0.85		
Aurora [BCR+19]	2019	q prime + NTT	71 KB	-		
Ligero [AHIV17]	2017	q prime + NTT	157 KB	-		

Tab. 4.2: Comparison with the existing exact protocols which prove the knowledge of the solution of an ISIS instance.

$\{0,1\}^n \subset \mathbb{Z}^n$. Let $s = (s_1, \ldots, s_n) \in \{0,1\}^n$ be a secret key. The TLWE encryption of a plaintext $\mu \in \mathbb{Z}_p$ under the secret key s and with *error* $e \in \mathbb{Z}$ is defined as

$$c = (a_1, \ldots, a_n, b) \in \mathbb{T}_q^{n+1} \quad \text{where} \quad \begin{cases} \mu^* = \frac{\delta\mu + e \bmod q}{q} \in \mathbb{T}_q \\ b = \sum_{j=1}^n s_j a_j + \mu^* \end{cases}$$

The a_i's are random elements of $\mathbb{T}_q$ which are sampled at encryption time or which arise from the homomorphic operations between other ciphertexts. The value $e \in \mathbb{Z}$ is the error which must satisfies $|e| < \delta/2$ to ensure the correctness of the decryption.

Proving the knowledge of a key s and plaintext μ for which $c = (a_1, \ldots, a_n, b)$ is a correct TLWE encryption of μ under s can be achieved by proving the knowledge of a binary vector

$$x = (s_1, \ldots, s_n) \mid (\mu_1, \ldots, \mu_{\ell_p}) \mid (e_1, \ldots, e_{\ell_e})$$

where $\ell_p = \log_2 p$ and ℓ_e is such that $e \in [-2^{\ell_e - 1}, 2^{\ell_e - 1} - 1]$, and which satisfies

$$\sum_{i=1}^{n} \bar{a}_i s_i + \sum_{i=1}^{\ell_p} (2^{i-1}\delta)\mu_i + \sum_{i=1}^{\ell_e} (2^{i-1}) e_i = \bar{b} + 2^{\ell_e - 1} \pmod q$$

where $\bar{a}_i \in \mathbb{Z}$ (resp. $\bar{b} \in \mathbb{Z}$) is the integer such that $a_i = \bar{a}_i/q \in \mathbb{T}_q$ (resp. $b = \bar{b}/q \in \mathbb{T}_q$) and where the error is $e := -2^{\ell_e - 1} + \sum_{i=1}^{\ell_e} (2^{i-1}) e_i$. The application of our protocols to this context is immediate. We note that the secret binary vector is of size $n' = n + \ell_p + \ell_e$ when the underlying plaintext must remain secret while it is of size $n' = n + \ell_e$ if the plaintext is public. In the latter case, the value of the sum is $t = \bar{b} + 2^{\ell_e - 1} - \mu$. We can also use our protocols to prove the knowledge of a secret key and a set of plaintexts matching a set of ciphertexts. For m ciphertexts, we obtain m linear relations with a binary vector of size $n' = n + m(\ell_p + \ell_e)$ (or $n' = n + m\ell_e$ in the public plaintext setting).

Remark 5. *Proving the knowledge of a single key-plaintext pair matching a given ciphertext might not be relevant on its own. Indeed, for the typical parameters given above, the obtained SSP instance might not be hard (i.e. finding a solution is not hard while finding the original key-plaintext pair is still hard). However, such proof is still useful whenever proving additional properties involving the underlying secret key and/or plaintext. In such contexts, finding a solution to the SSP instance which does not match the original key-plaintext pair is useless.*

According to [Joy21], typical parameters for a TLWE encryption are $q = 2^{32}$ or $q = 2^{64}$ and $n = 630$. Depending on the exact message space and error space, we have $n' \in [n, n + \log_2 q]$. Table 4.3 gives the obtained communication cost for proving the knowledge of the key (and plaintexts) corresponding to 1, 64 and 1024 TLWE ciphertexts using our protocols (assuming $q = 2^{64}$ and $\ell_e + \ell_p = 64$). For the sake of comparison, we also give the communication obtained with Shamir's protocol [Sha86]. We note that the latter and the LNSW protocol [LNSW13] are the only previous protocols which can work with such values of q and the LNSW protocol is always heavier than Shamir's in this context. We observe that our protocols always gain more than a factor 10 (for Protocol 5) and 20 (for Protocol 6) for the obtained communication cost compared to Shamir's protocol.

Besides TFHE, our proof techniques are also well suited to prove the correctness of a ciphertext produced by a public-key FHE encryption using the Rothblum transform [Rot11]. The latter can transform any secret-key FHE scheme into a public-key FHE scheme. The public key is built as a set of ciphertexts $c_1, \ldots, c_n$ each encrypting 0. Then to encrypt a plaintext μ, one draws a random secret vector $(x_1, \ldots, x_n) \in \{0,1\}^n$ and computes the encryption of μ as $\mu + \sum_{i=1}^n c_i$. (Here we implicitly assume that the ciphertexts are malleable as $\mathsf{Enc}(0) + \mu = \mathsf{Enc}(\mu)$ but the Rothblum transform

Protocol	Parameters					Proof size	Rej. rate	Soundness err.
	τ	η	N	A	M			
1 ciphertext								
Shamir [Sha86]	219	-	-	-	-	845 KB	-	128 bits
Protocol 5 (batching)	19	2	256	2^{15}	-	46.1 KB	0.007	128 bits
Protocol 6 (C&C)	24	3	256	2^{15}	952	34.0 KB	0.002	128 bits
64 ciphertexts								
Shamir [Sha86]	219	-	-	-	-	8.48 MB	-	128 bits
Protocol 5 (batching)	19	2	256	2^{18}	-	356 KB	0.005	129 bits
Protocol 6 (C&C)	24	3	256	2^{18}	952	236 KB	0.001	128 bits
1024 ciphertexts								
Shamir [Sha86]	219	-	-	-	-	77.9 MB	-	128 bits
Protocol 5 (batching)	19	2	256	2^{22}	-	5.90 MB	0.003	129 bits
Protocol 6 (C&C)	24	3	256	2^{21}	952	3.65 MB	0.006	128 bits

Tab. 4.3: Comparison of ZK protocols for TFHE decryption.

can also work more generally without this property.) In other words, this generic public-key FHE encryption process consists in building an SSP instance and our proof techniques directly apply to this context.

We stress that the performances reported in Table 4.3 are in the context of a relatively small q (64 bits). Although the results are already promising compared to the previous schemes, we expect this comparison to be much more in favor of our protocols in contexts where q is larger since the size of our proofs is independent of q. In particular, it may be interesting to apply our techniques to the SPDZ framework [DPSZ12] which is the state-of-the-art protocol for dishonest-majority MPC (with computational security). In the offline phase of SPDZ, parties have to jointly produce a zero-knowledge argument of plaintext knowledge for the Brakerski, Gentry, and Vaikuntanathan [BGV14] or the Brakerski/Fan-Vercauteren [Bra12, FV12] homomorphic encryption schemes. Recent works [KPR18, CKR+20] were devoted to providing communication-efficient such arguments (with slack) and since the modulus bit-lengths are within the range $[250, 700]$, our techniques look promising to provide short exact arguments in these contexts.

4.5.3 Digital signatures from Boneh-Halevi-Howgrave-Graham PRF

As another application, we present a short and efficient candidate post-quantum signature scheme based on an elegant pseudo-random function (PRF) proposed by Boneh, Halevi, and Howgrave-Graham in 2001 [BHH01].

Let p be a public m-bit prime number that defines the PRF message space as $\mathbb{Z}_p$. A secret key for the PRF is an element $x \in \mathbb{Z}_p$ picked uniformly at random. We denote $\text{MSB}_\delta(t)$ the δm most significant bits of an m-bit element $t \in \mathbb{Z}_p$ [3]. The value of the PRF on the message $m \in \mathbb{Z}_p$ for the secret-key $x \in \mathbb{Z}_p$ is $F_x(m) = \text{MSB}_\delta((x+m)^{-1} \bmod p)$.

Our signature scheme follows the blueprint of most signatures based on the MPCitH paradigm since the proposal of Picnic [CDG+17]: the public key is made of the outputs of Boneh *et al.*'s PRF on t public messages in $[1, t]$, *i.e.* the δm-bit elements $y_1, \ldots, y_t$ such that

$$y_i := \text{MSB}_\delta((x+i)^{-1} \bmod p) \text{ for } i \in [1, t]$$

and the signature consists of a non-interactive proof of knowledge of $x, z_1, \ldots, z_t$ (parametrized by the signed message using the Fiat-Shamir heuristic) such that

$$(x+1)(2^{(1-\delta)m} y_1 + z_1) \equiv \cdots \equiv (x+t)(2^{(1-\delta)m} y_t + z_t) \equiv 1 \bmod p \tag{4.13}$$

$$\text{and} \qquad z_1, \ldots, z_t \in [0, 2^{(1-\delta)m} - 1] \tag{4.14}$$

where $z_1, \ldots, z_t$ are the $(1-\delta)m$ least significant bits of $(x+1)^{-1} \bmod p, \ldots, (x+t)^{-1} \bmod p$. Note that the condition (4.14) on the size of the z_i's is fundamental since otherwise, it is easy for an attacker to find a witness.

In our applications, the values of t and δ are chosen to prevent all known classical attacks and target a 128-bit security level.

Let's fix t, the number of outputs of the PRF. Then, to ensure that the equations (4.13) and (4.14) have a unique witness, we add the constraint $\delta \geq 1/t$ so that the t PRF outputs define (heuristically) the secret x uniquely. To avoid brute-force attacks from a single output of the PRF, at least 128 bits should remain hidden for each output, thus $m := \log p \geq \frac{128}{1-\delta}$. Otherwise, an attacker could reconstruct a possible PRF key matching with the first output and then test it by evaluating the other outputs with this candidate.

[3] We assume hereafter that $\delta m \in \mathbb{Z}$. Otherwise, one should take the nearest integer $\lfloor \delta m \rceil$ instead.

It is possible to apply generically the MPCitH paradigm to prove (4.13) and (4.14), but proving (4.14) seems inefficient (*e.g.* by using a binary decomposition and proving consistency). Instead, we can use our secret sharing over the integers for proving the knowledge of small z_i's by sharing them as a sum of "small" integers which directly proves that the z_i's are indeed small.

Proving Equation (4.13). Instead of proving the t products of (4.13) separately, the prover can batch them into a linear combination where coefficients $\gamma_1, \dots, \gamma_t$ are provided by the verifier, *i.e.* the prover proves the equation

$$\sum_{i=1}^{t} \gamma_i \left((x+i)(2^{(1-\delta)m} y_i + z_i) - 1 \right) = 0 \bmod p,$$

or equivalently,

$$x \left(\sum_{i=1}^{t} \gamma_i z_i \right) = - \sum_{i=1}^{t} \gamma_i \left(x 2^{(1-\delta)m} y_i + i 2^{(1-\delta)m} y_i + i z_i - 1 \right) \bmod p. \tag{4.15}$$

If one of the products is not equal to 1 in (4.13), then (4.15) is satisfied only with a probability of $\frac{1}{p}$. And to prove (4.15), one can use the protocol of [BN20a] with a single multiplication on $\mathbb{Z}_p$ (for the left-hand side of (4.15), the right-hand side being a linear combination of the witness). The resulting MPC protocol produces false positives with probability $1/p + (1-1/p)1/p := 2/p - 1/p^2$, and thus the obtained zero-knowledge argument has a soundness error of

$$\epsilon = \frac{1}{N} + \left(1 - \frac{1}{N}\right) \left(\frac{2}{p} - \frac{1}{p^2} \right).$$

Proving Equation (4.14). It remains to prove that z_i is in $[0, B-1]$ with $B = 2^{(1-\delta)m}$ in (4.14) for $i \in [1,t]$. To share z_i, we use our secret sharing over the integers of Section 4.2.2. Since the z_i are not binary but in a larger range, we need to adapt the rejection rules. Following exactly the same reasoning as in Section 4.2.2, we get that the prover must abort if there exists an index $j \in [1,t]$ for which $z_j - [\![z_j]\!]_{i^*} \geq 1$ or $z_j - [\![z_j]\!]_{i^*} \leq -A + B - 1$. The resulting rejection rate is given by

$$p_{\text{rej}} = 1 - \left(1 - \frac{B-1}{A}\right)^{t\tau} \approx t\tau \frac{B-1}{A}.$$

Even without proving anything on the range of z_j, the verifier knows that

$$\forall j \in [1,t], -A + B \leq z_j \leq A - 1$$

thanks to (4.2) (generalized). In practice, we settle this range, implying that there is a slack between the underlying hard problem and the proven statement. A malicious prover can use bigger values for z_i, and this is equivalent to ignoring some bits of y_i. A malicious prover can ignore up to $\log_2 \frac{A}{B} \approx \log_2 \frac{t\tau}{p_{\text{rej}}}$ bits for each PRF output, and thus it reduces the security of $t \log_2 \frac{t\tau}{p_{\text{rej}}}$ bits. A way to fix this security loss without increasing the size of p (and of the key) is to reveal a few more PRF outputs to guarantee that the key is still heuristically unique. In theory, this decreases the security but for state-of-the-art algorithms, this stays beyond the capacity of the best-known algorithms for small t. In fact, we need to reveal $\tilde{t} \geq t$ outputs of the PRF such that

$$\tilde{t}\delta m - \tilde{t} \log \left(\frac{\tilde{t}\tau}{p_{\text{rej}}} \right) > m.$$

In other words, since $\delta \geq \frac{1}{t}$, we adapt this constraint as

$$\delta \geq \frac{1}{\tilde{t}} + \frac{1}{m} \log_2 \left(\frac{\tilde{t}\tau}{p_{\text{rej}}} \right).$$

This leads to the scheme described as Protocol 15 (in Appendix C) with the communication cost (in bits):

$$4\lambda + \tau(\underbrace{\log_2 p}_{\Delta x} + \underbrace{\tilde{t} \log_2 A}_{\Delta a_i} + \underbrace{\log_2 p}_{\Delta c} + \underbrace{\log_2 p}_{\Delta \alpha} + \lambda \log_2 N + 2\lambda),$$

with soundness error (if interactive)

$$\epsilon = \frac{1}{N} + \left(1 - \frac{1}{N}\right) \left(\frac{2}{p} - \frac{1}{p^2} \right),$$

and with forgery security (if non-interactive)

$$\text{cost}_{\text{forge}} = \min_{\tau_1,\tau_2:\tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} p'^i (1-p')^{\tau-i}} + N^{\tau_2} \right\},$$

with $p' := 2/p - 1/p^2$.

We propose in Table 4.4 some parameters which target 128-bit security (based on the hardness of the so-called *modular inverse hidden number problem*) according to the current cryptanalysis state-of-the-art for Boneh *et al.*'s PRF. We can remark that the achieved signature sizes were at that time the first below 5 KB (for signatures based on MPC-in-the-Head paradigm), outperforming all the other signatures of this type (Picnic4 [KZ22], PorcRoast [Bd20], SDitH [FJR22], ...).

Parameters							Size	p_{rej}
$p \approx 2^m$	$\tilde{t}$	δ	B	A	N	τ		
$\approx 2^{229}$	3	$88/229$	2^{141}	2^{141+12}	256	16	4 916 B	0.012
$\approx 2^{186}$	4	$58/186$	2^{128}	2^{128+12}	256	16	4 860 B	0.016
$\approx 2^{175}$	5	$47/175$	2^{128}	2^{128+12}	256	16	5 074 B	0.019

Tab. 4.4: Parameter sets and achieved performances of the signature based on Boneh *et al.*'s PRF, for a 128-bit security.

Regarding the cryptanalysis, the security of Boneh *et al.*'s PRF has been extensively analyzed since 20 years [BHH01, LSSW12, BVZ12, XSH+19] and relies strongly on δ and the number of known PRF outputs. The first natural attack is the brute-force search on one output of the PRF as explained previously. We choose our parameter sets such that

$$(1-\delta)m > 128 \tag{4.16}$$

to prevent this attack.

The first described attack [BHH01] is with [BVZ12] the best known lattice-based attack with a small number of PRF outputs and require larger δ's than the ones we use. In order to mount them, an adversary has to perform an exhaustive search on the missing bits on several outputs. Let us focus on the attack of [BHH01]. The attacker first chooses $n > 1$ arbitrary ouputs among the $\tilde{t}$ ones. To run the attack, they need to have at least $\frac{2n+1}{3n+1}m$ bits for each output, thus they can exhaustively search the $n\left(\frac{2n+1}{3n+1} - \delta\right)m$ missing bits. For each candidate, the attacker applies the attacks of [BHH01] which consists in reducing a lattice of dimension $O(n)$. To prevent this attack against our parameter sets, we select m, $\tilde{t}$ and δ such that

$$\forall 1 < n \leq \tilde{t},\ n\left(\frac{2n+1}{3n+1} - \delta\right)m \geq 128. \tag{4.17}$$

Similarly, we can build an attack based on [BVZ12] with an exhaustive search to get the missing bits. To prevent this attack, we select m, $\tilde{t}$ and δ such that

$$\forall 1 < n \leq \tilde{t},\ n\left(\frac{2^{n-1}}{2^n - 1} - \delta\right)m \geq 128. \tag{4.18}$$

Following this discussion, we choose our parameters as follows: by considering $N = 256$ and $\tau = 16$, we first fix $\tilde{t}$, then we take m minimal such that there exists δ satisfying the constraints (4.16), (4.17) and (4.18) together with the constraint ensuring the uniqueness of the secret

$$\tilde{t}\delta m - \tilde{t}\log\left(\frac{\tilde{t}\tau}{p_{\text{rej}}}\right) > m.$$

For all parameters provided in Table 4.4 an exhaustive search on (at least) 128 bits has to be performed by the adversary in order to run the attacks from [BHH01, BVZ12].

We should care about another kind of attack based on Coppersmith's method. Indeed, [XSH+19] presented a heuristic attack that breaks Boneh *et al.*'s PRF (for a sufficiently large modulus p) if the number of outputs of the PRF is large enough (depending on δ). However, this polynomial-time attack is not practical and hides *galactic* constant factors. For instance, for $\delta = 2/3$, this attack requires 45 outputs of the PRF and uses a lattice of dimension 209899 in Coppersmith's method. We have checked that for 3 outputs, the attack requires $\delta > 5/6$, for 4 outputs $\delta > 7/10$, and for 5 outputs $\delta > 5/8$. We can observe that our sets of parameters are secure against these values. More generally, for a small number of outputs, the other Coppersmith's style attacks are ineffective if $1 - \delta \geq 1/2$. Indeed, [LSSW12] need δ to be at least $2/3$ and [BHH01] proposed a second attack (not described) which needs a large number of outputs to get a δ close to $1/2$.

To the best of our knowledge, the quantum security of Boneh *et al.*'s PRF has not been analyzed yet. Our signature protocol is thus a post-quantum candidate and requires further analysis of its security by quantum algorithm specialists.

4.6 Commitments with Efficient Zero-Knowledge Arguments from Subset Sum Problems

4.6.1 Contributions

Commitment scheme. In a celebrated paper, Impagliazzo and Naor [IN96] presented in particular a pseudo-random generator and an elegant bit commitment scheme. We extend the latter to a simple string commitment scheme and provide efficient zero-knowledge proofs for any relation amongst committed values using the recent zero-knowledge proof system proposed by Feneuil, Maire, Rivain, and Vergnaud and based on the *MPC-in-the-head* paradigm. We first present a modified version of the bit-commitment based on the subset sum problem proposed in [IN96]. This new scheme enables commitments to bit-strings and is related to the one from [IN96] in a similar manner to how the well-known Pedersen commitment scheme [Ped92] is related to preliminary discrete-logarithm based bit-commitments from [BCC88, BKK90].

The design principle is simple but seems to have been overlooked for more than 30 years (even if similar ideas have been used in lattice-based cryptography). For a security level $\lambda \in \mathbb{N}$ (*i.e.* against an adversary making 2^λ bit-operations using a $2^{\lambda/2}$-bits memory), it enables to commit to bit-strings of length $\ell \leq 2\lambda$ using a 2λ-bits modulus q and $(\ell + 2\lambda)$ integers smaller than q. The setup thus requires $O(\lambda^2)$ random or pseudo-random bits that can be generated easily using a so-called *extendable-output function* (XOF). A commitment is a sum of a (randomized) subset of these integers modulo q; therefore, it is of optimal bit-length 2λ and can be computed in $O(\lambda^2)$ binary operations. The *hiding* property (*i.e.* one cannot learn anything about the committed message from the commitment) relies on the hardness of the subset-sum problem, while its *binding* property (*i.e.* one cannot open a commitment to two different messages) relies on the hardness of the related *weighted knapsack problem*. With the proposed parameters, both problems are believed to be resistant to a quantum adversary that makes at most $2^{\lambda/2}$ qubits operations.

Zero-Knowledge Protocols. The zero-knowledge arguments for the subset sum problem presented in Section 4.3 of this chapter readily offer an efficient way to prove knowledge of a committed bit-string (with the commitment scheme introduced in this section) without revealing anything about it.

We extend this work to prove that a committed triple $(b_1, b_2, b_3) \in \{0,1\}^3$ satisfy a Boolean relation (*e.g.* $b_1 \wedge b_2 = b_3$ or $b_1 \oplus b_2 = b_3$) without revealing any additional information about them. The bits can be in arbitrary positions in the same or in different commitments and the proof of the Boolean relation does not add any overhead compared to the basic opening proof. This flexibility allows proving that committed bits $m_0, m_1, \ldots, m_\ell$ satisfy $m_0 = C(m_1, \ldots, m_\ell)$ for any Boolean circuit C with good communication and computational complexity. Indeed, by packing the commitments of bits on the circuit wires, we obtain a protocol with communication complexity $\tilde{O}(|C|\lambda + \lambda^2)$ where $|C|$ denotes the number of AND/XOR gates of C. This has to be compared with the code-based protocol due to Jain, Krenn, Pietrzak, and Tentes [JKPT12]. They provide a commitment scheme with zero-knowledge proofs from the LPN-assumption (or hardness of decoding a random linear code). This scheme has $\tilde{O}(|C|\lambda^2)$ communication complexity and allows only proving Boolean relations bit-wise on binary strings (which may result in a large overhead depending on the circuit considered). There also exist lattice-based constructions of commitment schemes with zero-knowledge proofs [BKLP15, BDL+18, ALS20] but the messages committed are small integers. They can be used to prove the satisfiability of arithmetic circuits but proving the satisfiability of a Boolean circuit with these schemes results in an important overhead in communication and computation.

4.6.2 Subset sum problems

We define hereafter two variants of the subset sum problem on which the security of our commitment scheme relies. The first one is the standard subset sum problem previously introduced, while the second one is a slightly stronger assumption that has already been used in cryptography (see, *e.g.* [BM97, SPW06]).

Definition 15. *Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0,1]$. Let $\ell, m : \mathbb{N} \to \mathbb{N}$ and modulus be an algorithm which given $\lambda \in \mathbb{N}$ outputs an integer q of bit-length $m(\lambda)$. We consider the two following assumptions:*

- (t, ϵ)-(decision) subset-sum assumption for $(\ell, m, \mathsf{modulus})$*: for every algorithm $\mathcal{A}$, we have for all $\lambda \in \mathbb{N}$:*

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{l} q \xleftarrow{\$} \mathit{modulus}(1^\lambda), \boldsymbol{\gamma} \xleftarrow{\$} [0, q-1]^{\ell(\lambda)}, \boldsymbol{x} \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}, \\ t_0 = \langle \boldsymbol{\gamma}, \boldsymbol{x} \rangle \bmod q, t_1 \xleftarrow{\$} [0, q-1], b \xleftarrow{\$} \{0,1\}, \\ b' \xleftarrow{\$} \mathcal{A}(1^\lambda, q, \boldsymbol{\gamma}, t_b) \end{array} \right] \leq \frac{1}{2} + \epsilon(\lambda)$$

  when $\mathcal{A}$ runs in time at most $t(\lambda)$ in this probabilistic computational game.

- (t, ϵ)-weighted knapsack assumption for $(\ell, m, \mathsf{modulus})$*: for every algorithm $\mathcal{A}$, we have for all $\lambda \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{c} \langle \boldsymbol{\gamma}, \boldsymbol{y} \rangle = 0 \bmod q \\ \boldsymbol{y} \neq \mathbf{0} \in \{-1, 0, 1\}^{\ell(\lambda)} \end{array} \;\middle|\; \begin{array}{l} q \xleftarrow{\$} \mathit{modulus}(1^\lambda), \boldsymbol{\gamma} \xleftarrow{\$} [0, q-1]^{\ell(\lambda)}, \\ \boldsymbol{y} \xleftarrow{\$} \mathcal{A}(1^\lambda, q, \boldsymbol{\gamma}) \end{array} \right] \leq \epsilon(\lambda)$$

  when $\mathcal{A}$ runs in time at most $t(\lambda)$ in this probabilistic computational game.

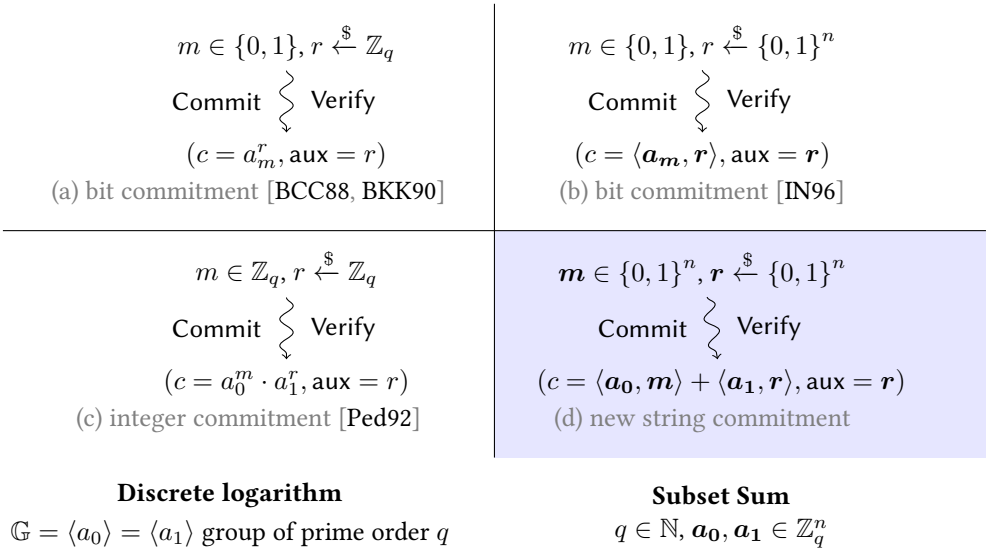


Fig. 4.2: Illustration of the Similarities between Commitment Schemes

The search version of the subset sum assumption is polynomial-time equivalent to the decision version stated above. The hardness of these problems depends greatly on the *density* defined as $d(\lambda) = \ell(\lambda)/m(\lambda)$. If the density is too small (e.g. $d(\lambda) < 1/\ell(\lambda)$) or too large (e.g. $d(\lambda) > \ell(\lambda)$) then both problems can be solved in polynomial time (see e.g. [CJL+92] and references therein). Coster, Joux, LaMacchia, Odlyzko, Schnorr, and Stern [CJL+92] proved that the subset sum problem can be solved in polynomial-time with a single call to an oracle that can find the shortest vector in a special lattice of dimension $\ell(\lambda)+1$ if $d(\lambda) < 0.9408$ and Li and Ma proved a similar result for the weighted knapsack problem if $d(\lambda) < 0.488$. It is worth mentioning that these results do not break the assumptions in polynomial time since the best algorithm for finding the shortest vector in these lattices has computational complexity $2^{\Theta(\ell(\lambda))}$ (and cryptographic protocols relying on these problems with much smaller densities have been proposed, e.g. [LPS10]).

In our construction, we will consider instances of these problems with density $d(\lambda) \simeq 1$ (i.e. $q \simeq 2^{\ell(\lambda)}$) for the subset sum problem since they are arguably the hardest ones [IN96]. This will result in instances for the weighted knapsack problem with density $d(\lambda) > 1$ and for conservative security, we will restrict ourselves to $d(\lambda) \leq 2$. In this case, lattice-based algorithms do not work and the best-known algorithms use very clever time-memory tradeoffs with the best algorithm due to Bonnetain, Bricout, Schrottenloher, and Shen [BBSS20] having time and memory complexities $\tilde{O}(2^{0.283\ell(\lambda)})$. These algorithms neglect the cost to access an exponential memory but even with this optimistic assumption, for $\ell(\lambda) = 256$, all known algorithms require at least a time complexity lower-bounded by 2^{128} operations or a memory of size at least 2^{64} bits. There also exists a vast literature on quantum algorithms for solving these problems (see [BBSS20] and references therein) and for $\ell(\lambda) = 256$, the best quantum algorithm requires about 2^{64} quantum operations and quantum memory.

4.6.3 String commitments from subset sum problems

We present our modified version of the bit-commitment based on the subset sum problem proposed in [IN96]. This new scheme enables commitments to bit-strings.

In [BCC88], Brassard, Chaum, and Crépeau introduced the notion of *blob*, which is very similar to bit commitment, and presented an elegant construction based on the discrete-logarithm problem in groups of known prime order q (see also [BKK90]). The commitment of a single bit consists of a group element (see Figure 4.2 (a) for an equivalent form of their commitment). Shortly afterward, Pedersen [Ped92] introduced his celebrated commitment scheme that enables committing to an integer in $\mathbb{Z}_q$ with a single group element (see Figure 4.2 (c)). Impagliazzo and Naor [IN96] proposed a bit-commitment whose hiding and binding security properties rely on the subset sum problem. It has many similarities with the discrete-logarithm-based blob from [BCC88, BKK90] (see Figure 4.2 (b)).

To build our string commitment scheme, we push this analogy and propose a variant of Pedersen's protocol based on the subset sum (see Figure 4.2 (d)). The design principle is simple and maybe folklore but does not seem to have been published in this form (even if similar ideas have been used in lattice-based cryptography).

4.6.4 Formal description and security analysis

Let $\ell, n, m : \mathbb{N} \to \mathbb{N}$ and let modulus be an algorithm which given $\lambda \in \mathbb{N}$ outputs an integer q of bit-length $m(\lambda)$. Typically, modulus outputs a random $m(\lambda)$-bit prime number or the unique integer $q = 2^{m(\lambda)-1}$. The function ℓ defines the message length while the function n defines the randomness length.

- $\mathsf{Setup}(1^\lambda) \to pp$. On input λ, the algorithm generates a modulus q by running $\mathsf{modulus}(1^\lambda)$ and picks uniformly at random $\boldsymbol{w} \in \mathbb{Z}_q^{\ell(\lambda)}$ and $\boldsymbol{s} \in \mathbb{Z}_q^{n(\lambda)}$. It outputs the public parameters $pp = (q, \boldsymbol{w}, \boldsymbol{s})$ and the message space is $\mathcal{M} = \{0,1\}^{\ell(\lambda)}$.
- $\mathsf{Com}(pp, m) \to (c, aux)$. On input pp and $\boldsymbol{m} \in \mathcal{M}$, the commit algorithm picks $aux = \boldsymbol{r} \in \{0,1\}^{n(\lambda)}$ uniformly at random, computes $c = \langle \boldsymbol{w}, \boldsymbol{m} \rangle + \langle \boldsymbol{s}, \boldsymbol{r} \rangle \bmod q$ and outputs (c, aux).
- $\mathsf{Ver}(pp, \boldsymbol{m}, c, aux) \to b \in \{0,1\}$. On input $pp = (q, \boldsymbol{w}, \boldsymbol{s})$, $\boldsymbol{m} \in \mathcal{M}$ and (c, aux), the verifier outputs 1 if $c = \langle \boldsymbol{w}, \boldsymbol{m} \rangle + \langle \boldsymbol{s}, \boldsymbol{r} \rangle \bmod q$ where $\boldsymbol{r} = aux \in \{0,1\}^{n(\lambda)}$, and 0 otherwise.

We prove that our commitment scheme is hiding and binding assuming the hardness of the subset sum and the weighted knapsack problems (respectively) for different lengths in the subset sum problems.

Theorem 10. *Let $\ell, n, m : \mathbb{N} \to \mathbb{N}$ and let modulus be an algorithm which given $\lambda \in \mathbb{N}$ outputs an integer of bit-length $m(\lambda)$. This commitment scheme above is:*

1. (t, ϵ)-computationally hiding *if the $(t + O(\ell(\lambda)m(\lambda)), \epsilon)$-decision subset-sum assumption holds for $(\ell, m,$ modulus$)$;*
2. (t, ϵ)-computationally binding *if the $(t + O(\ell(\lambda) + n(\lambda)), \epsilon)$-weighted knapsack assumption holds for $(\ell + n, m,$ modulus$)$.*

Proof. Both security reductions are simple, where adversaries against the commitment scheme properties are as in Definition 6.

1. Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be a (t, ϵ)-adversary (a two-phase algorithm) against the hiding property of the commitment scheme. We build a $(t + O(\ell(\lambda)m(\lambda)), \epsilon)$-adversary B breaking the decision subset sum assumption as follows. The algorithm B is given as inputs $(q, \boldsymbol{\gamma}, \alpha)$ where $\boldsymbol{\gamma} \in \mathbb{Z}_q^{n(\lambda)}$. The algorithm B picks uniformly at random $\boldsymbol{w} \in \mathbb{Z}_q^{\ell(\lambda)}$, and runs A_1 on input $pp = (q, \boldsymbol{w}, \boldsymbol{\gamma})$. When A_1 outputs two messages $\boldsymbol{m_0}, \boldsymbol{m_1} \in \{0,1\}^{\ell(\lambda)}$ and some state information s, the algorithm B picks uniformly at random a bit $b \in \{0,1\}$ and runs A_2 on $c = \langle \boldsymbol{w}, \boldsymbol{m_b} \rangle + \alpha \bmod q$ and s. Eventually, when A_2 outputs some bit b', B outputs 0 if $b' = b$ and 1 otherwise. A routine argument shows that the advantage of B for the decision subset sum problem is identical to the one of A for breaking the hiding property.
2. Let A be a (t, ϵ)-adversary against the binding property of the commitment scheme. We build a $(t + O(\ell(\lambda) + n(\lambda)), \epsilon)$-adversary B breaking the weighted knapsack assumption as follows. The algorithm B is given as inputs $(q, \boldsymbol{\gamma})$ where $\boldsymbol{\gamma} \in \mathbb{Z}_q^{\ell(\lambda)+n(\lambda)}$. It sets $\boldsymbol{w} = (\gamma_1, \ldots, \gamma_{\ell(\lambda)}) \in \mathbb{Z}_q^{\ell(\lambda)}$ and $\boldsymbol{s} = (\gamma_{\ell(\lambda)+1}, \ldots, \gamma_{\ell(\lambda)+n(\lambda)}) \in \mathbb{Z}_q^{n(\lambda)}$ and runs A on input $pp = (q, \boldsymbol{w}, \boldsymbol{s})$. When A outputs $(\boldsymbol{m_1}, \boldsymbol{m_2}, aux_1, aux_2, c)$, we have $\mathsf{Ver}(pp, \boldsymbol{m_1}, c, aux_1) = \mathsf{Ver}(pp, \boldsymbol{m_2}, c, aux_2) = 1$ and $\boldsymbol{m_1} \neq \boldsymbol{m_2}$ with probability $\epsilon(\lambda)$. Then, since $(\boldsymbol{m_1}, \mathsf{aux}_1), (\boldsymbol{m_2}, \mathsf{aux}_2) \in \{0,1\}^{\ell(\lambda)+n(\lambda)}$ and $\boldsymbol{m_1} \neq \boldsymbol{m_2}$, if B outputs the vector $\boldsymbol{y} = (\boldsymbol{m_1}, \mathsf{aux}_1) - (\boldsymbol{m_2}, \mathsf{aux}_2)$ (where the substraction is done coordinate-wise), it belongs to $\{-1, 0, 1\}^{\ell(\lambda)+n(\lambda)}$, is non-zero and satisfies $\langle \boldsymbol{\gamma}, \boldsymbol{y} \rangle = 0 \bmod q$ (and is thus a solution to the weighted knapsack problem $(q, \boldsymbol{\gamma})$).

□

Therefore, the hiding property relies on the hardness of the subset sum problem with density $n(\lambda)/m(\lambda)$ while its binding property on the hardness of the weighted knapsack problem with density $(\ell(\lambda) + n(\lambda))/m(\lambda)$. In the following, to simplify the protocols, we consider the case where $n(\lambda) = m(\lambda)$ (*i.e.* density 1 subset sum) and $\ell(\lambda) = n(\lambda)$ (*i.e.* density 2 weighted knapsack). To lighten the notations, we henceforth denote $n = n(\lambda) = \ell(\lambda)$.

4.6.5 Zero-knowledge arguments of opening

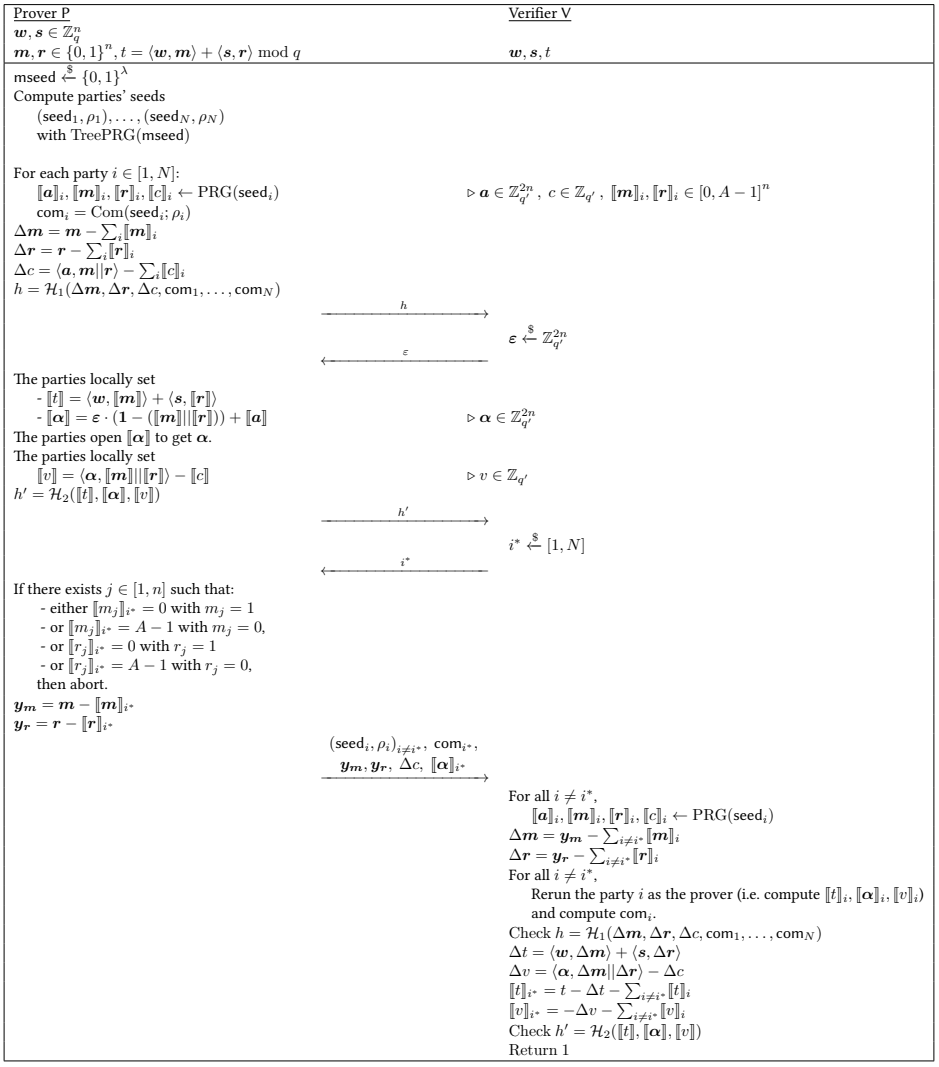
We present in Protocol 7 a zero-knowledge argument of knowledge of opening for our string commitment as an immediate application of the protocol proposed in Section 4.3 for the subset sum problem, hence based on our MPCitH *with rejection* framework. We refer to Section 4.3 for further details and a precise security analysis.

Concretely, let us consider the binary relation

$$\mathcal{R} = \{((q, \boldsymbol{w}, \boldsymbol{s}, t); (\boldsymbol{m}, \boldsymbol{r})) \mid \langle \boldsymbol{w}, \boldsymbol{m} \rangle + \langle \boldsymbol{s}, \boldsymbol{r} \rangle = t \bmod q\} \tag{4.19}$$

where $q \in \mathbb{N}$, $\boldsymbol{w}, \boldsymbol{s} \in \mathbb{Z}_q^n$, $t \in \mathbb{Z}_q$, and $\boldsymbol{m}, \boldsymbol{r} \in \{0,1\}^n$. Both the prover P and the verifier V know $(q, \boldsymbol{w}, \boldsymbol{s}, t)$ and P holds $(\boldsymbol{m}, \boldsymbol{r})$. P wants to convince V that they hold a valid message $\boldsymbol{m}$ and associated randomness $\boldsymbol{r}$ for the commitment t. The relation $\mathcal{R}$ defines an $\mathcal{NP}$ language and the membership of $(q, \boldsymbol{w}, \boldsymbol{s}, t)$ can be proved thanks to the witnesses $(\boldsymbol{m}, \boldsymbol{r})$ by verifying the relations (1) $\langle \boldsymbol{w}, \boldsymbol{m} \rangle + \langle \boldsymbol{s}, \boldsymbol{r} \rangle = t \bmod q$ and (2) $\boldsymbol{m}, \boldsymbol{r} \in \{0,1\}^n$.

Following the sharing on the integers technique developed at the beginning of this chapter, we choose $A \leq q$ for defining the interval on which the shares are going to be randomly picked, and q' the next prime after A. P starts by sharing on the integers the witnesses $(\boldsymbol{m}, \boldsymbol{r})$ among the N parties, and then emulates an $(N-1)$-private MPC protocol



Protocol 7: Zero-knowledge argument for string-commitment using batch product verification to prove binarity.

with N parties for verifying the relations (1) and (2). The verification of (1) is linear modulo q' and is therefore free in terms of communication with V but proving (2) requires performing some multiplications in the MPC protocol (using the simple fact that $x \in \{0,1\}$ if and only if $x(1-x) = 0 \mod q'$). The verification of these multiplications can be realized following [BN20a] (see Chapter 3). This implies a communication cost of $2\log(q')$ bits to prove one multiplication. Using a previously introduced batched version of this verification protocol [KZ22], one gets a communication cost of $(n+1)\log_2 q'$ for n multiplications, and the soundness error of this protocol follows from the Schwartz-Zippel Lemma [Zip79, Sch80]. It yields to a communication cost for the Protocol 7 with τ repetitions and λ bits of security of

$$4\lambda + \tau\left[2n(\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda\log_2 N + 2\lambda\right] \text{ bits.}$$

As mentioned previously, since the rejection rate after τ repetitions (*i.e.* that any of the τ repetition aborts) is given by $1-(1-1/A)^{2n\tau} \simeq 2n\tau/A$ where the approximation is tight when A is sufficiently large. Thus by taking $A = \Theta(n\tau)$, we get a (small) constant rejection probability.

Theorem 11 (Security proofs of Protocol 7)**.** *Let the* PRG *used in Protocol 7 be (t, ε_{PRG})-secure and the commitment scheme* Com *be (t, ε_{Com})-hiding. Then Protocol 7 is a zero-knowledge argument of knowledge for the relation $\mathcal{R}$ 4.19 with $(1-1/A)^{2n}$-completeness, $(1/q' + 1/N - 1/Nq')$-soundness, and $(t, \varepsilon_{PRG} + \varepsilon_{Com})$-zero-knowledge.*

Proof. Proofs are identical to those of Theorems 4, 5, and 6. □

Remark 6. *In Subsection 4.2.4, we proposed a second approach to prove (2) using "*cut-and-choose*". It can be used to prove the knowledge of a commitment opening, but it is not well adapted for the next applications, notably proving Boolean relations of committed values as our next application. Moreover, it is worth mentioning that our argument of knowledge of opening can be easily generalized to an argument of partial opening by revealing bits of the committed message, modifying the value of the commitment accordingly and proving the knowledge of the remaining hidden bits. This enables to provide a* range proof *of the committed message at no additional cost.*

4.6.6 Zero-knowledge arguments for Boolean relations

Coordinate-wise AND gates. Let us consider the case when three n-bits binary vectors $\boldsymbol{m^1}, \boldsymbol{m^2}, \boldsymbol{m^3}$ are committed and P wants to prove in zero-knowledge that $\boldsymbol{m^1} \circ \boldsymbol{m^2} = \boldsymbol{m^3}$. Note that proving $\boldsymbol{m^1}, \boldsymbol{m^2}$ are binary and $\boldsymbol{m^1} \circ \boldsymbol{m^2} = \boldsymbol{m^3} \mod q'$ implies that $\boldsymbol{m^3}$ is binary and $\boldsymbol{m^1} \circ \boldsymbol{m^2} = \boldsymbol{m^3}$. In addition, P has to prove that the three random vectors $\boldsymbol{r^1}, \boldsymbol{r^2}, \boldsymbol{r^3}$ used in the commitment are all binary (no relation is proved between them). Using this approach, P has to prove $6n$ multiplications and therefore the argument requires sending $6n+1$ elements of $\mathbb{Z}_{q'}$ via the batched version of the verification protocol [BN20a, KZ22] detailed in Chapter 3. Indeed, let us recall that for proving n coordinate-wise multiplications over $\mathbb{Z}_{q'}$ of the form $x \circ y = z$, where $x, y, z \in \mathbb{Z}_{q'}^n$ have been shared, we use the sacrifizing strategy. For this purpose, P shares a random $a \in \mathbb{Z}_{q'}^n$ and the value $c = \langle a, x\rangle$ to realize the batching proof. Hence, P has to send (1) $[\![\alpha]\!]_{i^*} = \epsilon[\![x]\!]_{i^*} + [\![a]\!]_{i^*} \in \mathbb{Z}_{q'}^n$ where ϵ is a challenge from V and (2) $\Delta c = \langle a, x\rangle - \sum_i [\![c]\!]_i$, yielding to $(n+1)\log_2 q'$ bits to communicate (on top of the sharing cost of x, y, z). In the following, we may deal with multiplications $x \circ y = z$ such that z is a linear combination (with public coefficients) between n-bits shared vectors, and the communication cost remains $(n+1)\log_2 q'$ bits.

Essentially, it is possible to batch some verification equations and reduce this number from $6n+1$ to $5n+1$. Indeed, checking $\boldsymbol{m^1} \circ \boldsymbol{m^2} = \boldsymbol{m^3} \mod q'$ and (for instance) the binarity of $\boldsymbol{m^2}$ is equivalent (with a false-positive error probability coming from the Schwartz-Zippel Lemma) to verify that

$$\lambda_1 \boldsymbol{m^2} \circ (\boldsymbol{1} - \boldsymbol{m^2}) + \lambda_2 \boldsymbol{m^1} \circ \boldsymbol{m^2} = \lambda_2 \boldsymbol{m^3} \mod q' \tag{4.20}$$

where $\lambda_1, \lambda_2 \in \mathbb{Z}_{q'}$ are random elements chosen by V. Then, the next batching equation aims to proof all these component-wise multiplications

$$(\boldsymbol{m^1}||\boldsymbol{r^1}||\boldsymbol{r^2}||\boldsymbol{r^3}||\boldsymbol{m^2}) \circ ((\boldsymbol{1} - (\boldsymbol{m^1}||\boldsymbol{r^1}||\boldsymbol{r^2}||\boldsymbol{r^3}))||(\lambda_1(\boldsymbol{1} - \boldsymbol{m^2}) + \lambda_2 \boldsymbol{m^1})) = (\boldsymbol{0}||\lambda_2 \boldsymbol{m^3}),$$

leading to the Protocol 8.

Arbitrary AND gates. Fundamentally, Protocol 8 is similar to the protocols from [JKPT12, BKLP15, BDL+18, ALS20] in the sense that it can only prove coordinate-wise multiplication relations. We generalize it to obtain a more flexible protocol able to prove relations of the form $m_i^1 \wedge m_j^2 = m_k^3$ for any coordinates $i, j, k \in [1, n]$ (where $\wedge$ denotes the AND gate).

Assume P has to prove the satisfiability of a set of AND gates whose inputs/output belong to a set of $L \in \mathbb{N}$ committed vectors $\{\boldsymbol{m^\ell}\}_{1 \le \ell \le L} \in \{0,1\}^n$. Given some committed vector $\boldsymbol{m^\ell}$ (with $\ell \in [1, L]$), let us consider a set of $M \in \mathbb{N}$ AND gates such that

$$m_{x_k}^{\ell} \wedge m_{y_k}^{\ell_k} = m_{z_k}^{\ell'_k}$$

Prover P		Verifier V
$\boldsymbol{w}, \boldsymbol{s} \in \mathbb{Z}_q^n, \boldsymbol{m^k}, \boldsymbol{r^k} \in \{0,1\}^n$ for $1 \le k \le 3$		
$\boldsymbol{m^1} \circ \boldsymbol{m^2} = \boldsymbol{m^3}, t^k = \langle \boldsymbol{w}, \boldsymbol{m^k} \rangle + \langle \boldsymbol{s}, \boldsymbol{r^k} \rangle$		$\boldsymbol{w}, \boldsymbol{s}, t^k$
mseed $\xleftarrow{\$} \{0,1\}^\lambda$ Compute parties' seeds $(\mathsf{seed}_1, \rho_1), \ldots, (\mathsf{seed}_N, \rho_N)$ with $\mathrm{TreePRG}(\mathsf{mseed})$		
For each party $i \in [1, N]$: $[\![\boldsymbol{a}]\!]_i, [\![c]\!]_i, \{[\![\boldsymbol{m^k}]\!]_i, [\![\boldsymbol{r^k}]\!]_i\}_{1 \le k \le 3}$ $\leftarrow \mathrm{PRG}(\mathsf{seed}_i)$		$\triangleright\ \boldsymbol{a} \in \mathbb{Z}_{q'}^{5n},\ c \in \mathbb{Z}_{q'},\ [\![\boldsymbol{m^k}]\!]_i, [\![\boldsymbol{r^k}]\!]_i \in [0, A-1]^n$
$\mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$		
For $1 \le k \le 3$: $\Delta \boldsymbol{m^k} = \boldsymbol{m^k} - \sum_i [\![\boldsymbol{m^k}]\!]_i$ $\Delta \boldsymbol{r^k} = \boldsymbol{r^k} - \sum_i [\![\boldsymbol{r^k}]\!]_i$		
$\Delta c = -\langle \boldsymbol{a}, \boldsymbol{m^1} \| \boldsymbol{r^1} \| \boldsymbol{r^2} \| \boldsymbol{r^3} \| \boldsymbol{m^2} \rangle - \sum_i [\![c]\!]_i$		
$h = \mathcal{H}_1(\{\Delta \boldsymbol{m^k}, \Delta \boldsymbol{r^k}\}_{1 \le k \le 3}, \Delta c,$ $\mathsf{com}_1, \ldots, \mathsf{com}_N)$		
	$\xrightarrow{h}$	
		$\boldsymbol{\varepsilon} \xleftarrow{\$} \mathbb{Z}_{q'}^{5n}, \lambda_1, \lambda_2 \xleftarrow{\$} \mathbb{Z}_{q'}$
	$\xleftarrow{\varepsilon}$	
The parties locally set - $[\![t^k]\!] = \langle \boldsymbol{w}, [\![\boldsymbol{m^k}]\!] \rangle + \langle \boldsymbol{s}, [\![\boldsymbol{r^k}]\!] \rangle$ for $1 \le k \le 3$ - $[\![\boldsymbol{\alpha}]\!] = \boldsymbol{\varepsilon} \circ ((\mathbf{1} - [\![\boldsymbol{m^1} \| \boldsymbol{r^1} \| \boldsymbol{r^2} \| \boldsymbol{r^3}]\!]) \|$ $(\lambda_1(\mathbf{1} - [\![\boldsymbol{m^2}]\!]) + \lambda_2 [\![\boldsymbol{m^1}]\!])) + [\![\boldsymbol{a}]\!]$		$\triangleright\ \boldsymbol{\alpha} \in \mathbb{Z}_{q'}^{5n}$ (computation in $\mathbb{Z}_{q'}$)
The parties open $[\![\boldsymbol{\alpha}]\!]$ to get $\boldsymbol{\alpha}$.		
The parties locally set $[\![v]\!] = \langle \boldsymbol{\alpha}, [\![\boldsymbol{m^1} \| \boldsymbol{r^1} \| \boldsymbol{r^2} \| \boldsymbol{r^3} \| \boldsymbol{m^2}]\!] \rangle - [\![c]\!] -$ $\langle \boldsymbol{\varepsilon}, \mathbf{0} \| \lambda_2 [\![\boldsymbol{m^3}]\!] \rangle$		$\triangleright\ v \in \mathbb{Z}_{q'}$ (computation in $\mathbb{Z}_{q'}$)
$h' = \mathcal{H}_2(\{[\![t^k]\!]\}_{1 \le k \le 3}, [\![\boldsymbol{\alpha}]\!], [\![v]\!])$		
	$\xrightarrow{h'}$	
		$i^* \xleftarrow{\$} [1, N]$
	$\xleftarrow{i^*}$	
If there exists $k \in [1,3]$ and $j \in [1,n]$ such that: - either $[\![m_j^k]\!]_{i^*} = 0$ with $m_j^k = 1$ - or $[\![m_j^k]\!]_{i^*} = A - 1$ with $m_j^k = 0$, - or $[\![r_j^k]\!]_{i^*} = 0$ with $r_j^k = 1$ - or $[\![r_j^k]\!]_{i^*} = A - 1$ with $r_j^k = 0$, then abort.		
$\boldsymbol{y}_{\boldsymbol{m^k}} = \boldsymbol{m^k} - [\![\boldsymbol{m^k}]\!]_{i^*}$ and $\boldsymbol{y}_{\boldsymbol{r^k}} = \boldsymbol{r^k} - [\![\boldsymbol{r^k}]\!]_{i^*}$ for $k \in [1,3]$		
	$(\mathsf{seed}_i, \rho_i)_{i \ne i^*},\ \mathsf{com}_{i^*},$ $\{\boldsymbol{y}_{\boldsymbol{m^k}}, \boldsymbol{y}_{\boldsymbol{r^k}}\}_{1 \le k \le 3},\ \Delta c,\ [\![\boldsymbol{\alpha}]\!]_{i^*}$ $\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad}$	
		For all $i \ne i^*$, $[\![\boldsymbol{a}]\!]_i, [\![c]\!]_i, \{[\![\boldsymbol{m^k}]\!]_i, [\![\boldsymbol{r^k}]\!]_i\}_{1 \le k \le 3}$ $\leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		For all $i \ne i^*$, Rerun the party i as the prover and compute com_i.
		For $1 \le k \le 3$, $\Delta \boldsymbol{m^k} = \boldsymbol{y}_{\boldsymbol{m^k}} - \sum_{i \ne i^*} [\![\boldsymbol{m^k}]\!]_i$ $\Delta \boldsymbol{r^k} = \boldsymbol{y}_{\boldsymbol{r^k}} - \sum_{i \ne i^*} [\![\boldsymbol{r^k}]\!]_i$ $\Delta t^k = \langle \boldsymbol{w}, \Delta \boldsymbol{m^k} \rangle + \langle \boldsymbol{s}, \Delta \boldsymbol{r^k} \rangle$ $[\![t^k]\!]_{i^*} = t^k - \Delta t^k - \sum_{i \ne i^*} [\![t^k]\!]_i$
		$\Delta v = \langle \boldsymbol{\alpha}, \Delta \boldsymbol{m^1} \| \Delta \boldsymbol{r^1} \| \Delta \boldsymbol{r^2} \| \Delta \boldsymbol{r^3} \| \Delta \boldsymbol{m^2} \rangle$ $-\Delta c - \langle \boldsymbol{\varepsilon}, \mathbf{0} \| \lambda_2 \Delta \boldsymbol{m^3} \rangle$
		$[\![v]\!]_{i^*} = -\Delta v - \sum_{i \ne i^*} [\![v]\!]_i$
		Check $h = \mathcal{H}_1(\{\Delta \boldsymbol{m^k}, \Delta \boldsymbol{r^k}\}_{1 \le k \le 3}, \Delta c,$ $\mathsf{com}_1, \ldots, \mathsf{com}_N)$
		Check $h' = \mathcal{H}_2(\{[\![t^k]\!]\}_{1 \le k \le 3}, [\![\boldsymbol{\alpha}]\!], [\![v]\!])$
		Return 1

Protocol 8: Zero-knowledge argument for proving coordinate-wise AND relations between committed vectors.

for $k \in [1, M], \ell_k, \ell'_k \in [1, L], x_k, y_k, z_k \in [1, n]$. Moreover, as seen previously to check that $\boldsymbol{m}^{\boldsymbol{\ell}}$ is binary, V can verify $\boldsymbol{m}^{\boldsymbol{\ell}} \circ (\mathbf{1} - \boldsymbol{m}^{\boldsymbol{\ell}}) = 0 \mod q'$. Then we can batch these proofs as

$$\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} \circ (\mathbf{1} - \boldsymbol{m}^{\boldsymbol{\ell}}) + \sum_{k=1}^{M} \lambda_k m_{x_k}^{\ell} m_{y_k}^{\ell_k} \boldsymbol{e}_{\boldsymbol{x_k}} = \sum_{k=1}^{M} \lambda_k m_{z_k}^{\ell'_k} \boldsymbol{e}_{\boldsymbol{x_k}} \mod q'$$

i.e.

$$\boldsymbol{m}^{\boldsymbol{\ell}} \circ [-\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} + \sum_{k=1}^{M} \lambda_k m_{y_k}^{\ell_k} \boldsymbol{e}_{\boldsymbol{x_k}}] = -\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} + \sum_{k=1}^{M} \lambda_k m_{z_k}^{\ell'_k} \boldsymbol{e}_{\boldsymbol{x_k}} \mod q' \tag{4.21}$$

where $\boldsymbol{e_i}$ is the i-th vector of the canonical basis of $\mathbb{Z}_{q'}^n$ and $\{\lambda_k\}_{0 \leq k \leq M} \in \mathbb{Z}_{q'}$ are random elements chosen by V.

To summary, P can batch the proofs for the satisfiability of a set of gates that involve as input a component from the same committed vector. This batching can include the binary verification of this specific vector.

Importantly, the number of equations does not depend anymore on the number of gates to prove. We obtain the generalized Protocol 17 as a direct extension of Protocol 8 (essentially the batching part is slightly different) which can be found in Appendix D.

Security analysis. The following theorems state the completeness, soundness, and zero-knowledge of Protocols 8 and 17. The proofs are similar to those of Theorems 4,5, and 6. For the sake of simplicity, we consider the worst case by assuming that all the coordinates of the committed vectors are involved in the gates' evaluation.

Theorem 12 (Protocol 8)**.** *Let the* PRG *used in Protocol 8 be* (t, ε_{PRG})*-secure and the commitment scheme* Com *be* (t, ε_{Com})*-hiding. Then, Protocol 8 is a (honest-verifier) zero-knowledge argument of knowledge for the coordinate-wise AND gates satisfiability (from* n*-length vectors) with* $(1 - 1/A)^{6n}$*-completeness,* $(1/N + (1 - 1/N)1/q')$*- soundness and* $(t, \varepsilon_{PRG} + \varepsilon_{Com})$*-zero-knowledge.*

Theorem 13 (Protocol 17)**.** *Let the* PRG *used in Protocol 17 be* (t, ε_{PRG})*-secure and the commitment scheme* Com *be* (t, ε_{Com})*-hiding. Then, Protocol 17 is a (honest-verifier) zero-knowledge argument of knowledge for arbitrary AND gates satisfiability (from* L *committed vectors of length* n*) with* $(1 - 1/A)^{2Ln}$*-completeness,* $(1/N + (1 - 1/N)1/q')$*-soundness and* $(t, \varepsilon_{PRG} + \varepsilon_{Com})$*-zero-knowledge.*

Note that Protocol 8 and 17 (in Appendix D) have the same soundness as Protocol 7. This follows from the Schwartz-Zippel Lemma, since the false-positive error probability of the MPC protocols after batching techniques still corresponds to the probability that some multinomial of degree one vanishes at some random field element. For example, the linear combination in Equation (4.21) equals 0 with probability at most $1/q'$ when evaluating the multinomial $P(X_0, X_1, \ldots, X_M) = -X_0 \boldsymbol{m}^{\boldsymbol{\ell}} + \sum_{k=1}^{M} X_k m_{y_k}^{\ell_k} \boldsymbol{e}_{\boldsymbol{x_k}}$ at random points $(\lambda_0, \lambda_1, \ldots, \lambda_M)$ from the Schwartz Zippel lemma. The rejection rates are bigger than in Protocol 7 since respectively $6n$ and $2Ln$ sharing on the integers are emulated. This requires a slight increase of A (as shown in Table 4.1).

The communication cost (in bits) of Protocol 8 with τ repetitions is at most:

$$4\lambda + \tau \left[n(6 \log_2(A - 1) + 5 \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda \right],$$

and for the generalized Protocol 17:

$$4\lambda + \tau \left[2Ln(\log_2(A - 1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda \right].$$

We notice that it does not depend on the number M of AND gates to prove.

4.6.7 XOR gates

Coordinate-wise XOR gates. We first consider three n-bits committed vectors $\boldsymbol{m^1}, \boldsymbol{m^2}, \boldsymbol{m^3}$ such that P wants to prove $\boldsymbol{m^1} \oplus \boldsymbol{m^2} = \boldsymbol{m^3}$ (where $\oplus$ denotes the XOR gate).

Let f be the polynomial $f(x) = 2x - x^2$ over $\mathbb{Z}_{q'}$ with $q' \geq 3$ a prime number. One can easily check that if $\boldsymbol{m^1}$ and $\boldsymbol{m^2}$ are binary vectors, then $f(\boldsymbol{m^1} + \boldsymbol{m^2}) \bmod q' = \boldsymbol{m^1} \oplus \boldsymbol{m^2} \in \{0, 1\}$. Thus, proving that $f(\boldsymbol{m^1} + \boldsymbol{m^2}) = \boldsymbol{m^3} \mod q'$ in conjunction with the argument of knowledge for opening of the corresponding commitments, implies $\boldsymbol{m^1} \oplus \boldsymbol{m^2} = \boldsymbol{m^3}$.

With the same techniques as in Protocol 8 for the $\wedge$ gate, we obtain Protocol 16 for bit-wise $\oplus$ gates which can be found in Appendix D.

Arbitrary XOR Gates. Again, the previous protocol is not enough flexible and can not be used to prove relations such as $m_i^1 \oplus m_j^2 = m_k^3$ for arbitrary $i, j, k \in [1, n]$, but we outline how to generalize it.

Assume P has to prove the satisfiability of a set of XOR gates whose inputs/output belong to a set of $L \in \mathbb{N}$ committed vectors $\{\boldsymbol{m}^{\boldsymbol{\ell}}\}_{1 \leq \ell \leq L} \in \{0,1\}^n$. Given some committed vector $\boldsymbol{m}^{\boldsymbol{\ell}}$ (with $\ell \in [1, L]$), let us consider a set of $M \in \mathbb{N}$ XOR gates such that

$$m_{x_k}^{\ell} \oplus m_{y_k}^{\ell_k} = m_{z_k}^{\ell'_k}$$

for $k \in [1, M], \ell_k, \ell'_k \in [1, L], x_k, y_k, z_k \in [1, n]$. If we assume that the binarity of each committed vector is checked during the protocol, then

$$f(m_{x_k}^{\ell} + m_{y_k}^{\ell_k}) = 2(m_{x_k}^{\ell} + m_{y_k}^{\ell_k}) - (m_{x_k}^{\ell} + m_{y_k}^{\ell_k})^2 = m_{x_k}^{\ell} \oplus m_{y_k}^{\ell_k} = m_{z_k}^{\ell'_k} \mod q'.$$

Moreover, we can prove that $\boldsymbol{m}^{\boldsymbol{\ell}}$ is binary via $\boldsymbol{m}^{\boldsymbol{\ell}} \circ (\mathbf{1} - \boldsymbol{m}^{\boldsymbol{\ell}}) = 0 \mod q'$, and then batch all these equations as

$$\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} \circ (\mathbf{1} - \boldsymbol{m}^{\boldsymbol{\ell}}) + \sum_{k=1}^{K} \lambda_k \left(2(m_{x_k}^{\ell} + m_{y_k}^{\ell_k}) - (m_{x_k}^{\ell} + m_{y_k}^{\ell_k})^2\right) \boldsymbol{e}_{\boldsymbol{x_k}} = \sum_{k=1}^{K} \lambda_k m_{z_k}^{\ell'_k} \boldsymbol{e}_{\boldsymbol{x_k}} \mod q'.$$

If the binarity of $\boldsymbol{m}^{\boldsymbol{\ell_k}}$ and $\boldsymbol{m}^{\boldsymbol{\ell}}$ is proven elsewhere, V is convinced that $m_{y_k}^{\ell_k} m_{y_k}^{\ell_k} = m_{y_k}^{\ell_k} \mod q'$ and $m_{x_k}^{\ell} m_{x_k}^{\ell} = m_{x_k}^{\ell} \mod q'$. Hence, the batching equation becomes

$$\boldsymbol{m}^{\boldsymbol{\ell}} \circ [-\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} - 2\sum_{k=1}^{M} \lambda_k m_{y_k}^{\ell_k} \boldsymbol{e}_{\boldsymbol{x_k}}] = -\lambda_0 \boldsymbol{m}^{\boldsymbol{\ell}} + \sum_{k=1}^{M} \lambda_k \left(m_{z_k}^{\ell'_k} - m_{y_k}^{\ell_k} - m_{x_k}^{\ell}\right) \boldsymbol{e}_{\boldsymbol{x_k}} \mod q', \quad (4.22)$$

where $\boldsymbol{e_i}$ is the i-th vector of the canonical basis of $\mathbb{Z}_{q'}^n$ and $\{\lambda_k\}_{0 \leq k \leq M} \in \mathbb{Z}_{q'}$ are random elements chosen by V.

Security analysis. The theorems stating the completeness, soundness and zero-knowledge of the protocol for the bit-wise XOR (Protocol 16) and for its generalization can be directly derived drawing inspiration from Theorems 12 and 13 (respectively).

The communication complexity (in bits) of the protocol for arbitrary XOR gates with τ repetitions is:

$$4\lambda + \tau \left[2Ln(\log_2(A-1) + \log_2(q')) + \log_2(q') + \lambda \log_2 N + 2\lambda\right],$$

while the one for the bit-wise XOR is the subcase when $L = 3$. We notice that the proof size is the same as for Protocol 17 and is independent of M.

4.6.8 Instantiation and performances

We present some sets of parameters for an instantiation of our commitment scheme with $m(\lambda) = \ell(\lambda) = n(\lambda) = 256$ (*i.e.* with security based on density 1 for the subset-sum and density 2 for the weighted knapsack). We look at the performances of the protocols for component-wise AND and XOR gates. To decrease the rejection rate, we use a strategy introduced in Subsection 4.3.5 that consists in allowing P to abort in $0 \leq \eta < \tau$ out of the τ iterations. V accepts the proof if P can answer to $\tau - \eta$ challenges among the τ iterations. This relaxed proof has a significantly lower rejection rate (at the cost of a small increase of the soundness error).

Protocol	Parameters				Proof size	Rej. rate	Soundness err.
	τ	η	N	A			
Protocol 7 (Opening)	21	3	256	2^{13}	35.4 KB	0.035	133 bits
Protocol 7 (Opening)	19	2	256	2^{13}	33.3 KB	0.104	128 bits
Protocol 8 (AND)	21	3	256	2^{15}	98.9 KB	0.014	133 bits
Protocol 8 (AND)	19	2	256	2^{15}	93.4 KB	0.054	128 bits
Protocol 16 (XOR)	21	3	256	2^{15}	107.4 KB	0.014	133 bits
Protocol 16 (XOR)	19	2	256	2^{15}	101.3 KB	0.054	128 bits

Tab. 4.5: Comparison of performances with $n = 256$ and $q \approx 2^{256}$.

4.6.9 Arguments for circuit satisfiability

Let C be a Boolean circuit with $|C|$ gates (AND or XOR) and T input bits. Let $\boldsymbol{m} \in \{0,1\}^T$ and $v_1, \ldots, v_{|C|} \in \{0,1\}$ be committed elements such that $\boldsymbol{m}$ is an input that satisfy C and the v's are the outputs of each gates of C when evaluated on $\boldsymbol{m}$, *i.e.*, $C(m_1, \ldots, m_T) = v_{|C|} = 1$. The prover $\mathcal{P}$ wants to prove in zero-knowledge that $\boldsymbol{m}$ indeed satisfies the

public circuit C. For this purpose, we use the commitment scheme introduced in Subsection 4.6.3 along with protocols for proving Boolean relations from Subsection 4.6.6. For simplicity, we assume without loss of generality that $T \leq n$. Since n bits can be committed via the same commitment (n is the size of the subset-sum instance), we need $|C|/n + 1$ string commitments. We introduce the following notation to simplify the batching equation: for $k \in [0, |C|/n]$,

$$\boldsymbol{v^0} = (\boldsymbol{m}||v_1|| \dots ||v_{n-T}), \dots, \boldsymbol{v}^{|\boldsymbol{C}|/\boldsymbol{n}} = (v_{|C|-T+1}|| \dots ||v_{|C|}||\boldsymbol{0}),$$

where these vectors are n-bits long. Following the batching from Equation (4.21) and Equation (4.22), we can set $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ as follows so that the circuit satisfiability verification consists in checking that $\boldsymbol{x} \circ \boldsymbol{y} = \boldsymbol{z}$:

$$\boldsymbol{y} = (\boldsymbol{v^0}|| \dots ||\boldsymbol{v}^{|\boldsymbol{C}|/\boldsymbol{n}}||\boldsymbol{r^0}|| \dots ||\boldsymbol{r}^{|\boldsymbol{C}|/\boldsymbol{n}}),$$

$$\begin{aligned}\boldsymbol{x} = \Bigg(-\lambda_0 \boldsymbol{v^0} + \sum_{i=1}^{n} \sum_{j=0}^{|C|/n} \sum_{k=1}^{n} \lambda_{v_k^j} \left(\delta_{0,i,j,k} v_k^j - 2\zeta_{0,i,j,k} v_k^j \right) \boldsymbol{e_i} || \dots \\ || - \lambda_{|C|/n} \boldsymbol{v}^{|\boldsymbol{C}|/\boldsymbol{n}} + \sum_{i=1}^{n} \sum_{j=0}^{|C|/n} \sum_{k=1}^{n} \lambda_{v_k^j} \left(\delta_{|C|/n,i,j,k} v_k^j - 2\zeta_{|C|/n,i,j,k} v_k^j \right) \boldsymbol{e_i} \\ ||\boldsymbol{1} - \boldsymbol{r^1}|| \dots ||\boldsymbol{1} - \boldsymbol{r}^{|\boldsymbol{C}|/\boldsymbol{n}} \Bigg)\end{aligned}$$

where $\boldsymbol{r^0}, \dots, \boldsymbol{r}^{|\boldsymbol{C}|/\boldsymbol{n}}$ is the randomness used in the commitments and the vector $\boldsymbol{z}$ can be computed as a linear combination of $\boldsymbol{v^0}, \dots, \boldsymbol{v}^{|\boldsymbol{C}|/\boldsymbol{n}}$. As above, $\boldsymbol{e_i}$ is the i-th vector of the canonical basis of $\mathbb{Z}_{q'}^n$, λ's are random public values chosen by the verifier V, and the binary elements ζ and δ depend on the circuit structure, *i.e.* $\delta_{\ell,i,j,k} = 1$ if and only if $v_i^\ell \wedge v_k^j = v_p^u$ for some $u \in [0, |C|/n]$ and $v \in [1, n]$ (and $\zeta_{\ell,i,j,k} = 1$ if and only if $v_i^\ell \oplus v_k^j = v_p^u$). Hence, V has to check $\boldsymbol{x} \circ \boldsymbol{y} = \boldsymbol{z}$ to be convinced of the binarity of the vectors, and of the satisfiability of the circuit. The full protocol is given as Protocol 18 in Appendix D.

Theorem 14 (Protocol 18)**.** *Let the* PRG *used in Protocol 18 be (t, ε_{PRG})-secure and the commitment scheme* Com*be (t, ε_{Com})-hiding. The protocol 18 is a zero-knowledge proof of knowledge for the relation $\mathcal{R}$ with $(1 - 1/A)^{2(|C|+n)}$-completeness, $(1/N + (1 - 1/N)1/q')$-soundness and $(t, \varepsilon_{PRG} + \varepsilon_{Com})$-zero-knowledge.*

The communication cost (in bits) of Protocol 18 with τ repetitions is:

$$4\lambda + \tau \left[2(|C| + n) \log(q') + 2|C| \log(A - 1) + \log(q') + \lambda \log N + 2\lambda \right].$$

With $n = 2\lambda$ and $A = \Theta((|C| + n)\tau)$ (for a small constant rejection probability), its asymptotic complexity is $\Theta\left(\frac{\lambda(|C|+\lambda)}{\log N} \log\left(\frac{\lambda(|C|+\lambda)}{\log N} \right) + \lambda^2 \right)$. With $N = \Theta(\lambda)$ to minimize, we get asymptotic complexity $\tilde{\Theta}(\lambda|C| + \lambda^2)$ to be compared with $\tilde{\Theta}(|C|\lambda^2)$ in [JKPT12] (which can only prove Boolean relations bit-wise on binary strings and may result in a large overhead depending on the circuit considered).

5. ZERO-KNOWLEDGE ARGUMENTS VIA SHARING CONVERSION

The goal of this chapter is to add another string to the MPCitH's bow by integrating secret sharing conversion, a technique that has already been used in general MPC [GPS12], or for protecting against auxiliary channel attacks [Gou01].

5.1 Related Works and Contributions

We present a new technique to expand the MPCitH toolbox further by allowing a prover to use simultaneously in the MPC protocol additive sharings and multiplicative sharings of its secret information. The former are used for linear relations, while the latter are used to prove efficiently multiplicative relations. To ensure consistency, we propose a simple technique to transform a multiplicative share into an additive share of the same value. Converting shares from one type of secret sharing scheme into another is ubiquitous in MPC [GPS12] and the idea has already been used in the MPCitH realm [DGH$^+$21] (but for different sharings). Our technique finds several applications in (post-quantum) zero-knowledge arguments and digital signature schemes.

Double Discrete Logarithm Problem (DDLP): A double discrete logarithm of an element $y \neq 1_{\mathbb{G}}$ in a cyclic group $\mathbb{G}$ of prime order q with respect to bases $g \in \mathbb{G}$ and $h \in \mathbb{F}_q^*$ (generators of $\mathbb{G}$ and $\mathbb{F}_q^*$ respectively) is an integer $x \in [0, q-1]$ such that $y = g^{h^x}$. Initially introduced by Stadler [Sta96] for verifiable secret-sharing, this computational problem has found applications in various cryptographic protocols, including group signatures [CS97], blind signatures [ASM10], e-cash systems [CG07], credential systems [CGM16], and verifiable randomness generation [BTV20]. Stadler proposed a zero-knowledge protocol, which has a computational and communication complexity of $\Omega(\log q)$ (in terms of group elements). However, in the recent work [BTV20], Blazy, Towa, and Vergnaud presented a new protocol that outputs arguments with only $O(\log \log q)$ group elements. It relies on the "*Bulletproofs*" technique proposed by Bünz, Bootle, Boneh, Poelstra, Wuille and Maxwell in 2018 [BBB$^+$18]. This reduced communication complexity comes at a security price since the security analysis should rely on stronger idealized assumptions [GOP$^+$22] or achieve only non-meaningful concrete security [DG23]. For a use-case considered in [BTV20], the length of Stadler arguments are 24.6 Kilobytes (KB) and those of Blazy *et al.* are 10.2KB long. As a first simple application of our conversion *in the head* technique, we present (for similar prover and verifier efficiency) arguments of size about 16.6KB (depending on the parameters). Even if this is longer than the previous approach, this still improves the communication complexity of Stadler's protocol by about 30%. By increasing the prover and verifier computational complexity, it is possible to decrease the communication complexity to 7.2KB (with better security guarantees than [BTV20]). It is worth mentioning that even by increasing the prover/verifier running times, the arguments of [Sta96, BTV20] cannot be shortened.

Permuted Kernel Problem (PKP): The PKP is a classical $\mathcal{NP}$-hard computational problem, where, given a matrix and a vector (of matching dimensions) defined over a finite field, one has to find a permutation of the vector coordinates that belongs to the matrix kernel. This problem was introduced in cryptography by Shamir [Sha90], who designed a zero-knowledge argument of knowledge of a solution of a PKP problem (and used it for a cryptographic post-quantum identification scheme). This protocol was improved subsequently in a long series of work [Ste94a, BFK$^+$19, Beu20, FJR23, Fen24, BG22]. We apply our technique to this problem and obtain a zero-knowledge argument of knowledge protocol which does not involve permutations that are not easy to implement securely, in particular in the presence of side-channel attacks.

One-way functions from "Fewnomials": A cryptographic one-way function $f : S \to S$ is a function that is computationally easy to compute but computationally difficult to invert. If S is a finite field (e.g. $S = \mathbb{F}_p$ for some prime number p), then it is well-known that f can be represented as a polynomial in $\mathbb{F}_p[X]$ (with degree upper-bounded by $(p-1)$). Ad hoc examples of such functions are cryptographic hash functions or functions derived from block ciphers (using for instance the Davies-Meyer construction [Win84]). Still, the polynomial representations of such functions are usually of very high complexity (which makes them not convenient for the MPCitH paradigm). Several works were devoted to designing efficient symmetric cryptographic primitives suitable for efficient implementation using MPCitH (*e.g.* the Picnic [CDG$^+$20, KZ22] and the Rainier [DKR$^+$22] signature schemes). As a third application of our technique, we propose a reverse approach to design a cryptographic system with simplicity and minimal complexity. The motivation is to remove potential points of failure and to obtain schemes easier to implement correctly. To do so, we consider the simplest polynomials defined over a finite field $\mathbb{F}_p$ that are good one-way function candidates. The simplest polynomials are certainly the monomials $f_1 : \mathbb{F}_p \to \mathbb{F}_p$, $x \mapsto f_1(x) = x^n \mod p$ but they are trivially not one-way. If n is coprime with $(p-1)$, this is a permutation on which one can apply the Davies-Meyer construction to obtain the binomials $f_2 : \mathbb{F}_p \to \mathbb{F}_p$, $x \mapsto f_2(x) = x^n + x \mod p$ which seem difficult to invert (the best-known algorithm

for $n = \Omega(p)$ has arithmetic complexity $O(p^{1/2})$ [BCR13]). More generally, a *fewnomial* is a term used in algebraic geometry and computational algebra, to describe a polynomial with a few terms (i.e. with a relatively low number of monomials compared to its degree). If one considers a fewnomial of high degree with $t \geq 2$ monomials over $\mathbb{F}_p$, the best known algorithm has arithmetic complexity $O(p^{(t-1)/t})$ [BCR13]. These candidate one-way functions are not suitable for symmetric cryptography (since evaluating them is much more costly than popular hash functions and block ciphers) but they are particularly interesting for our new conversion technique. In particular, we propose (candidate) post-quantum signatures with lengths of about 10.5KB. The produced signatures are thus not the shortest ones, but our goal with this application is to propose a new simpler, and cleaner one-way function suitable for the MPCitH paradigm with competitive performances and to motivate future research in this area.

5.2 Sharing Conversion and Design Principle

In the MPCitH paradigm, when the secret is shared additively, multiplicative relations are costly to prove, and vice versa. Whence converting secret sharing *in the Head* naturally comes to mind.

5.2.1 Sharing conversion technique

Let us denote $[\![\cdot]\!]$ as an additive sharing and $\langle\cdot\rangle$ as a multiplicative sharing. For the sharing conversion considered in the following, we need a uniformly random pre-computed couple of sharing $([\![r]\!], \langle s\rangle)$ such that $r = s \in \mathbb{F}^\times$. The N-party MPC protocol is the following:

Input: The parties have $\langle x\rangle$.
Output: The parties get $[\![x]\!]$.

Preprocessing phase: A trusted dealer generates two random sharings $r = \sum_{i=1}^{N} [\![r]\!]_i$ and $s = \prod_{i=1}^{N} \langle s\rangle_i$ such that $r = s$. They give $([\![r]\!]_i, \langle s\rangle_i)$ to party P_i for $i \in [1, N]$.
Online phase:
1. The parties compute $\langle\alpha\rangle = \langle x\rangle / \langle s\rangle$ and broadcast it.
2. The parties locally compute $\alpha[\![r]\!] := [\![x]\!]$.

Protocol 9: Sharing conversion protocol Π_{conv}

In practice, during the preprocessing phase, one starts by generating

$$\{[\![r]\!]_i\}_{1\leq i\leq N} \xleftarrow{\$} \mathbb{F}_q \text{ and } \{\langle s\rangle_i\}_{1\leq i\leq N} \xleftarrow{\$} \mathbb{F}_q^\times.$$

Then after defining $r = \sum_{i=1}^{N} [\![r]\!]_i$, one computes Δs such that $r = \Delta s \prod_{i=1}^{N} \langle s\rangle_i := s$. If $r = s = 0$, *i.e.* $\Delta s = 0$, one starts again. Correctness of Π_{conv} relies on the fact that if $r = s$, then it outputs an additive sharing of x. It provides security in the passive setting and α does not reveal any information on the parties' share thanks to the random choice of s.

5.2.2 General protocol

To remove the preprocessing phase from Protocol 9 and thus to be able to incorporate this technique in a prover-verifier interactive argument, we rely on the MPCitH *with helper* framework presented in Chapter 3 by following a cut-and-choose approach. We develop a 5-round protocol, presented in a general manner, and that can be adapted to each of the problems considered in the rest of this chapter.

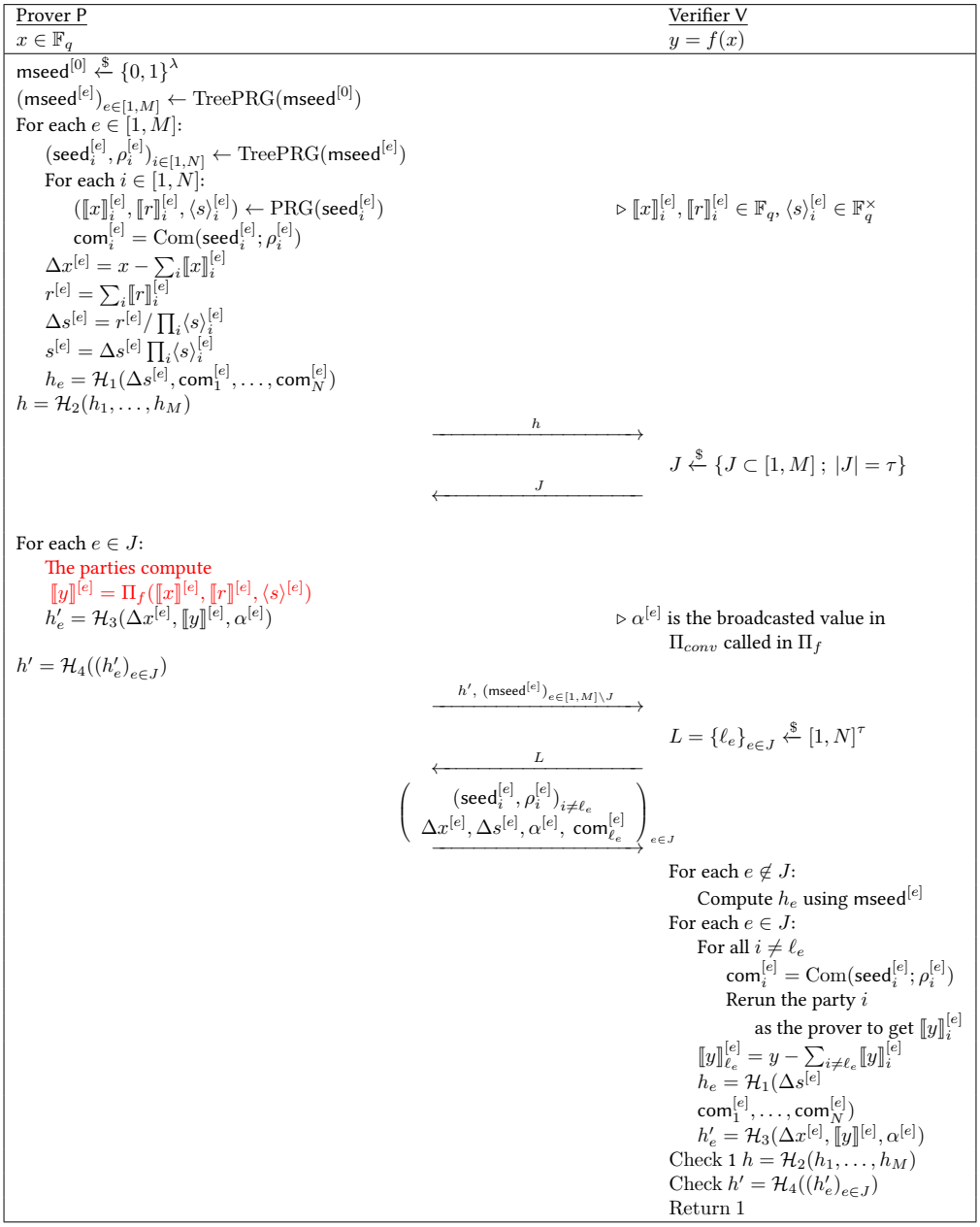
Let us consider a public statement from the following $\mathcal{NP}$-language:

$$\mathcal{L}_f = \{y \mid \exists x \text{ s.t. } f(x) = y\}$$

for some fixed one-way function f, and let Π_f be a passively secure N-party MPC protocol realizing the functionality as in Equation (3.1). More precisely, given a secret sharing input of x (either $[\![x]\!]$, $\langle x\rangle$, or a sharing on the integers from Chapter 4) and the public input y, the output of Π_f is 1 if $f(x) = y$ and 0 otherwise. Essentially, Π_f also takes as input a couple (or many couples) of secret sharing $([\![r]\!], \langle s\rangle)$ with $r = s \in \mathbb{F}_q^\times$. For the PKP application, Π_f takes as additional input, some prime number q' greater than q. The resulting interactive argument is presented with Protocol 10. The protocol makes use of a pseudo-random generator PRG, a tree-based pseudo-random generator $\mathrm{TreePRG}$, four collision-resistant hash functions $\mathcal{H}_i$ for $i \in [1, 4]$ and a commitment scheme $(\mathrm{Com}, \mathrm{Verif})$. The red part of the protocol has to be adapted depending on the problem considered. We choose to use $[\![\cdot]\!]$ in the protocol for the sharing of x and $f(x)$, but it can be substituted by $\langle\cdot\rangle$.

Soundness error. Let ϵ be the soundness of one repetition of the protocol. We perform τ parallel repetitions of the protocol to get a soundness error $\epsilon^\tau < (1/2)^\lambda$. As explained previously, each of these repetitions uses a cut-and-chose phase to prove the correctness of the helper. Instead of performing τ parallel cut-and-chose phases each resulting in trusting one couple of sharing $(\llbracket r^{[\ell]} \rrbracket, \langle s^{[\ell]} \rangle)$ among M, we follow the more efficient approach from [KKW18] and perform a global cut-and-choose phase resulting in τ trusted sharing among a larger M. As detailed in Chapter 3,the soundness error is then

$$\epsilon = \max_{M-\tau \leq k \leq M} \left\{ \frac{\binom{k}{M-\tau}\left(\frac{1}{N} + (1-\frac{1}{N})\beta\right)^{k-M+\tau}}{\binom{M}{M-\tau}} \right\}.$$



Protocol 10: Zero-knowledge argument for a pre-image of f.

For each conversion, when incorporating Π_{conv} into Protocol 10, there is one broadcast value α and one auxiliary value Δs to communicate, hence the sharing conversion protocol needs 2 field elements to communicate for each single conversion (we can not reuse the couple of sharing for another conversion).

Parameters selection. Recall that we are dealing with a preprocessing phase, that is proved with a cut-and-choose strategy. The total number of parties to set up is MN, which impacts the prover's computational complexity, therefore we choose sets of parameters that keep a reasonable running time. We start by fixing a number of parties N to be either 2^5 or 2^8. Then we look for the best trade-off between τ, M while keeping a soundness error below $2^{-\lambda}$. Decreasing τ

leads to better sizes but to higher M and so slower proofs. The MPC emulation does not impact a lot the running time, since the hypercube optimization is consistent with our scheme (see Chapter 3).

5.2.3 Legendre PRF

To begin with, we present a pseudo-random function (PRF) that we looked at because it was tempting to apply our sharing technique to it. Let p be an odd prime number. An integer a is a *quadratic residue* modulo p if it is congruent to a square number modulo p and is a quadratic nonresidue modulo p otherwise. The *Legendre symbol* is a function of a and p defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p \text{ and } a \not\equiv 0 \pmod{p}, \\ -1 & \text{if } a \text{ is a quadratic nonresidue modulo } p, \\ 0 & \text{if } a \equiv 0 \pmod{p}. \end{cases}$$

The *Legendre pseudo-random function* is a one-bit PRF $\mathbb{F}_p \to \{0,1\}$ defined using the Legendre symbol:

$$L_{p,K}(x) = \left(\frac{K+x}{p}\right) .$$

Legendre PRF Problem

Given p be an odd prime, and consider γ outputs $L_{p,K}(1), \ldots, L_{p,K}(\gamma)$, then the Legendre PRF Problem is to recover the secret key K.

For large γ, with overwhelming probability this K is uniquely defined. Let us share the secret K with an additive sharing, then an additive sharing of $K+1, K+2, \ldots, K+\gamma$ can be straightly derived. By applying our conversion technique to these γ additive sharings, we get the following multiplicative sharings $\langle K+1\rangle, \ldots, \langle K+\gamma\rangle$. By recalling the well-known multiplicativity of the Legendre symbol, *i.e.* $\left(\frac{x}{p}\right)\left(\frac{y}{p}\right) = \left(\frac{xy}{p}\right)$, we are able to distributively compute each symbol (this defines our MPC protocol). However, a priori one needs of γ conversions and this γ is large for security purposes (even by considering some relaxed PRF relation as in [BdSG20]). This makes our construction ineffective until we manage to introduce *e.g.* some batching into the conversions.

5.3 Proving Knowledge of a Double Discrete Logarithm

We start by proposing a direct application of our sharing conversion technique based on the Double Discrete Logarithm Problem (DDLP), which has found numerous applications in cryptography [CS97, ASM10, CG07, CGM16, BTV20]. However, even if the following zero-knowledge argument improves the state-of-the-art in terms of communication complexity (compared to schemes with the same assumptions), we present this protocol primarily for pedagogical purposes. Indeed, our zero-knowledge argument for the DDLP based on a forward-backward technique developed as an appetizer in Chapter 3 is more efficient than this proposal.

Double Discrete Logarithm Problem (DDLP)

Let $\mathbb{G}$ be a cyclic group of prime order q with some generator $g \in \mathbb{G}$, and let $h \in \mathbb{F}_q^*$ of prime order p with $p|(q-1)$. Given $(y,g,h) \in \mathbb{G} \setminus \{1_\mathbb{G}\} \times \mathbb{G} \times \mathbb{F}_q^*$, the *DDLP* asks to find some $x \in \mathbb{F}_p^\times$ such that $y = g^{h^x}$.

Consider the function $f : \mathbb{F}_p^\times \to \mathbb{G}, x \mapsto f(x) = g^{h^x}$ realizing the "double discrete exponentiation". We present an N-party MPC protocol Π_{DDLP} to securely compute the corresponding binary relation (via the computation of a multiplicative sharing of $f(x)$).

Input: $y \neq 1_\mathbb{G}$ in a cyclic group $\mathbb{G}$ of prime order q, $h \in \mathbb{F}_q^*$ of prime order p with $p|(q-1)$, and an additive sharing of $x \in \mathbb{F}_p^\times$.
Output: 1 if $y = g^{h^x}$, 0 otherwise.

1. Parties locally compute a multiplicative sharing $\langle h^x\rangle$ via $h^x = \prod_{j=1}^N h^{[\![x]\!]_j} \bmod q$.
2. Parties convert it into an additive sharing $[\![h^x]\!]$ over $\mathbb{F}_q$ using Π_{conv} 9.
3. Parties locally compute $\langle g^{h^x}\rangle$ via $g^{h^x} = \prod_{j=1}^N g^{[\![h^x]\!]_j}$.
4. Parties broadcast $\langle g^{h^x}\rangle$ and output 1 if $g^{h^x} = y$ and 0 otherwise.

Protocol 11: MPC protocol Π_{DDLP}

protocol	Parameters				Argument size (KB)
	$\log_2 q$	τ	N	M	
Protocol 11	2048	16	2^8	4096	16.6
Protocol 11	2048	17	2^8	1744	17.2
Protocol 4	3072	16	2^8	4096	5.6
Protocol 4	3072	17	2^8	1744	5.9

Tab. 5.1: Achieved performances of our zero-knowledge protocol for proving the knowledge of a solution of a DDLP instance.

The correctness of Π_{DDLP} comes from the fact that $h^x = h^{\sum_{j=1}^N [\![x]\!]_j \bmod p} = \prod_{j=1}^N h^{[\![x]\!]_j}$ since h has order p. The same reasoning holds for step 3 because g has order q. Plugging Π_{DDLP} into the red part of Protocol 10 with $\alpha^{[e]} := h^{[e]}/s^{[e]}$, we readily get a zero-knowledge argument of knowledge of a solution to the given DDLP instance. Note that we must slightly adapt Protocol 10 since $x \in \mathbb{F}_p$, and the output y is shared multiplicatively, but this is straightforward.

5.3.1 Performances

To estimate the communication complexity, we remark that for each iteration of the protocol, three values have to be communicated: the auxiliary value $\Delta x \in \mathbb{F}_p$ to fix the secret, and $(\Delta s, \alpha) \in \mathbb{F}_q^2$ from the sharing conversion protocol 9 (there is a sole conversion). This leads to a total communication cost of at most:

$$4\lambda + \lambda\tau \log_2 \frac{M}{\tau} + \tau\left[2\log_2(q) + \log_2 p + \lambda \log_2 N + 2\lambda\right] \text{ bits},$$

where M the number of parallel phases in the cut-and-choose, and τ the number of unrevealed phases (see Chapter 3).

In [BTV20], the authors considered the case of a group $\mathbb{G}$ of prime order $q = (4p+18)p+1$ where p is the Sophie Germain prime $p = 2^{1535} + 554415$ that divides $q-1$. Their arguments involve $2\lceil \log_2(2\lceil \log_2(\ell)\rceil + 1)\rceil + 8$ elements in $\mathbb{G}$ and 5 elements in $\mathbb{F}_q$. Taking $\mathbb{G}$ as the subgroup of order q in $\mathbb{F}_\ell^*$ for $\ell = 1572q+1$, one obtains an argument of size 10.2KB for [BTV20] and of size 24.6KB for [Sta96] (for a soundness error of 2^{-128}). We propose another set of parameters, more adapted to our scheme, by considering p, q as $\sim$2048-bit prime. Table 5.1 presents the communication complexity of our arguments. Note that they are always shorter than those from [Sta96] and provide better security guarantees than [BTV20]. Contrary to [BTV20], we could compress our argument size and construct parameter sets with argument size below 10KB (but at the cost of an increase in the computational complexity of the prover and the verifier). Moreover, the protocol 4 developed in Chapter 3 leads to an argument proof whose size is reduced by 77% compared to [Sta96], and beats the bulletproof approach of [BTV20].

5.3.2 Security proofs

Theorem 15. *Considered an instance $(y,g,h) \in \mathbb{G}\setminus\{1_\mathbb{G}\}\times\mathbb{G}\times\mathbb{F}_q^*$ of the DDLP. Then, the interactive protocol 10 combined with MPC protocol 11 is an honest-verifier zero-knowledge argument of knowledge for the relation $\{(x; g, h, y) \mid g^{h^x} = y\}$, with perfect completeness, and special soundness ϵ equals to*

$$\max_{M-\tau \le k \le M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau} N^{k-M+\tau}} \right\}.$$

Proof. Completeness. For any sampling of the random coins of P and V, if the computation described in the protocol 10 combined with protocol 11 is honestly performed, all the checks of V pass. The completeness is hence perfect.

Soundness. To prove the special soundness, one builds an efficient knowledge extractor that returns 3 specific transcripts, from which we can extract a solution of the DDLP instance. This extractor has rewindable black-box access to $\tilde{\mathsf{P}}$. Assume that we can get three transcripts $T_i = (h, J^{(i)}, \textsc{Rsp}_1^{(i)}, \{\ell_j^{(i)}\}_{j\in J^{(i)}}, \textsc{Rsp}_2^{(i)})$ for $i \in \{1,2,3\}$ from $\tilde{\mathsf{P}}$ with the same first commitment. We additionally require that there exists $j_0 \in (J^{(1)} \cap J^{(2)}) \setminus J^{(3)}$ such that $\ell_{j_0}^{(1)} \neq \ell_{j_0}^{(2)}$. Moreover, T_1 and T_2 are supposed to be successful transcripts (*i.e.* which pass all the tests of V). Finally, we assume that $\text{seed}^{[j_0]}$ from $\textsc{Rsp}_1^{(3)}$ is consistent with the $(x^{[j_0]}, r^{[j_0]}, s^{[j_0]})$ from T_1 and T_2.

We show how to extract a solution of the DDLP instance (y,g,h) from the three transcripts. First, we can assume that all the revealed shares are mutually consistent between the three transcripts. Otherwise, we find a hash collision (since they have the same first commitment). Thus, we know all the shares for the iteration j_0 from T_1 and T_2. For the sake of clarity, we only consider the variables of the j_0-th iteration, and this notation is omitted in the following. Consider $x' := \sum_{j=1}^N [\![x]\!]_j \bmod p$ as a natural candidate solution for x. Then, following the MPC protocol 11 we compute:

- $h^{x'} = h^{\sum_{j=1}^{N} [\![x]\!]_j} = \prod_{j=1}^{N} h^{[\![x]\!]_j} = \prod_{j=1}^{N} \langle h^x \rangle_j \bmod q$
- the broadcasting of $\langle \alpha \rangle = \frac{\langle h^x \rangle}{\langle s \rangle}$ i.e. $\alpha = \frac{h^x}{s} \bmod q$
- an additive sharing of h^x via $\alpha [\![r]\!] = \frac{h^x}{s} [\![r]\!] = [\![h^x]\!]$, since from the checked equations at the end of T_3 we get that $r = s$.
- $y = \prod_{j=1}^{N} \langle y \rangle_j \bmod q$ with $\langle y \rangle_j = g^{[\![h^x]\!]_j} \bmod q$.

Hence, $g^{h^{x'}} = g^{\prod_{j=1}^{N} \langle h^x \rangle_j} = g^{\sum_{j=1}^{N} [\![h^x]\!]_j} = \prod_{j=1}^{N} g^{[\![h^x]\!]_j} = \prod_{j=1}^{N} \langle y \rangle_j = y \bmod q$. Therefore, x' is a solution of the considered DDLP.

Our protocol 10 has the same structure as the protocol 5 in [FJR23], with the same first challenge for revealing $M - \tau$ out of M cut-and-choose phases, and the second challenge for opening $N - 1$ views. Hence, we can use the extractor described in appendix E of [FJR23] for producing the above three transcripts T_1, T_2, T_3 by calling in average at most

$$\frac{4}{\tilde{\epsilon} - \epsilon} \left(1 + \tilde{\epsilon} \frac{8M}{\tilde{\epsilon} - \epsilon} \right)$$

times $\tilde{\mathsf{P}}$ (the analysis of the average number of calls to $\tilde{\mathsf{P}}$ is also identical).

Honest-verifier zero-knowledge. We build a PPT simulator Sim (*i.e.* an algorithm that outputs transcripts that are indistinguishable from real transcripts without knowing a valid witness) given random challenges J and L (because we assume an honest verifier), and works as follows:

Simulator Sim:

1. Sample $J \xleftarrow{\$} \{J \subset [1, M]; |J| = \tau\}$ and $L = \{\ell_e\}_{e \in J} \xleftarrow{\$} [1, N]^\tau$
2. Sample $\mathsf{mseed}^{[0]} \xleftarrow{\$} \{0,1\}^\lambda$
3. $(\mathsf{mseed}^{[e]})_{e \in [1,M]} \leftarrow \mathrm{TreePRG}(\mathsf{mseed}^{[0]})$
4. For $e \in [1, M] \backslash J$, follow honestly the protocol and deduce h_e.
5. For $e \in J$:
   - Compute $(\mathsf{seed}_1^{[e]}, \rho_1^{[e]}), \ldots, (\mathsf{seed}_N^{[e]}, \rho_N^{[e]})$ with $\mathrm{TreePRG}(\mathsf{mseed}^{[e]})$.
   - For each party $j \in [1, N] \backslash \{\ell_e\}$: $([\![x]\!]_j^{[e]}, [\![r]\!]_j^{[e]}, \langle s \rangle_j^{[e]}) \leftarrow \mathrm{PRG}(\mathsf{seed}_j^{[e]})$, $\mathsf{com}_j^{[e]} = \mathrm{Com}(\mathsf{seed}_j^{[e]}; \rho_j^{[e]})$
   - Sample $\Delta x^{[e]} \xleftarrow{\$} \mathbb{F}_p$, $[\![x]\!]_{\ell_e}^{[e]} \xleftarrow{\$} \mathbb{F}_p$, $[\![r]\!]_{\ell_e}^{[e]} \xleftarrow{\$} \mathbb{F}_q$, $\langle s \rangle_{\ell_e}^{[e]} \xleftarrow{\$} \mathbb{F}_q^\times$
   - $\Delta s^{[e]} = \sum_{j=1}^{N} [\![r]\!]_j^{[e]} / \prod_j \langle s \rangle_j^{[e]} \bmod q$
   - $\alpha^{[e]} = h^{\sum_{j=1}^{N} [\![x]\!]_j^{[e]} + \Delta x^{[e]}} / (\Delta s^{[e]} \prod_{j=1}^{N} \langle s \rangle_j^{[e]}) \bmod q$
   - $\langle g^{h^x} \rangle_j^{[e]} = g^{\alpha^{[e]} [\![r]\!]_j^{[e]}} \bmod q$
   - Adapt the output of the party ℓ_e: $\langle g^{h^x} \rangle_{\ell_e}^{[e]} = y / \prod_{j \neq \ell_e} \langle g^{h^x} \rangle_j^{[e]} \bmod q$
   - Sample a random commitment $\mathsf{com}_{\ell_e}^{[e]}$
   - Compute $h_e = \mathcal{H}_1(\Delta s^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$, $h'_e = \mathcal{H}_3(\Delta x^{[e]}, \langle g^{h^x} \rangle^{[e]}, \alpha^{[e]})$
6. Compute $h = \mathcal{H}_2(h_1, \ldots, h_M)$, $h' = \mathcal{H}_4((h'_e)_{e \in J})$
7. Outputs the transcript

$$\left(h, h', (\mathsf{mseed}^{[e]})_{e \in [1,M] \backslash J}, ((\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i \neq \ell_e}, \mathsf{com}_{\ell_e}^{[e]}, \Delta x^{[e]}, \Delta s^{[e]}, \alpha^{[e]})_{e \in J}\right) .$$

The distribution of the output transcript is identical to a real one, except for the commitment of the party ℓ_e in each execution $e \in J$. Distinguishing them means breaking the commitment hiding property or the PRG security. □

5.4 Proving Knowledge of a PKP Solution

We denote by $\mathcal{S}_n$ the symmetric group of degree n. For a permutation $\pi \in \mathcal{S}_n$ and a vector $v \in \mathbb{F}_q^n$, $\pi(v)$ is the action of the permutation on the coordinates of v.

Permuted Kernel Problem (PKP/IPKP)

Let (q, m, n) be positive integers, $H \in \mathbb{F}_q^{m \times n}$ a random matrix, and a vector $v \in \mathbb{F}_q^n$. The PKP is to find a permutation $\pi \in \mathcal{S}_n$, such that $H\pi(v) = 0$. The inhomogeneous version of the problem (IPKP) is, given a target vector $y \in \mathbb{F}_q^m$, to find a permutation $\pi \in \mathcal{S}_n$, such that $H\pi(v) = y$.

We consider the PKP variant, but this work can be straightly extended for the IPKP (without loss of performances since y is public). We want to prove the knowledge of a solution to a PKP instance, *i.e.*, the knowledge of some $x \in \mathbb{F}_q^n$ and $\pi \in \mathcal{S}_n$ such that $Hx = 0$ and $\pi(v) = x$.

For this purpose, we adapt the protocol 10 as follows:

- the input $x \in \mathbb{F}_q^n$ is a vector, so we should consider one conversion by coordinate;
- the sharing of x is over the integers, so $[\![x]\!]_j \in [0, A-1]^n$ for some $A > q$. Thus, we should add a rejection rule as explained in Chapter 4;
- V sends an additional challenge $g \xleftarrow{\$} \mathbb{F}_{q'}^{\times}$ (as an evaluation point) at the same time as the challenge J, where q' is a prime greater than q whose choice is explained afterward.

5.4.1 A first approach for proving the knowledge of a permutation

Consider the polynomial $f_{x,v}(X) = \sum_{i=1}^n X^{x_i} - \sum_{i=1}^n X^{v_i}$ of degree at most $q-1$ (x_i, v_i denotes the components of the vectors x, v), and some uniformly random element $g \in \mathbb{F}_{q'}$. If $x = \pi(v)$ for some $\pi \in \mathcal{S}_n$, then $f_{x,v}$ is identically zero. If there is no permutation $\pi \in \mathcal{S}_n$ such that $\pi(v) = x$, then via the Schwartz-Zippel Lemma [Zip79, Sch80], the probability that $f_{x,v}(g) = 0 \bmod q'$ is bounded by $(q-1)/q'$. Indeed, the probability that a random element in $\mathbb{F}_{q'}$ is a root of a polynomial in $\mathbb{F}_{q'}[X]$ of degree at most $q-1$ is bounded by $(q-1)/q'$.

Initially, the sharing over the integers was introduced to share small values. In this work, when computing $f_{x,v}(g)$ over $\mathbb{F}_{q'}$ in a distributed way, the random challenge $g \xleftarrow{\$} \mathbb{F}_{q'}^{\times}$ may not satisfy $g^q = 1 \bmod q'$ and then employing a modular additive sharing would lead to a wrong computation. This is the motivation for using a sharing on the integers for x.

Recall that the verifier knows that $[\![x_i]\!]_j \in [0, A-1]$ (this is verified for open parties) and checks that $-A + q \leq x_i - [\![x_i]\!]_{j^*} \leq 0$. This implies that V is convinced by the fact that $-A+q \leq x_i \leq A-1$. In particular, V is not guaranteed that P chooses $x_i \leq q-1$, and then the degree of $f_{x,v}$ is a priori bounded by $A-1$, whence a *slack*. Indeed, a malicious prover may choose some x whose coordinates are upper bounded by $A-1$ instead of $q-1$. By taking the modulus q' large enough compared to A, with the Schwartz-Zippel Lemma, we can achieve a small probability of picking a random element in $\mathbb{F}_{q'}$ as a root of $f_{x,v}$ whose degree is bounded by $A-1$ (when $f_{x,v}$ is not identically zero).

MPC protocol. We describe the MPC protocol to plug in the red part of protocol 10. As input, x is shared among the parties via a secret sharing on the integers, *i.e.*, $[\![x]\!]_j \xleftarrow{\$} [0, A-1]^n$ for $j \in [1, N]$. The rejection rate of the sharing is $1 - \left(1 - \frac{q-1}{A}\right)^n$ (see Chapter 4 with application to the Boneh's PRF 4.5.3) since we are sharing elements of $\mathbb{F}_q$ instead of binary elements. Parties also get some $g \in \mathbb{F}_{q'}^{\times}$ with q' a prime number greater than $\beta(A-1)$, where $1/\beta$ is the false positive probability of the MPC protocol (*i.e.* the probability to randomly pick a root of a non-zero polynomial is bounded by $1/\beta$). We present the following MPC protocol Π_{PKP} to securely compute the corresponding binary relation via the computation of a sharing of $\{Hx, f_{x,v}(g)\}$.

Input: $x \in \mathbb{F}_q^n$ shared on the integers as $x = \sum_{j=1}^{N} [\![x]\!]_j$, $H \in \mathbb{F}_q^{m \times n}$, $g \xleftarrow{\$} \mathbb{F}_{q'}^{\times}$ with q' the next prime after $\beta(A-1)$.
Output: 1 if $Hx = 0 \bmod q$ and $\pi(x) = v$ for some $\pi \in \mathcal{S}_n$, 0 otherwise.

---

1. From the sharing over the integers of each x_i, parties locally compute $\langle g^{x_i} \rangle$, a multiplicative sharing of $g^{x_i} = \prod_{j=1}^{N} g^{[\![x_i]\!]_j} \bmod q'$, for each $i \in [1, n]$.
2. Parties convert it into an additive sharing $[\![g^{x_i}]\!]$ using Π_{conv} 9, for each $i \in [1, n]$.
3. Parties locally compute their share of $[\![f_{x,v}(g)]\!] = \sum_{i=1}^{n} [\![g^{x_i}]\!] - \sum_{i=1}^{n} g^{v_i} \bmod q'$.
4. Parties locally compute their share of $[\![Hx]\!] = H[\![x]\!] \bmod q$.
5. Parties broadcast $[\![Hx]\!]$ and $[\![f_{x,v}(g)]\!]$. If $Hx = 0$ and $f_{x,v}(g) = 0$, output 1, otherwise 0.

Protocol 12: MPC protocol Π_{PKP}

Notice that the correctness of $g^{x_i} = g^{\sum_{j=1}^{N} [\![x_i]\!]} \bmod q'$ follows from the sharing on the integers. For each coordinate one conversion is required, thus two values over $\mathbb{F}_{q'}$ has to be communicated to V in addition to one auxiliary value for the secret coordinate (over $[0, A-1]$). Hence, the obtained argument size is

$$4\lambda + \lambda\tau \log_2 \frac{M}{\tau} + \tau \left[n(2 \log_2 q' + \log_2(A-1)) + \lambda \log_2 N + 2\lambda \right] \text{ bits},$$

where M is the number of parallel phases in the cut-and-choose, and τ the number of unrevealed phases.

The security of the PKP/IPKP has been well-studied for many years. We consider the parameter sets proposed in [BFK+19] to achieve 128 bits of security, *i.e.* $n = 61$, $m = 28$, $q = 997$. The choice of the remaining parameters τ and M are chosen as a trade-off between argument size and signing speed. Let $1/\beta$ be the false positive probability of the MPC protocol 12 (*i.e.* when checking the existence of a permutation). We fix $\beta = 2^8$, thus q' can be chosen as the next prime after $2^8(A-1)$.

Parameters				Argument size (KB)	Rej. rate
τ	N	A	M		
19	2^8	$2^{14}q$	1289	13.3	0.068
27	2^5	$2^{14}q$	541	16.9	0.096

Tab. 5.2: Obtained performances for proving the knowledge of a witness of a PKP instance.

5.4.2 Security proofs

Theorem 16 (Security Proofs). *Considered an instance $(H, v) \in \mathbb{F}_q^{m \times n} \times \mathbb{F}_q^n$ of the PKP. Then, the identification scheme protocol 10 combined with MPC protocol 12 is an honest-verifier zero-knowledge argument of knowledge of $(x, \sigma) \in \mathbb{F}_q^n \times \mathcal{S}_n$ such that $Hx = 0$ and $\sigma(x) = v$, with $\left(1 - \frac{q-1}{A}\right)^{\tau\, n}$-completeness and special soundness ϵ equals to*

$$\max_{M-\tau \le k \le M} \left\{ \frac{\binom{k}{M-\tau} \left(\frac{1}{N} + (1 - \frac{1}{N}) \frac{1}{\beta} \right)^{k-M+\tau}}{\binom{M}{M-\tau}} \right\}.$$

Proof. Completeness. For any sampling of the random coins of P and V, if the computation described in the protocol 10 (with the slight adaptation described in section 5.4 for the first challenge and the rejection rule) combined with MPC protocol 12 is honestly performed and if there is *no abort*, all the checks of $\mathcal{V}$ pass. Since the probability of abortion is $1 - \left(1 - \frac{q-1}{A}\right)^{\tau\, n}$ (see Chapter 4), the completeness probability is hence equals to $\left(1 - \frac{q-1}{A}\right)^{\tau\, n}$.

Special soundness. Compared to the DDLP identification scheme, the ongoing scheme has a larger first challenge $J \cup \{g \xleftarrow{\$} \mathbb{F}_{q'}\}$ where J is the cut-and-choose challenge, but the second challenge is the same. Consider the same extractor E as for Theorem 15 which outputs some specific three transcripts T_1, T_2, T_3. Define $J^{(i)} \cup \{g^{(i)} \xleftarrow{\$} \mathbb{F}_{q'}\}$ the first challenge in transcript T_i for $i \in [1, 3]$. Thus, $J^{(1)}$, $J^{(2)}$ and $J^{(3)}$ satisfy the conditions enumerated in proof of Theorem 15, but $g^{(1)}$, $g^{(2)}$ and $g^{(3)}$ are random. As detailed in proof of Theorem 15, T_1 and T_2 aims to recover some natural candidate solution x'. T_3 provides a valid couple of sharing $([\![r]\!], \langle r \rangle)$. All together, we can reconstruct the polynomial $f_{x',v}(X)$. Since T_1 is successful, $f_{x',v}(g^{(1)}) = 0 \bmod q'$.

Honest-verifier zero-knowledge. We detail the change to carry out in the simulator Sim from Theorem 15. First, we shall sample $g \xleftarrow{\$} \mathbb{F}_{q'}$ during step 1. We adapt step 5 accordingly to MPC protocol 12. Finally, we add an additional step "Abort with probability $1 - \left(1 - \frac{q-1}{A}\right)^{n\tau}$" before step 7. The final analysis still holds. □

5.4.3 Other analysis and approach

Conjecture of [Kel16]. We analyze the protocol from Subsetion 5.4.1 by replacing the Schwartz-Zippel lemma use with a finer probability of randomly drawing a root.

Let F be the set of polynomials over $\mathbb{F}_{q'}$ of degree less than $q'-1$, with t terms, and which does not vanish on any entire coset of any nontrivial subgroup of $\mathbb{F}_{q'}^*$. Let $R(f)$ be the number of distinct non-zero roots of a polynomial f in $\mathbb{F}_{q'}$. Then the conjecture of [Kel16] states that

$$R := \max\{R(f) : f \in F\} = O(t \ln q'),$$

where $\ln$ is the natural logarithm. More precisely, they conjecture that there exists some constant $\gamma > 0$ such that

$$R \leq 2t/\gamma \ln q',$$

and this γ has been shown heuristically that it is greater than or equal to $1/2$.

In our case, $t = 2n$ (n has typically 6 bits), and we want to have a probability of picking a root of $f_{x,v}(X) \in \mathbb{F}_{q'}[X]$ less than $1/\beta$ with $\beta = 2^8$, ie $R/q' \leq 1/2^8$ (to obtain the same false positive probability of the MPC protocol than in Subsection 5.4.1). The conjecture tells us to use primes of at least 21 bits, and since A is around 24 bits, we can consider $q' \simeq A$ ($f_{x,v}$ has degree less than A, thus we shall have $q' > A$). Compared to the first approach, this conjecture ables us to work over a smaller field (we save around $\log \beta = 8$ bits on the size of the field $\mathbb{F}_{q'}$) leading to better performances.

The modulus as a challenge. A different approach could be to consider the modulus q' as a challenge, *i.e.*, the field point g is now fixed (such that g is not a root of $f_{x,v}$ on $\mathbb{Z}$ for $f_{x,v} \not\equiv 0$), and the question is to control the probability that some uniformly random prime divides $f_{x,v}(g)$. For this purpose, we can fix a some subset of primes $S \subset \mathbb{P}$ such that

$$\Pr[f_{x,v}(g) = 0 \bmod q \mid q \xleftarrow{\$} S, f_{x,v}(g) \neq 0] \leq 2^8.$$

Although this approach seems to be advantageous in terms of proof size, it does not fit well with the cut-and-choose strategy since we should change the prime q' for each iteration of the MPCitH, and so derive τ different cut-and-choose instead of one (the last optimization from Chapter 3) can not be applied anymore.

5.5 Proving Knowledge of a Fewnomial Pre-Image

We propose a new (candidate) post-quantum one-way function and a digital signature scheme constructed as an argument of knowledge of a pre-image of this function. Our goal is to design a simple and somewhat minimalistic scheme.

We consider a prime number p and the simplest one-way polynomials defined over the finite field $\mathbb{F}_p$. Those polynomials are called *fewnomials* and are simply polynomials with a relatively low number of monomials compared to their degree. If one considers a fewnomial with $t \geq 2$ monomials of large degrees over $\mathbb{F}_p$, the best known classical algorithm has arithmetic complexity $O(p^{(t-1)/t})$ [BCR13]. Essentially, this is the running time of their algorithm for detecting the existence of a root in $\mathbb{F}_p$ for univariate t-nomials. Combining this algorithm with Grover's algorithm [Gro96], leads to the best-known quantum algorithm with complexity $O(p^{(t-1)/2t})$.

A prime number q is called Sophie Germain prime is $2q+1 := p$ is also a prime number. These prime numbers of the form $2q+1$ are called safe prime numbers since the multiplicative group of integers modulo $2q+1$ has a subgroup of large prime order.

Fewnomial Inversion Problem (FIP).

Let q be a Sophie Germain prime number where $p = 2q+1$ is also a prime number. Let $t \geq 2$ be an integer and $f : \mathbb{F}_p \to \mathbb{F}_p$ be a fewnomial with t monomials defined as $f(X) = \sum_{i \in S} X^i$ where S is a set of t integers in $[\lceil q/2 \rceil, q-1]$. The Fewnomial Inversion Problem is given $y = f(x) \in \mathbb{F}_p$ to find $x' \in \mathbb{F}_p$ such that $y = f(x')$.

We construct a digital signature scheme based on the hardness of the FIP. Note that we consider the case of unitary monomials but adding non-zero (public) coefficients does not change the following analysis and performances. The choice of t and p will be discussed later on. It is worth mentioning that the monomial $X^n \bmod p$ would be easy to invert except if we replace the prime p by a modulus with unknown factorization, and this would be essentially an RSA instance with a larger modulus.

5.5.1 Protocol and performances

We first present the MPC protocol Π_{FIP} to plug in protocol 10, in which parties securely compute the corresponding binary relation via the computation of a sharing of $f(x)$. The prover/signer starts by sharing the secret/signing key x multiplicatively.

Input: A multiplicative sharing of $x \in \mathbb{F}_p^\times$, *i.e.* $x = \prod_{j=1}^{N} \langle x \rangle_j \bmod p$. A fewnomial $f : X \to \sum_{i \in S} X^i$, with a finite subset S of t integers in $[\lceil q/2 \rceil, q-1]$, and some public value $y \in \mathbb{F}_p$.
Output: 1 if $f(x) = y$, and 0 otherwise.

1. Parties locally compute $\langle x^i \rangle$ via $x^i = \prod_{j=1}^{N} \langle x \rangle_j^i \mod p$ for $i \in S$.
2. For each $i \in S$, parties convert $\langle x^i \rangle$ into an additive sharing $[\![x^i]\!]$ using Π_{conv} 9.
3. Parties locally compute $[\![f(x)]\!] = \sum_{i \in S} [\![x^i]\!]$.
4. Parties broadcast $[\![f(x)]\!]$, and output 1 if $f(x) = y$ and 0 otherwise.

Protocol 13: MPC protocol Π_{FIP}

In the MPC protocol 13, parties apply the conversion protocol for each $i \in S$ and each conversion requests to communicate 2 field elements as explained in Section 5.2. The rest of the communication is standard. Hence, the communication cost of the protocol is

$$4\lambda + \lambda\tau \log_2 \frac{M}{\tau} + \tau \left[(1+2t) \log_2 p + \lambda \log_2 N + 2\lambda \right] \text{ bits},$$

where t is the size of S, M the number of parallel phases in the cut-and-choose, and τ the number of unrevealed phases (see Section 5.2).

5.5.2 Security proofs

Theorem 17 (Security Proofs). *Considered an instance $(f, y) \in \mathbb{F}_p[X] \times \mathbb{F}_p$ of the FIP. Then, the interactive protocol 10 combined with MPC protocol 13 is an honest-verifier zero-knowledge argument of knowledge for the relation $\{(x; f, y) \mid f(x) = y\}$, with perfect completeness, and special soundness ϵ equals to*

$$\max_{M-\tau \le k \le M} \left\{ \frac{\binom{k}{M-\tau}}{\binom{M}{M-\tau} N^{k-M+\tau}} \right\}.$$

Proof. Proofs of completeness, special soundness, and honest-verifier zero-knowledge are identical to those of Theorem 15 (they are both based on the same protocol 10), but require a slight but natural adaptation. In the soundness, we adapt to the FIP the proof that the natural solution x' extracted from the first two transcripts satisfies $f(x') = y \bmod p$. In the simulator Sim, step 5 has to be adapted according to the FIP. □

5.5.3 Digital signature based on the FIP

We apply the Fiat-Shamir transform [FS87] to get a non-interactive protocol, and so a signature scheme. Since our protocol has 5 rounds, we have to take into consideration the attack of Kales and Zaverucha [KZ20] for the security of the signature. More precisely, the attack can be set up as long as the number of rounds is greater than 3, and its complexity is influenced only by the size of the challenge spaces. The forgery cost of the signature scheme is then given by

$$\min_{M-\tau \le k \le M} \left\{ \frac{\binom{M}{M-\tau}}{\binom{k}{M-\tau}} + N^{k-M+\tau} \right\}.$$

To build a signature, we choose $x \in \mathbb{F}_p^\times$ as the private key and $y = f(x) \bmod p$ as the public key. To achieve a forgery cost of $1/\epsilon$, we could increase τ, but this would not lead to an efficient scheme. Instead, we transform our 5-round protocol into a 3-round before applying the Fiat-Shamir transform, hence [KZ20] attack does not apply anymore. The 5-to-3-round convert's idea is to emulate M MPC protocols before the first round of communication, *i.e.*, before getting the challenge for the cut-and-choose. After values are committed, V sends both challenges during the same round, and after receiving the P's response, V can realize the verification as before. The reduction of the number of rounds comes at the cost of emulating $M - \tau$ additional MPC protocols since P does not know yet the challenge subset τ when it has to emulate MPC protocols. It leads to an overhead in terms of signing speed, *i.e.*, there are $M(1 + \log_2(N))$ parties to

emulate instead of $\tau(1 + \log_2 N)$, but the hypercube optimization attenuates it. Moreover, the communication cost is slightly greater for the 3-round version, the size of the signature scheme is then

$$4\lambda + 3\lambda\tau \log_2 \frac{M}{\tau} + \tau\left[(1+2s)\log_2 p + \lambda \log_2 N + 2\lambda\right] \text{ bits},$$

with s the size of $S^{\times}$, M the number of parallel phases in the cut-and-choose, and τ the number of unrevealed phases. The resulting 3-round protocol is also an honest-verifier zero-knowledge proof with the same soundness. It can be checked that the round reduction described here does not impact the proofs of Theorem 17 (*i.e.* the proofs of Theorem 15).

Performances. As mentioned above, considering a fewnomial with $t \geq 2$ monomials over $\mathbb{F}_p$, the best (classical) known algorithm has arithmetic complexity $O(p^{(t-1)/t})$ [BCR13]. Hence, there is a trade-off between the size of the modulus p and the number of monomials t to consider achieving (classical) 128 bits of security. The optimal one to minimize the signature size is a trinomial over a prime of 170 bits.

Parameters			Signature size (KB)
τ	N	M	
28	2^5	389	12.2
18	2^8	1251	10.6

Tab. 5.3: Achieved signature size based on the FIP.

6. THRESHOLD PROOFS FROM SECURE MULTIPARTY COMPUTATION

6.1 Introduction

In the mosaic of modern cryptography, distributed computation and interactive zero-knowledge proofs both serve as the backbone of many cryptographic systems. Bringing these two concepts together leads to customary use cases for interactive proofs in a distributing setting, where a set of users must conjointly prove computational tasks to some verifier in an adversarial environment. This proof distribution among multiple users has been studied for a while [BOGKW88] with the idea of sharing the trust in mind. Most of the works up to now studied the situation where all the users in the model have to be part of the proof to succeed (either in passive or active security). However, the non-trivial threshold case — where only a subset of the users is enough to realize a valid proof — is particularly interesting in terms of security for removing points of failure while guaranteeing that a sufficient large subset of users has validated the statement. Hence, a natural question comes to mind:

Given a generic $\mathcal{NP}$-statement, how to conceive a threshold zero-knowledge proof (of knowledge) of a shared witness?

Therefore, provers should own a shared secret input, and this restricts the number of available references in the literature. As an additional security requirement, we would like that even if a bounded subset of provers deviates from the protocol, a valid proof can still be produced. However, satisfying solutions to the previous challenge when dealing with threshold proof (unlike with multi-prover proof) stand as an open-ended question. Specifically, a generic building system for zero-knowledge proofs of a solution to the distributed analog of any $\mathcal{NP}$-problem, in active security, without adding computational hypothesis than the hardness of the considered $\mathcal{NP}$-problem. As an example, the IACR website specifies additional considerations for the *E-Voting Systems for the IACR*. One of them is "minimizing the level of trust that is required of any entity in the system. (For example, servers' secrets could be shared with a 3-out-of-5 threshold, corresponding to the three people required for ballot-counting by the current version of the IACR bylaws)". This is a typical use-case for a threshold zero-knowledge proof.

From one-prover to multi-prover zero-knowledge proof systems. A zero-knowledge proof system is a family of interactive proof system [Bab85, GMR89], consisting of an all-powerful prover who attempts to convince a probabilistic polynomial-time verifier of the truth of a statement. The prover and verifier receive a common input and can exchange up to a polynomial number of messages (in the security parameter), at the end of which the verifier either accepts or rejects the statement. In the literature, the extension of the notion of an interactive proof system to multi-prover exists and leads to the usually named *multi-prover zero-knowledge proof* system [BOGKW88]. It was first more of a theoretical notion. Multiple provers are allowed to communicate with each other before the beginning of the interaction with the verifier, but once the interaction starts, they are assumed to be isolated. This strong condition gives the model additional power. Indeed, intuitively, the verifier will be much more confident if provers answer their question without signaling with each other. Hence, the soundness property protecting the verifier assumes that the provers can not communicate with each other during the proof process, leading to a physical separation between the provers. This also aims to build perfect zero-knowledge proofs independently of any complexity-theoretic assumptions. However, Crépeau and Yang [CY19] noticed that special attention must be paid to soundness proofs in this paradigm, since for instance, provers may be able to indirectly exchange information via the verifier. This has been an additional motivation for us to consider a model tolerating (corrupted) provers to signal even if the protocol does not require communication.

Threshold motivation. Given a multi-prover proof system, its thresholdization is a natural approach to mitigate the risks associated with single points of failure and ensures that a certain number of users must collaborate to produce a valid proof. A first example is threshold digital signatures [Des87, Des94], where the signing authority is delegated to a set of users such that any large enough subset of them are able to conjointly succeed in producing a valid signature. Another example is threshold verifiable decryption, a cryptographic primitive that distributes the decryption authority among multiple users. On top of decrypting a ciphertext, many applications require proving that the decryption has been correctly handled without revealing the secret key. This is termed verifiable decryption, and examples include anonymous communication, decryption of ballots in electronic voting, and various uses of verifiable fully homomorphic encryption.

To a larger extent, with the advent of quantum computers, post-quantum schemes gained importance, but many of the prominent known threshold schemes are not post-quantum. Hence, a generic construction for threshold zero-knowledge

proofs, where its security can be reduced to the chosen hard-problem, is of high interest. This work also meets the recent call of the NIST for post-quantum threshold schemes.

Prior work. The objective of this chapter being natural, several papers have proposed solutions in that direction and looked at relatively close questions.

Threshold proofs for homomorphism preimage. A somewhat olden work [KMR12] proposes techniques for constructing threshold protocols, including Schnorr's protocol for discrete logarithms, and Guillou-Quisquater's protocol for modular roots, via a proof of knowledge of a preimage of a homomorphism. However, this construction only stands for group homomorphisms. Indeed, they built a naive-distributed version of the single-prover proof: let $\phi : G \to H$ be a group homomorphism between two groups G and H, assume that the secret input belongs to G and is shared over G. Then each prover's computation is basically the homomorphsim evaluation of their share, leading to a sharing of the output over H. Hence, considering a threshold linear secret sharing over G, we directly get a threshold proof. To match with their objective of efficiency, their protocol requires an additional user named a combiner, combining the messages of the provers and communicating with the verifier.

Negative results in the all-but-one corruption setting. Recently, [DKR23] provides negative results when considering the non-interactive setting of our multi-prover context. In particular, proofs with security against all-but-one corruption can not achieve sizes sublinear in the number of provers in the random oracle model. Therefore, authors consider the relaxed corruption condition of a constant fraction of provers and work by example by focusing their attention on a specific language for the discrete logarithm problem.

The next papers are a series of recent works that are closer to ours, both in the approach and in the results. Note that works that have been done in this area, aiming to outsource computation or designed for applications where provers do not hold private inputs, do not compete with ours.

Publicly-Auditable-MPC (PA-MPC). It is natural to compare our work with the paper [BDO14], which introduced PA-MPC by extending MPC with a publicly verifiable proof, ensuring that the computation was performed correctly with respect to the commitments to the inputs. It is achieved by publishing a transcript of the protocol that anyone can read and be convinced. This approach differs from previous works in two ways: (1) they look at the regime where all the servers are corrupted; (2) anyone with access to the transcript can be the auditor and does not need to be online while the protocol is executed. [BDO14] developed a PA-MPC with a high-speed online phase by relying on SPDZ techniques, hence on a trusted setup setting. But this offline should be publicly verifiable as well, thus they consider Pedersen commitment in order to make SPDZ auditable, and use a common reference string (CRS) for generating these commitments. If this setup is compromised, the security of the entire protocol can be at risk.

Collaborative zk-SNARKs. Classic PA-MPC constructions [BDO14] have linear size proofs. Boneh and Ozdemir [OB22] introduced collaborative zk-SNARKs, which turns out to be an efficient way of building PA-MPC protocols with constant size proofs. Their security model is quite similar to ours, except that they focus on efficient verifier. Their construction starts from a single-prover zk-SNARK, to then run its proof generation algorithm as an MPC among the provers. Since running it naively through a general purpose MPC protocol would lead to poorly performances, they focus on some MPC-friendly zk-SNARKs in the sense that proof generation lends itself to a very efficient MPC protocol. The main difference with our work is (1) they rely on additional computational hypothesis (they instantiate it with SNARKs such as Groth16 [Gro16b], vanilla PLONK [GWC19], Marlin [CHM$^+$20] that are vulnerable to quantum computers); (2) it requires communication between provers with a conditional $\Omega(n)$ lower bound on the communication where n is the number of provers; (3) a collaborative zk-SNARK only makes sense in a setting where all provers want to jointly generate a proof leading to the multi-prover proof setting, and not achieving a strict threshold. Therefore, collaborative zk-SNARK is particularly effective in settings where all provers are committed to jointly generating a proof, as opposed to a strict threshold setting where only a subset of provers is involved. Moreover, their systems all require the same trusted setup as the zk-SNARK they build on (hence, a Fractal-based collaborative zk-SNARK would not require a trusted setup).

One key difference with our approach is that we focus on building a generic system that achieves broad security properties with minimal assumptions, whereas they concentrate more on efficiency, providing an in-depth analysis of it.

PA-MPC-as-a-Service. The work of [KZGM21] emphasizes MPC-as-a-Service by developing a PA-MPC protocol using Marlin [CHM$^+$20], positioning itself as a concurrent work to [OB22]. The three points that differentiate our work from [OB22] are also applicable to [KZGM21]. Although they rely on a trusted setup, only a single trusted ceremony is required for the system's entire lifetime, even when programs are updated dynamically. Moreover, their construction for auditable MPC necessitates a stronger variant of commitment schemes known as extractable trapdoor commitment.

Publicly Accountable Robust MPC. Finally [RRRK22] explicitly focuses on a form of threshold collaboration with robust security. The authors improve upon traditional SPDZ-like protocols by integrating threshold secret-sharing and ensuring compatibility with a suitable homomorphic commitment scheme. They replace the Pedersen commitment with a lattice-based commitment scheme to achieve better integration with the lattice-based BGV encryption used in SPDZ. Consequently, they believe their protocol, with small modifications, could provide post-quantum security. Additionally, they utilize a preprocessing phase for a common reference string.

Our contributions. We consider the active setting in a security model tolerating an adversary corrupting at most t provers. In this context, an MPC protocol is said to satisfy the properties of *integrity* if the computation has been correctly performed (*i.e.*, the correct function has been evaluated in the correct inputs), and of *availability* if the computation has been completed. Our threshold proof involves k provers and a verifier and can be viewed as a $(k+1)$-party MPC protocol. To ensure the correctness property of our proof, the MPC protocol must achieve availability when the number of corrupted parties is less than $t+1$. For the soundness, even if the number of corrupted provers exceeds this threshold t, the verifier should not be convinced when the statement is false, in other words, the MPC protocol must have integrity. This meets the PA-MPC protocol [BDO14] that achieves public verification even if all the servers involved in the computation are corrupted. Here is a summary:

	$\#$ corrupted provers $\leq t$	$\#$ corrupted provers $> t$
with-abort	integrity	
with-abort auditable	integrity	integrity
robust	integrity, availability	
robust auditable	integrity, availability	integrity

Tab. 6.1: Achieved properties of different classes of MPC protocols in our security model depending on the number of corrupted provers.

Therefore, we have to target a notion relative to robust auditable.

A straightforward solution to build such a robust auditable proof is to require each prover to commit to all their secret inputs and provide a zero-knowledge proof for every message sent, demonstrating that the message was computed according to the protocol. If a common reference string is available, non-interactive zero-knowledge proofs can be used, allowing anyone to verify the proofs at any later time. However, this method introduces substantial computational overhead, resulting in a highly inefficient protocol. We could follow two approaches to get more efficient protocols, both involving robust MPC among provers and making the results publicly verifiable. One option treats the provers network as an k-server setup where servers conjointly perform computations and create a public file for audit purposes, like in PA-MPC. Alternatively, in an $k+1$-server model, k servers conjointly realize a computation and produce individual proofs for verifiability with the last server acting as the verifier. Eventually, we can turn them into non-interactive proofs. E.g., these proofs could be based on zk-SNARKs or MPCitH. Although it remains unclear how to effectively handle SNARKs for proving the k-party computation, as opposed to MPCitH-style proofs.

Previous introduced works follow the first approach. Even [OB22] since they choose a zk-SNARK, use its generation function, and build an n-party MPC to compute this function, ultimately outputting 0/1. We follow the second approach, and we achieve the properties detailed in Table 6.2.

	robust	threshold	preprocessing	post-quantum
[BDO14]	no	no	yes	no
[OB22]	no	no	yes $(\star)$	no $(\star\star)$
[KZGM21]	no	no	yes $(\star)$	no $(\star\star)$
[RRRK22]	yes	yes	yes	conjecture
our proof system	yes	yes	no	yes

Tab. 6.2: Comparison of exisiting works

$(\star)$: While most zk-SNARK systems rely on a trusted setup to generate specific parameters (often referred to as the structured reference string), some of these systems are transparent and could be use by these works.

$(\star\star)$: These works could use post-quantum zk-SNARKs, but they were focusing on efficiency with SNARKs having an MPC-friendly generation function.

Regarding the preprocessing, most existing works rely on setup components that are assumed to be pre-distributed at the start of the protocol. While we relax this strict assumption—since they are not required in our approach—we do assume that the private keys/witnesses are distributed in advance. Note that among existing auditable MPC protocols, we focus more on comparing the achieved properties and required assumptions than on studying efficiency.

Road map. Sections 6.2 and 6.3 contain the main theoretical novelty.

In particular, Section 6.2 involves the security definitions for a threshold zero-knowledge proofs (TZKP). This system is generic in the sense that, given any $\mathcal{NP}$ statement, we can produce a TZKP for it. Proofs in that system achieve robust security in the active setting, since it seems to be a must for future applications. Indeed, it tolerates the presence of a malicious adversary that may corrupt a constant subset of provers, while ensuring any large enough subset of provers to produce a valid proof.

Section 6.3 proposes a black-box construction for building a TZKP, employing the MPC-in-the-Head machinery. This is based on two layers of MPC. The first one corresponds to computation among provers. Even if the security model does not assume it, in practice communications among provers cease before the first interaction with the verifier. The second layer is composed of protocols emulated in the head of provers in order to make the first layer of MPC verifiable.

In Section 6.4, we instantiate our black-box construction with the popular BGW [BGW88] protocol with some verifiable secret sharing.

Finally, Section 6.5 discusses effective variants of TZKP, as well as their potential applications, in particular when the $\mathcal{NP}$-problem holds a low-depth arithmetic circuit. This leads to proofs without communication among provers (even if the security model tolerates signaling). Since our proof system is particularly valuable when a secret key is shared among the provers, application to threshold digital signatures and threshold verifiable decryption are examined. Its versatility and simplicity offers a new approach for effective, post-quantum threshold protocols, leveraging recent advancements in the MPCitH world. As an additional motivation for this work, the security of the resulting proofs does not reduce to other computational hypothesis than the hardness of the chosen $\mathcal{NP}$-problem (assuming secure commitment scheme and pseudo-random generator).

6.2 Threshold Zero-Knowledge Proof System

We introduce the threshold zero-knowledge proof system TZKP as a $(k+1)$-party interactive proofs, featuring k provers and one verifier. As a first step, we generalize the notion of a binary relation to accommodate multiple witnesses in a distributed setting.

Definition 16 (Distributed analog of $\mathcal{NP}$-language)**.** *Let $x \in \mathcal{L}(\mathcal{R})$ be a word of an $\mathcal{NP}$-language with binary relation $\mathcal{R}$, i.e. there exists some witness w such that $(x, w) \in \mathcal{R}$. Let us choose any LSSS from Definition 9 with $\mathbb{V}_2$ the share space and t the threshold. We define the corresponding $(|I|+1)$-ary threshold relation $\mathcal{R}^{LSSS}$ as*

$$\mathcal{R}^{LSSS}(x, \{w_i\}_{i\in I}) := \mathcal{R}(x, \textit{LSSS.Reconstruct}(\{w_i\}_{i\in I})),$$

given any set $\{w_i\}_{i\in I} \subset \mathbb{V}_2^{|I|}$ such that the set of indexes I satisfies $|I| \geq t+1$.

The corresponding threshold $\mathcal{NP}$-language follows

$$\mathcal{L}(\mathcal{R}^{\text{LSSS}}) = \{x : \exists \{w_i\}_{i\in I, |I|\geq t+1} \text{ s.t. } (x, \{w_i\}_{i\in I}) \in \mathcal{R}^{\text{LSSS}}\}.$$

Note that LSSS.Reconstruct is running in polynomial time (in the size of the entries), hence $\mathcal{R}^{\text{LSSS}}$ is verifiable in polynomial time by a deterministic algorithm.

Given some witness w for $x \in \mathcal{L}(\mathcal{R})$, we can construct a set of witnesses for $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$ by calling to LSSS.Share(w) and taking any subset of at least $t+1$ shares. More generally, $\mathcal{L}(\mathcal{R}) = \mathcal{L}(\mathcal{R}^{\text{LSSS}})$ as an equality of sets. In the remainder of this chapter, we may use the notation $\mathbf{w} := \{w_i\}_{i\in I} \subset \mathbb{V}_2^{|I|}$ when I is a set of $|I| \geq t+1$ indexes.

6.2.1 TZKP proof system

The purpose of this subsection is to introduce our *threshold zero-knowledge proof* system for threshold relation.

Communication between provers. Our proof system does not impose restrictions on communication among provers, and security holds as long as it occurs via secure channels/authenticated broadcast channels. We tolerate communication for two primary reasons: for general construction purposes, and to achieve a practical protocol that does not rely on physical constraints (*i.e.*, on the isolation of provers). Although the model allows corrupted provers to communicate throughout the proof, our black-box construction in Section 6.3 only involves communication among provers before the first interaction with the verifier V. Finally, Section 6.5 discusses variants of our system that do not require communication.

Communication with the verifier. We define a *pass* between the set of provers and the verifier V as a round of communication such that either V sends a message called a *challenge* to each prover (the challenge may be different for each prover), or each prover sends a message to V (either some commitments or a *response* to a previous challenge). We consider the synchronous model, and if a corrupted prover did not send a value after a clock time, it can be discarded

so that the pass between provers and V can be completely realized.

At this juncture, we shall provide additional details on the threshold lower bound achievable by our system. First, it is reasonable to assume that honest provers receive valid witnesses (*i.e.* shares from the same sharing of a valid witness for the corresponding binary relation). To hope for achieving robustness in the active setting, Theorem 1 establishes a lower bound on the required number of provers in the process to correct errors from corrupted provers. Indeed, the number of provers, k, must be at least $t + 1 + 2e$ to correct up to e errors. By aligning this with the maximum number of corrupted provers tolerated by our security model, we derive that k must satisfy $\geq 3t + 1$.

A *threshold interactive proof* system is an interactive protocol between a set of provers and a verifier, described through a set of passes, and formalized as follows.

Definition 17 (Threshold interactive proof)**.** *Let us consider some (t, n)-LSSS with parameters $(t, n) \in \mathbb{N}^2$ satisfying $t < n$, a set of n computationally unbounded algorithms $P_1, \ldots, P_n$ (called provers), and a PPT algorithm V (called verifier). Let us consider the statement $x \in \mathcal{L}(\mathcal{R}^{LSSS})$ such that V holds x and each prover P_i gets (x, w_i), where w_i is called the witness of P_i for the statement. Then, any subset $P := \{P_1, \ldots, P_k\}$ of $k \geq 3t + 1$ provers interacts with V in a* threshold interactive proof *via some $(k + 1)$-party MPC protocol $\Pi_{\mathcal{L}}$ if:*

- *Ignoring communication with V in $\Pi_{\mathcal{L}}$, provers realize between themselves a k-party MPC subprotocol taking place in the general model 10, in active security tolerating the presence of an algorithm Adv (called adversary) who may corrupt up to t provers;*
- *V interacts with P via a set of passes in $\Pi_{\mathcal{L}}$;*
- *At the end of the passes, V either outputs* ACCEPT *or* REJECT *and the proof is denoted as*

$$\langle (P_i(w_i))_{i\in[1,k]}; V\rangle(x).$$

Let $\alpha, \delta : \mathbb{N} \to [0, 1]$ (which exceptionally denotes a set of real numbers). Then, a threshold interactive proof shall satisfy the following security requirements:

- α-robustness: *If at most t provers are corrupted by Adv, $k \geq 3t + 1$, and V is honest, then V accepts the proof with probability at least $1 - \alpha(\lambda)$, i.e.,*

$$\Pr\left[\langle (P_i(w_i))_{i\in[1,k]}; V\rangle(x) \to \textsc{accept} \mid k \geq 3t + 1\right] \geq 1 - \alpha(\lambda).$$

  If $\alpha = 0$, then the proof has perfect robustness.

- δ-soundness*: If there is no subset $I \subseteq P$ of size at least $t + 1$ such that $\mathcal{R}^{LSSS}(x, \boldsymbol{w}) = 1$ for all $|I|$-tuple $\boldsymbol{w}$ (i.e. $x \notin \mathcal{L}(\mathcal{R}^{LSSS})$), and possibly more than t provers are corrupted by Adv, then the interaction of these k provers with V on input x makes V reject except with probability at most $\delta(\lambda)$ for some negligible function $\delta(\cdot)$, i.e.,*

$$\Pr\left[\langle (P_i(w_i))_{i\in[1,k]}; V\rangle(x) \to \textsc{accept} \mid x \notin \mathcal{L}(\mathcal{R}^{LSSS})\right] \leq \delta(\lambda).$$

The resulting proof is termed a *proof of membership*, where at the end of the interactions, V is convinced of the existence of shared witnesses for the statement $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$. Depending on the objective, a stronger notion of proof, namely *proof of knowledge*, may be required, where provers additionally demonstrate knowledge of these witnesses (to authenticate them). As a second remark, if $\mathsf{P}_1, \ldots, \mathsf{P}_n$ were assumed computationally bounded, it would result in the weaker notion of interactive *argument*.

Our threshold interactive proof can achieve an additional property called *zero-knowledge*. It captures the idea that the views of a malicious verifier and corrupted provers, all together, should not reveal any honest provers' secret information (and more generally no information on the witness that has been shared among the provers).

Definition 18 (Threshold zero-knowledge proof (TZKP))**.** *With notation according to Definition 17, a threshold interactive proof $\langle (P_i(w_i))_{i\in[1,k]}; V\rangle(x)$ for an $\mathcal{NP}$-statement $x \in \mathcal{L}(\mathcal{R}^{LSSS})$ is a threshold zero-knowledge proof if it satisfies the additional property:*

- zero-knowledge: *For any adversary Adv controlling the verifier V and at most t provers $\{P_i\}_{i\in I}$ in the real model, there exists in the ideal model a simulator Sim^{Adv} with oracle access to Adv, and a trusted party computing some functionality F_{TZKP}, such that for every auxiliary input z, the distribution of $\textsc{ideal}_{F_{TZKP}, Sim^{Adv(z)}, I}(x, w_1, \ldots, w_k)$ is perfectly/statistically/computationally close to the distribution of $\textsc{real}_{\Pi_{\mathcal{L}}, Adv(z), I}(x, w_1, \ldots, w_k)$.*

F_{TZKP} is a reactive functionality for the ideal model (see subsection 2.4.2), where the trusted party obtains inputs and sends outputs in phases, aiming to simulate the malicious behavior of an adversary that may produce arbitrary computation as well as communication.

If the zero-knowledge property only holds for the genuine verifier V, then the protocol is deemed *honest-verifier* zero-knowledge. In that case, $\mathsf{Sim}^{\mathsf{Adv}}$ is given random challenges instead of a rewindable black-box access to V.

6.3 A Black-Box Construction for TZKP

We propose a framework for constructing a TZKP applicable to any threshold $\mathcal{NP}$-language. Consider a subset of provers $\mathsf{P}_1, \ldots, \mathsf{P}_k$, and let $\mathbf{w} = (w_1, \ldots, w_k)$ be the corresponding string of witnesses for the public $\mathcal{NP}$-statement $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$. Let f be a functionality with binary output defined as:

$$f(x, \mathbf{w}) = \mathcal{R}^{\text{LSSS}}(x, \mathbf{w}) \; . \tag{6.1}$$

The core objective is to design an interactive protocol between the k provers and the verifier, ensuring it constitutes a TZKP for $x \in \mathcal{L}$. Let us denote such a protocol by $\Pi_{\mathcal{L}}$.

To achieve this, we introduce a two-layer MPC structure, which is described in the following subsections. The first layer is a robust MPC protocol Π_f executed among the provers, which models the functionality f. The second layer consists of MPC protocols simulated within the provers' heads, guaranteeing verifiable computation.

6.3.1 First layer: MPC protocol between the provers

Let Π_f denote a k-party MPC protocol that securely implements the functionality f where the inputs are given as $(x, \mathbf{w})$, with $\mathbf{w}$ representing the shares derived from a (t, k)-LSSS. The purpose of Π_f is to compute a sharing of all the intermediate gates of the computation, ultimately leading to a shared result of $f(x, \mathbf{w})$. Note that the degree of the polynomial used in sharing intermediate and final values may exceed t. If the degree is refreshed at each multiplicative gate, the shares maintain their (t, k)-LSSS form. We consider Π_f as a general MPC protocol (see Definition 10), which must satisfy specific security properties: it must ensure perfect, statistical, or computational t-privacy along with perfect or statistical t-robustness in the malicious setting with the presence of an adversary Adv capable of corrupting up to t provers.

Remark 7. *We can not deal with computational t-robustness, unless we computationally bound malicious provers in the soundness property. But it would lead to zero-knowledge arguments. However, since the verifier is assumed to be PPT, it makes sense to consider computational t-privacy.*

Assume that Π_f consists of r rounds, denoted $\Pi_f = (\Pi_f^1, \ldots, \Pi_f^r, \text{output}_f)$. By the conclusion of these r rounds, the arithmetic circuit representing f is conjointly computed. During the final stage output_f, each prover P_i broadcasts the result of their last local computation, denoted y_i. The provers then execute LSSS.Reconstruct associated with the (t', k)-LSSS of y_i where $t' \geq t$ (it may include a public decoding algorithm for the underlying codeword). According to Theorem 1, the total number of provers k must satisfy $k \geq t' + 1 + 2t$ for the reconstruction algorithm to tolerate up to t errors among $(y_1, \ldots, y_k)$. If the reconstructed value $\mathsf{LSSS.Reconstruct}(y_1, \ldots, y_k)$ matches the public value $f(x, \mathbf{w})$, the protocol Π_f outputs 1; otherwise, it outputs 0.

It is important to note that revealing y_i during output_f might expose some private information from P_i's input, a concern which will be addressed in the following subsection.

6.3.2 Second layer: MPC protocols in the head of provers

The second MPC layer is designed to ensure the verifiability of the computation carried out by Π_f. Recall that we denote by $\Pi_{\mathcal{L}}$ the interactive protocol between a subset of k provers $\mathsf{P}_1, \ldots, \mathsf{P}_k$ and a verifier V. Consider a LSSS that produces $N \geq 3$ shares, and for simplicity, let us use the modular additive secret sharing scheme, denoted as $\langle\!|\cdot|\!\rangle$. Each prover in the protocol simulates N virtual parties, leading to a total of Nk virtual parties participating in the execution of $\Pi_{\mathcal{L}}$. Although the modular additive scheme is chosen here for simplicity, other LSSS constructions, such as Shamir's secret sharing scheme, are also viable and can be applied in this context by leveraging techniques from [FR23b, FR23a].

Communication between provers. Once we have specified the type of sharing emulated among the provers, we can be more precise about how they communicate. For the sake of consistency and verifiability, the system assumes that the provers send messages to each other in the form of such sharing $\langle\!|\cdot|\!\rangle$, so that the share of the i-th party in the head of the sender is addressed to the i-th party in the head of receivers.

Computation in the head. The inputs of P_i in Π_f^r are: the public $\mathcal{NP}$-statement x, the private input w_i, the random tape, the sharing sent/received/broadcast during the previous rounds. During output_f, it additionally takes as input challenges received from V in $\Pi_{\mathcal{L}}$. Given those inputs, each prover emulates MPC protocols with N parties in "their head" in the passive setting and the broadcasting model.

For a prover P_i, the parties in their head perform three types of actions during each round (on the inputs that we have mentioned):

- Receiving randomness: The parties receive the same random value(s). This simulates the challenges sent by V to P_i in $\Pi_{\mathcal{L}}$.

- Receiving hint: The parties get sharing or public values. This simulates both new sharing generated by P_i (*e.g.*, for multiplication checking techniques such as [BN20b]) and sharing/broadcast received by P_i in Π_f.
- Computation in the broadcasting model: The parties locally compute a linear function with some sharings as input (these sharings are assumed to be linear), where the function depends on challenges and previous broadcast values. Then they may broadcast their resulting shares and recover the shared evaluated function.

Let us denote by π_j the N-party MPC protocol simultaneously emulated in the head of each prover during Π_f^j (to prove the computation of the j-th round), and π_f the N-party MPC protocol emulated during output_f. Essentially, π_f realizes the traditional MPCitH part by receiving V's challenges (for proving multiplicative relations and opening parties' view). From the security point of view, these simulated protocols have to achieve perfect/statistic/computational $(N-1)$-privacy and perfect/statistic correctness in the passive setting.

Recall that y_i is the last reconstructed value in the head of P_i, and this value is broadcast during output_f as mentioned in the previous subsection. This may break the prover's privacy. For this purpose, we add a local refreshing step (in output_f), where each prover P_i does:

1. Generates a random r_i and a random sharing of this r_i;
2. Generates a random public sharing of 0;
3. Parties in the head compute and set $[\![y_i]\!] \leftarrow [\![y_i]\!] + [\![r_i]\!]([\![0]\!])$.

Views and consistency. We define a notion of consistency for views, taking into account the communication among provers and capturing the broadcasting between parties. For a party $\mathsf{P}_{i,\ell}$ ($i \in [1,k], \ell \in [1,N]$), one writes $V_{i,\ell}$ to denote its *view*.

Definition 19 (Consistent views)**.** *Let Π_f be a k-party MPC as defined in subsection 6.3.1, with corresponding N-party protocols $\pi_1, \ldots, \pi_f$ from subsection 6.3.2. Let x be the public input, $\mathcal{A} \subseteq \{P_1, \ldots, P_k\}$ be a subset of users, and $I \subset [N]$ be a subset of indexes. Then we say that views $\{V_{i,j}\}_{i:P_i \in \mathcal{A}, j \in I}$ are consistent if the following consistency checks are fulfilled:*

- *For every prover $P_i \in \mathcal{A}$ and every index $\ell \in I$, the broadcasts sent by $V_{i,\ell}$ have to be consistent (with respect to x and $\pi_1, \ldots, \pi_f$), i.e. the outgoing values implicit in $V_{i,\ell}$ are identical to the incoming values reported in $\{V_{i,\ell'}\}_{\ell' \in I \setminus \{\ell\}}$.*
- *For every prover $P_i \in \mathcal{A}$ and every index $\ell \in I$, the shares/broadcasts sent by $V_{i,\ell}$ have to be consistent (with respect to x and Π_f), i.e. the outgoing shares/broadcasts implicit in $V_{i,\ell}$ are identical to the incoming shares/broadcasts reported in $\{V_{j,\ell}\}_{j:P_j \in \mathcal{A} \setminus \{P_i\}}$.*

Therefore, the views $\{V_{i,j}\}_{i:\mathsf{P}_i \in \mathcal{A}, j \in I}$ are consistent with respect to protocols Π_f, $\pi_1, \ldots, \pi_f$ and to x if and only if there exists an honest execution of $\Pi_{\mathcal{L}}$ by $\{\mathsf{P}_i\}_{i:\mathsf{P}_i \in \mathcal{A}}$ with public input x. The access structure of our construction is detailed in Appendix E.

6.3.3 A TZKP protocol

We present our TZKP protocol (Protocol 6.2) in a five-round framework, although it can be generalized to a $2\gamma+1$-pass interaction between $\mathsf{P}_1, \ldots, \mathsf{P}_k$ and V. This generalized version includes $\gamma - 1$ iterations of round 2 and 3, each involving additional challenge-response phases, where γ represents the number of challenge phases from V. In this setting, the final challenge corresponds to the decision on which views to reveal, while the preceding $\gamma - 1$ challenges are necessary to prove the multiplicative relations. Additionally, although not explicitly stated, Protocol 6.2 is presented within the commitment-hybrid model framework (see subsection 2.4.2).

We detail the reactive functionality F_{TZKP} for the ideal model in Figure 6.1, where the trusted party obtains inputs and sends outputs in phases. It aims to simulate the malicious behavior of an adversary that may produce arbitrary computation as well as communication (see subsection 2.4.2).

Theorem 18 (Security proofs)**.** *Let x be an $\mathcal{NP}$-statement for some relation $\mathcal{R}$, let $(t,n) \in \mathbb{N}^2$ be such that $t < n$, and let us fix some (t,n)-LSSS. Define the k-ary functionality f computing the threshold relation $\mathcal{R}^{LSSS}$ as in Equation 6.1 with $k \geq 3t+1$, and the corresponding k-party MPC protocol Π_f computing f in active security with the presence of an adversary Adv corrupting up to t users. If Π_f realizes f with perfect/statistic t-robustness, and perfect/statistic/computational t-privacy for the F_{TZKP} functionality 6.1 in the commitment-hybrid model. Then Protocol 6.2 is a TZKP for $x \in \mathcal{L}(\mathcal{R}^{LSSS})$, with soundness error $\epsilon \leq 1/N$ where N is the number of parties emulated in the head of each prover.*

Proof. We prove Protocol 6.2, whcih we call $\Pi_{\mathcal{L}}$, meets Definitions 17 and 18 of a TZKP.

Robustness: Let us begin by assuming that Π_f is perfectly t-robust (according to Definition 13), we show $\Pi_{\mathcal{L}}$ achieves perfect robustness (according to Definition 17). Let $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$ with witnesses $w_1, \ldots, w_k$, and assume that at most t provers are controlled by Adv. Each honest prover P_i generates a valid sharing of w_i in their head. Since Π_f is perfectly t-robust, Adv can not prevent honest provers from producing an honest execution of the protocol Π_f and getting valid

Functionality F_{TZKP}
Input: A set of indices $I \subset [1, k]$ denoting corrupted provers.

1. For each round $v \in [1, r]$:

   (i) F_{TZKP} receives inputs of each honest prover and computes their output of π_v.

   (ii) F_{TZKP} computes and sends to the ideal adversary the communication in Π_f^v from honest provers to corrupted ones. F_{TZKP} also computes the communication in Π_f^v between honest provers.

   (iii) F_{TZKP} receives communication from the ideal adversary as the communication from corrupted provers to honest ones in Π_f^v. If a sharing is not received or unfit (not of the right format), then F_{TZKP} replaces it with a sharing of 0.

2. F_{TZKP} computes OUTPUT_f for each honest prover and sends these values to the ideal adversary.

3. F_{TZKP} receives the outputs of corrupted provers from the ideal adversary.

Fig. 6.1: Functionality F_{TZKP}

outputs. Hence, there exists a subset of $t + 1$ honest provers for whom all the parties' views inside this subset are consistent following the Definition 19. Moreover, there are at most t errors in the broadcast output codeword $(y_1, \ldots, y_k)$, hence a public decoding algorithm can correct them. Therefore, an honest verifier will always accept the proof. If Π_f is statistically t-robust, then the verifier accepts the proof except with probability the error probability of Π_f, hence $\Pi_{\mathcal{L}}$ achieves statistical robustness.

Soundness: First assume that Π_f is perfectly t-robust, and that $\mathcal{R}^{\text{LSSS}}(x, w_1, \ldots, w_k) = 0$ for all k-tuple $(w_1, \ldots, w_k)$. By the perfect t-robustness of Π_f, and for all choices of sharing $\{\langle\!|w_i|\!\rangle\}_{i\in[1,k]}$, and all choices of randomness, the output of an honest execution of Π_f must be 0. Either all the provers inside $\mathcal{A}$ (the subset chosen by V during the verification step of Protocol 6.2) honestly perform the protocol, and their output are consistent with their input. But then the broadcast outputs form a tuple $(y_1, \ldots, y_k)$ such that V cannot decode it into a word that would result in the proof being accepted. Or at least one prover cheats inside $\mathcal{A}$, and there exists at least one inconsistent view that is detected by V with probability $1 - 1/N$.

Zero-knowledge: Let us begin by assuming that Π_f is perfectly t-private in active security (according to Definition 14), and that $\pi_1, \ldots, \pi_r, \pi_f$ are perfectly $(N - 1)$-private in passive security (according to Definition 12). Let Adv be an adversary in the real world controlling V and $\{\mathsf{P}_i\}_{i\in I}$ where $I \subset [1, k]$ is a subset of size at most t. We begin by describing the simulator $\mathsf{Sim}^{\mathsf{Adv}}$, the adversary in the ideal model. It interacts externally with the trusted party computing the functionality F_{TZKP} 6.1, and internally invokes the adversary Adv (via an oracle access), hence simulating an execution of Protocol 6.2 for Adv. $\mathsf{Sim}^{\mathsf{Adv}}$ takes as inputs: auxiliary input $z \in \{0, 1\}^*$, x, and $\{w_i\}_{i\in I}$. $\mathsf{Sim}^{\mathsf{Adv}}$ works as follows:

1. $\mathsf{Sim}^{\mathsf{Adv}}$ internally invokes Adv with the auxiliary input z.

2. For each round $j \in [1, r]$:

   (i) External interaction with Functionality F_{TZKP} (Step (i)): After the honest provers send their inputs to the trusted party, the simulator $\mathsf{Sim}^{\mathsf{Adv}}$ receives communication.

   (ii) $\mathsf{Sim}^{\mathsf{Adv}}$ simulates Step (ii) of F_{TZKP} and hands the adversary Adv the communication from the honest provers.

   (iii) $\mathsf{Sim}^{\mathsf{Adv}}$ simulates Step (iii) of F_{TZKP} and receives from Adv the communication from corrupted provers.

3. $\mathsf{Sim}^{\mathsf{Adv}}$ internally calls V for challenges.

4. External interaction with Functionality F_{TZKP} (Step 2): The simulator $\mathsf{Sim}^{\mathsf{Adv}}$ receives the outputs of the honest provers.

5. $\mathsf{Sim}^{\mathsf{Adv}}$ internally invokes Adv with the honest provers' outputs and receives the outputs of corrupted provers (Step 3 of F_{TZKP}).

□

The output of the MPC protocol Π_f usually follows a probability distribution with some negligible non-zero false positive probability (*i.e.* Π_f usually realizes the functionality f with statistical robustness typically when $\Pi_{\mathcal{L}}$ has more than 3-pass). Hence, we discuss on Theorem 18 when Π_f is statistically t-robust. The soundness of the Protocol 6.2 holds

TZKP protocol $\Pi_{\mathcal{L}}$ for $x \in \mathcal{L}$
$k+1$ users: k provers $\mathsf{P}_1, \ldots, \mathsf{P}_k$ and one verifier V
A public threshold $\mathcal{NP}$-statement $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$ with some LSSS from Definition 9, and a functionality f from Equation 6.1.
Input: Each prover P_i holds some witness w_i for $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$.

Round 1: (i) Each prover P_i generates a random sharing $(\!|w_i|\!)$ among N virtual parties $\{\mathsf{P}_{i,j}\}_{j\in[1,N]}$.

(ii) Provers conjointly compute Π_f, *i.e.* for each round $j \in [1, r]$, P_i simulates the computation of π_j by the N parties, and then provers send sharing/broadcast to each other.

(iii) Each prover computes their own commitment to the views of their N parties and sends it to V (*i.e.* one commitment per prover).

Round 2: V sends a random challenge ch_1^i to each P_i for proving multiplications (typically via a batching equation of all the multiplications to prove).

Round 3: Each prover P_i simulates the computation of π_f by the N parties, and commits to each of these views (one commitment per prover).

Round 4: V picks an opening views challenge $ch_2 = \ell^* \xleftarrow{\$} [1, N]$ and sends it to each prover.

Round 5: Each P_i opens the commitments corresponding to all the parties except to P_{i,ℓ^*}, and sends broadcasts of P_{i,ℓ^*}.

Verification: V accepts if and only if there exists a subset $\mathcal{A}$ of at least $t+1$ provers such that:

(i) the provers in $\mathcal{A}$ successfully opened the requested views;

(ii) the reconstructed output of Π_f is 1: given $(y_1, \ldots, y_k) \leftarrow \mathsf{output}_f$, $\mathsf{LSSS.Reconstruct}(\{y_i\}_{i\in[1,k]}) = 1$;

(iii) the opened views in $\mathcal{A}$ are consistent according to Definition 19.

Fig. 6.2: TZKP protocol

only if Π_f is perfectly t-robust. Indeed, when the protocol Π_f is only statistically robust, an adversary may leverage the false positive probability by getting some "good" random coins on which the protocol accepts a wrong statement by incorrectly outputting 1, making V accept whatever the last challenge of V. To deal with statistical robustness, we follow [IKOS09] by generating a coin tossing. Each prover commits to the randomness of every party. V then sends some randomness for each party. Each prover proceeds to execute the protocol Π_f using the sum of both randomness as the randomness of the parties. In the decommitment phase, each prover opens its commitment to the randomness of parties whose views have to be opened. The verifier finally checks if the randomness used by those parties has been correctly computed. The soundness error is now bounded by $\epsilon \leq 1/N + \delta(\lambda)$ for some negligible function $\delta(\cdot)$.

6.4 A Construction Based on VSS-BGW

This section does not introduce any new concepts but instead applies established protocols to our specific context. For the remainder of this section, consider the Shamir secret sharing of degree t, referred to as $[\![\cdot]\!]$, with $t < k \leq n$, for computation among the provers. The secret sharing scheme in the head of the provers is assumed to be the additive modular one (denoted as $(\!|\cdot|\!)$), ensuring uniformity across all provers. Let $\mathcal{L}$ be a threshold $\mathcal{NP}$-language with the corresponding k-ary functionality f such that

$$f(x, \mathbf{w}) = \mathcal{R}^{\text{LSSS}}(x, \mathbf{w}).$$

The heart of the matter consists of deriving a k-party MPC protocol Π_f that securely realizes f and which is made verifiable, leading to a TZKP for the statement. Consider the $\Pi_{\mathcal{L}}$ Protocol 6.2, where provers start by conjointly computing an (t, n)-LSSS of each gate's output of the arithmetic circuit of f. Therefore, the essence lies in showing how to deal with arithmetic gates.

Additive gates. Any linear operation within Π_f is effortlessly handled due to the linearity of sharing. When the provers possess any two sharing $[\![a]\!]$ and $[\![b]\!]$, they can locally compute a sharing $[\![a+b]\!]$ of $a+b$ following $\Pi_{\mathcal{L}}$. Indeed, parties in the head of P_i conjointly hold $(\!|[\![a]\!]_i|\!)$ and $(\!|[\![b]\!]_i|\!)$, and proceed as follows: they locally compute a sharing $(\!|[\![a]\!]_i|\!) + (\!|[\![b]\!]_i|\!) = (\!|[\![a]\!]_i + [\![b]\!]_i|\!) = (\!|[\![a+b]\!]_i|\!)$ of $[\![a+b]\!]_i$.

Multiplicative gates. For each multiplicative gate in the circuit, additional steps are necessary. Given $[\![a]\!]$ and $[\![b]\!]$, the provers aim to securely compute $[\![ab]\!]$ following $\Pi_{\mathcal{L}}$. Multiplying two Shamir secret shares of degree t yields a Shamir secret share of degree $2t$, imposing additional constraints on both the number of provers and the size of the base field. A common strategy to handle this threshold increment is to use the BGW degree refreshment protocol [BGW88] (which also includes its verifiable secret sharing variant for active setting).

6.4.1 BGW protocol and its robustness

We recall the well-known BGW protocol. For any couple of sharing $([\![a]\!], [\![b]\!])$ held by the provers, the latter can locally compute a sharing $[\![ab]\!]^{2t}$ (to denote the $2t$-degree) via $[\![ab]\!]^{2t}_i = [\![a]\!]_i[\![b]\!]_i$ for each prover P_i. The goal is to conjointly compute $[\![ab]\!]$ (*i.e.* a sharing of degree t). This *refreshing* step consists in re-sharing each share of $[\![ab]\!]^{2t}$ with a new random polynomial of degree t. Concretely, provers hold $([\![a]\!], [\![b]\!])$ such that each P_i owns the shares $(A(i), B(i))$ with $(A(x), B(x))$ polynomials of degree t with $(A(0), B(0)) = (a, b)$, and does the following:

1. Each P_i locally computes the product $A(i)B(i)$;
2. Each P_i randomly generates a degree t polynomial $H_i(X)$ such that $H_i(0) = A(i)B(i)$, and shares it by sending $H_i(j)$ to P_j for every $j \neq i$;
3. Given some public Vandermonde linear system, the first row of its inverse matrix is usually named the *recombination vector*; one names it μ. Then each P_i locally computes $H(i)$, where $H(x) := \sum_{i=j}^{k} \mu_j H_j(x)$ is of degree t and satisfies $H(0) = ab$.

This protocol can be readily adapted to $\Pi_{\mathcal{L}}$ with simulated parties in the head of each prover. The communication during step 2 can be managed as detailed in Section 6.3. However, this method suffers from a significant drawback: if a single corrupted prover provides incorrect shares to honest provers during step 2, the computation of $H(x)$ may be compromised, rendering none of the shares $H(1), \ldots, H(k)$ valid (thus, there is no possibility of error correction). A solution to this issue arises from a stronger notion of secret sharing schemes, namely *verifiable* secret sharing scheme.

6.4.2 Verifiable secret sharing scheme

In active setting, we must pay particular attention to communications during which a potentially corrupted user $\mathsf{D} \in \mathsf{P}$ (called the *dealer*) may share inconsistent values with the other users. Hence, the latter may wish to proceed to some verification on the sharing, particularly whether the shares have been generated from the same polynomial of degree t. This sake of robustness imposes a stronger notion of secret sharing schemes, namely the *verifiable secret sharing scheme* (VSS). We only focus on perfectly-secure VSS.

Like any secret sharing scheme, a VSS comprises both a sharing phase and a reconstruction phase. However, during the sharing phase, D distributes its secret in a verifiable manner. Over the past few decades, numerous verifiable secret sharing schemes have been proposed. Asharov and Lindell [AL17] proved that, when the number of users k satisfies $k \geq 3t+1$, the protocol from [GIKR01] is perfectly t-private and perfectly t-robust in active security. This VSS constructs shares for a value $s \in \mathbb{F}$ as follows:

1. D randomly chooses a bivariate polynomial of degree t in both variables $S(x, y) \in \mathbb{F}[x, y]$ with $S(x, 0) = F(x)$ such that $F(0) = s$. Then D sends $\{F(\alpha_i), G_i(y) := S(\alpha_i, y), F_i(x) := S(x, \alpha_i)\}$ to P_i for each i.
2. Each P_i checks that $F_i(0) = F(\alpha_i)$, and conjointly with each P_j, checks that $F_i(\alpha_j) = G_j(\alpha_i)$ and $F_j(\alpha_i) = G_i(\alpha_j)$.

The security proof of this sharing relies on the following lemma. See [AL17] for further details.

Lemma 2. *Let $I \subseteq [1, k]$ be a set of indices with $|I| \geq t + 1$. Consider a set of pairs $\{F_\ell(y), G_\ell(x)\}_{\ell \in I}$ of degree-t polynomials over $\mathbb{F}$, and let $\{\alpha_\ell\}_{\ell \in I}$ be distinct non-zero elements in $\mathbb{F}$. If for every $(i, j) \in I^2$, it holds that $F_i(\alpha_j) = G_j(\alpha_i)$, then there exists a unique bivariate polynomial $S(x, y) \in \mathbb{F}[x, y]$ of degree t in both variables such that $F_\ell(y) = S(\alpha_\ell, y)$ and $G_\ell(x) = S(x, \alpha_\ell)$ for every $\ell \in I$.*

A *complaint* broadcast by P_i is a set $(i, j, F_i(\alpha_j), G_i(\alpha_j))$ sent by P_i for some $j \in [1, k]$, claiming that either $F_i(\alpha_j) \neq G_j(\alpha_i)$ or $G_i(\alpha_j) \neq F_j(\alpha_i)$. In other words, P_i asserts that P_j does not send a valid value or that their common shares are not consistent. We say that a complaint is *conjoint* if both P_i and P_j broadcast a complaint for the shares of the other one. In a nutshell, the consistency checking procedure is based on conjoint complaints. If an honest prover receives a non-consistent share, by Lemma 2, they raise a joint complaint with at least one other honest prover. Then, D has to manage this joint complaint with some public broadcast value. Otherwise, if the joint complaint was not solved, all the honest provers would reject the sharing. When $t + 1$ provers reject the sharing, the dealer is fired. Note that an honest dealer can not be fired, since this dealer will always solve a joint complaint, and there are at most t corrupted provers. The original protocol is more involved than this discussion since it shall capture all the different situations. We

refer to [CCP22, AL17] for further details. But as a result, if a corrupted dealer attempts to cheat during the sharing process, D can only cheat on the constant term of the polynomial, *i.e.*, the secret. Naturally, we rely on the Reed-Solomon decoding algorithm for correcting those potential cheats, and whose adaption to our black-box construction 6.2 is relatively straightforward at this stage.

6.4.3 Multiplicative gate protocol

Given a pair of t-degree polynomials $(A(x), B(x))$, the polynomial $A(x)B(x)$ defines a code of dimension $2t+1$, and Theorem 1 provides the lower bound $k \geq 4t+1$ on the length of the code to correct up to t errors. When $k = 3t+1$, the underlying MDS code has parameters $[3t+1, 2t+1]$, and thus can only correct up to $t/2$ errors, which is a barrier for t-robustness. However, there exists a set of polynomials $\{D_1, \ldots, D_t\}$ of degree at most t such that

$$C(x) = A(x)B(x) - \sum_{l=1}^{t} x^l D_l(x)$$

is a polynomial of degree t that falls to $A(0)B(0)$ in 0. Hence, if $k \geq 3t+1$, sharing the degree t polynomials $D_1, \ldots, D_t$ with some VSS can be done with t-robust security in our model, and so is the computation of $C(x)$. See [AL17] for further details. To introduce a simpler protocol, we focus on the $k \geq 4t+1$ case.

Recall that in BGW protocol, each user P_i shares $A(i)B(i)$ with a VSS. At the end of this sub-sharing phase, each P_i holds the set of shares $\{F_1(i), \ldots, F_k(i)\}$ such that $F_j(x)$ is a degree t polynomial for each $j \in [k]$. Moreover, in the presence of an adversary corrupting up to t users, $(F_1(0), \ldots, F_k(0))$ is a word at distance at most t from $(A(1)B(1), \ldots, A(k)B(k))$. Hence, we have to clarify how users can conjointly compute the syndrome of $(F_1(0), \ldots, F_k(0))$. This is detailed in Protocol 6.3. At the end of this protocol, they get in the clear $(F_1(0), \ldots, F_k(0))H^T = eH^T$ where H is the parity-check matrix of the code, and e the error vector for the word $(F_1(0), \ldots, F_k(0))$. Assuming that we have access to a Reed-Solomon decoding algorithm in the clear, users can locally deduce e and correct the potential wrong shares.

Input: Let $k \geq 4t+1$. For each $\mathsf{P}_i \in \{\mathsf{P}_1, \ldots, \mathsf{P}_k\}$, parties hold $\llparenthesis(F_1(i), \ldots, F_k(i))\rrparenthesis$, with $F_1, \ldots, F_k$ t-degree polynomials, and $\llparenthesis\cdot\rrparenthesis$ the LSSS used by virtual parties.
Output: $(F_1(0), \ldots, F_k(0))H^T$ in the clear.

1. For each P_i, parties compute $\llparenthesis(F_1(i), \ldots, F_k(i))\rrparenthesis H^T := \llparenthesis(Z_1(i), \ldots, Z_k(i))\rrparenthesis^T$ and broadcast it;
2. For each set $\{Z_j(1), \ldots, Z_j(k)\}$ (for $j \in [1,k]$), at most t values are wrong, and this can be corrected by applying the Reed-Solomon decoding algorithm locally.

Fig. 6.3: Conjoint syndrome computation with our black-box construction 6.3

When $k \geq 4t+1$, [AL17] proved that the original version of Protocol 6.3 (*i.e.* without virtual parties) is perfectly t-private and perfectly t-robust for some functionality in some VSS-hybrid model, in the active setting. Moreover, they also proved that the original version of Protocol 6.4 is perfectly t-private and perfectly t-robust.

Theorem 19. *Let $x \in \mathcal{L}$ be a threshold $\mathcal{NP}$-statement, and $k \geq 4t+1$. Define the k-ary functionality f computing $\mathcal{R}$ as in Equation 6.1, and consider a k-party MPC protocol Π_f computing f in the active setting with the presence of an adversary Adv corrupting up to t provers. If Π_f is realized with the above additive and multiplicative Protocol 6.4, then the Protocol 6.2 is a TZKP for $x \in \mathcal{L}$, with soundness error $\epsilon \leq 1/N$, where N is the number of parties emulated in the head of each prover.*

Proof. We prove that our construction is a TZKP in the BGW $-$ VSS-hybrid model. As mentioned previously, Asharov and Lindell [AL17] proved the perfect t-privacy and perfect t-robustness of the BGW $-$ VSS multiplicative protocol (Protocol 6.4). Then our construction for Π_f is immediately a modular composition of these subprotocols. We conclude with Theorem 18. □

6.5 Low-Depth Arithmetic Circuits and Applications

Establishing communication in an insecure environment would require secure/authenticated channels. This could become a barrier in a post-quantum regime, all the more so if we add the desire for security assumptions relying on the hardness of the chosen $\mathcal{NP}$-problem. While efficient post-quantum lattice-based key exchange protocols exist [BCD+16, ADPS16, BDK+18], they significantly differ from the traditional Diffie-Hellman key exchange in that they require additional rounds of interaction. For many applications, the non-interactive property is essential, making their transition to post-quantum significantly more complex. All of this is pushing our motivation for removing communication among provers and developing an isolated variant of our TZKP Protocol 6.2. The first attempt is as follows.

Protocol for multiplicative gates with our TZKP black-box construction
Fix an LSSS for computation in the head of provers, denoted as $(\!|\cdot|\!)$. The (t,k)-LSSS for computation between provers is denoted as $[\![\cdot]\!]$.
Input: For each P_i, parties hold $(\!|A(i)|\!) = (\!|[\![a]\!]_i|\!), (\!|B(i)|\!) = (\!|[\![b]\!]_i|\!)$ with $A(x), B(x)$ degree-t polynomials such that $A(0) = a, B(0) = b$.
Output: For each P_i, parties get $(\!|[\![ab]\!]_i|\!)$.

1. Each prover P_i emulates in their head a sharing $(\!|A(i)B(i)|\!)$, where $A(i)B(i) = [\![ab]\!]^{2t}$.
2. Each prover P_i plays the dealer and calls the VSS functionality with input $(\!|A(i)B(i)|\!)$. When all provers have played the dealer, parties in P_i hold $\{(\!|\tilde{F}_j(i)|\!)\}_{j\in[1,k]}$, where $\tilde{F}_j(x)$ is a degree-t polynomial. However, no guarantee is given on $\tilde{F}_j(0) \stackrel{?}{=} A(j)B(j)$ (for $j \in [1,k]$).
3. Provers conjointly compute the syndrome $(\tilde{F}_1(0), \ldots, \tilde{F}_k(0))H^T$ via Protocol 6.3 and locally call in their head to the decoding algorithm. They get the error vector $e \in \mathbb{F}^k$ such that $(\tilde{F}_1(0), \ldots, \tilde{F}_k(0)) - e^T$ is a codeword.
4. Each P_i uniformly at random generates a sharing $(\!|e|\!)$ so that parties can compute $(\!|(\tilde{F}_1(i), \ldots, \tilde{F}_k(i))|\!) - (\!|e^T|\!) = (\!|(F_1(i), \ldots, F_k(i))|\!)$.
5. For each P_i, parties compute $\sum_j \lambda_i (\!|F_j(i)|\!)$ where $\sum_j \lambda_i F_j(X)$ is a degree-t polynomial with constant term ab, and λ is a public recombination vector.

Fig. 6.4: Protocol for multiplicative gates with our black-box construction

Pre-computation techniques. It's tempting to pre-compute the circuit realizing the function underlying the chosen $\mathcal{NP}$-problem, and enlarge the secret input information of each prover by providing them the set of sharing of each gate's output of the circuit. This common technique has already been considered for high-degree arithmetic circuit in the MPCitH paradigm, *e.g.* for building efficient signatures based on the hardness of computing a preimage of the AES [BdSGK+21, BBM+24]. Imagine a scenario where each prover holds, at the beginning of the proof, a (t,n)-LSSS share for the output of each gate in the circuit. These shares would have been pre-computed, raising the possibility of a protocol that operates without the need for communication. However, in addition to proving that a gate has been correctly evaluated, provers must ensure that the degree-refreshing computation is verifiable, necessitating a conjoint effort. Hence, a pre-computation method has no chance to succeed without communication among provers, and thus loses all its interest.

Low-depth circuits. As we have just seen, a priori there is no general technique for getting rid of communications given a generic $\mathcal{NP}$-problem. Instead, we should focus on a subset of these problems. Define the multiplicative depth of a circuit as the maximum number of multiplications along any path through the circuit. We focus our efforts on circuits with a low multiplicative depth to obtain an isolated variant of the TZKP Protocol 6.2. At the cost of degrading the threshold, one could opt not to refresh any gate's ouptut sharing. For a circuit C with multiplicative depth $\alpha \in \mathbb{N}$, public input x, and secret input w shared with a (t,n)-LSSS, after locally computing the circuit (*i.e.* $C(x, [\![w]\!]_i)$), each prover P_i gets a share $[\![C(x,w)]\!]_i$ from some $(\alpha t, n)$-LSSS. Consequently, the new lower bound on the number of provers realizing the proof has to be $k \geq \alpha t + 1 + 2t$ (from Theorem 1) to align with our security model. Finally, it's reasonable to consider that this degradation is manageable for small values of α. We take this opportunity to remind that even if one gets rid of communication, the security model of our TZKP still assumes that malicious provers may signal with each other (one does not introduce any provers' isolation assumption). We also note that the obtained protocol is still t-private.

6.5.1 Achieved complexity

Let $x \in \mathcal{L}(\mathcal{R}^{\text{LSSS}})$ be a threshold $\mathcal{NP}$-statement, α the multiplicative depth of the underlying arithmetic circuit computing the relation, and $k \geq (\alpha+2)t+1$ the number of provers realizing the proof. Additionally, choose any MPCitH-like proof system for $x \in \mathcal{L}(\mathcal{R})$. Let σ_i be such a proof realizing by P_i in $\Pi_{\mathcal{L}}$. Then the overall TZKP $\Pi_{\mathcal{L}}$ Protocol 6.2 has the following complexities:

- Communication complexity to V (*i.e.* the size of the proof) is linear in the number of provers: $|\Pi_{\mathcal{L}}| = \sum_{i=1}^{k} |\sigma_i|$.
- Computation complexity of P_i in $\Pi_{\mathcal{L}}$ is the computation complexity of the prover in σ_i.
- Computation complexity of V in $\Pi_{\mathcal{L}}$ is k times the computation complexity of the verifier in any σ_i.

6.5.2 Turning TZKP into a non-interactive proofs system

Consider a TZKP with public coin, *i.e.* all of V's random choices are made public. Additionally, assume that the underlying circuit (for the relation $\mathcal{R}$) has a low multiplicative depth, so that we use the isolated variant of Protocol 6.2.

The 3-pass version of Protocol 6.2, where the sole challenge of V is for the opening views, can be directly transformed into a non-interactive scheme via the Fiat-Shamir heuristic [FS87]. Note that, since we consider the isolated variant, the challenge for the opening views does not need to be the same for each prover. Consider the random oracle RO called by the provers in the non-interactive setting. Assume the codomain of RO to be super-polynomial in the security parameter λ, and the number of provers to be $\mathsf{poly}(\lambda)$. Then, a strategy for the adversary in the non-interactive setting is to simulate additional random oracle outputs, until finding a collision with V's challenge space. If this space is $\mathsf{poly}(\lambda)$, then even after $\mathsf{poly}(\lambda)$ trials, none of the provers corrupted by the adversary would have found a collision, except with negligible probability.

6.5.3 Applications

Although this work is primarily theoretical, some applications are worth mentioning.

Threshold signatures.

Several definitions of threshold signatures exist in the literature, and it is important to distinguish between two types based on the private inputs of the signers. In a *threshold multi-signature*, each signer possesses their own independent secret key, similar to a multi-signature scheme, but a threshold number of signers is required to generate the signature. In contrast, the typical definition of a *threshold signature* assumes that the signing key is distributed among the signers. Therefore, the latter aligns precisely with our model. On top of that, achieving robustness in active security is a must for this kind of application.

Putting aside constructions based on symmetric primitives, most of the so-called post-quantum digital signature schemes based on MPCitH rely on problems with very low-depth arithmetic circuits. Therefore, it is tempting to apply our construction to these problems to directly obtain threshold signatures.

We have seen that the proof size of $\Pi_{\mathcal{L}}$ equals k times the proof size of the chosen MPCitH-like proof (where k is the number of provers required for producing a robust proof). Given the recent advancements in the MPCitH framework, which aim to achieve appealing performance when building proofs based on this paradigm, and despite the linearity boundary for the proof size in terms of the number of provers, for reasonable values of k, it provides additional motivations for our TZKP proof system. To get an idea of the sizes we could achieve, let us look at the performances in MPCitH. Some of the most efficient post-quantum digital signatures based on MPCitH are relying on the difficulty of the rank syndrome decoding, MinRank problems, and (non-structured) instances of MQ problems (solving multivariate systems of quadratic equations). Indeed, [FR23a, BFG$^+$24] built signatures with sizes below 3.5-4 kB for 128 bits of security (depending on the number of parties emulated by the provers) based on the hardness of these three problems. Recently, Baum et al. [BBM$^+$24] optimized the cost of GGM trees [GGM84] bringing these signature sizes around 3 kB (for 128 bits of security). Moreover, computational complexity of the prover and the verifier have also been improved recently [MGH$^+$23, FR23a].

Threshold verifiable decryption

An unforgeable threshold verifiable decryption scheme with robust security requires that any sufficiently large subset of provers holding shares of the decryption key can efficiently generate their part of the decryption algorithm (and thus their share of the plaintext) in an adversarial environment. Additionally, each user with access to the public key must be able to efficiently verify the decryption, and no group of provers without the full decryption key should be able to collectively produce a valid plaintext.

Since TZKP inherently assumes that the secret input is shared among the provers, our black-box construction from Section 6.3 can be the foundation for building a threshold verifiable decryption scheme. Obviously, given any decryption circuit, hoping its multiplicative depth is low, we can get a threshold verifiable decryption scheme by plugging this circuit as Π_f into Protocol 6.2.

As an application, we take a look at the CKKS homomorphic encryption scheme [CKKS17] based on the Ring-LWE problem [LPR10]. Let us give a bird's-eye view of the original scheme. Let N be a power of two, and let $R = \mathbb{Z}[x]/\langle x^N + 1\rangle$ be a cyclotomic polynomial ring. We denote $M = 2N$ and by $\mathbb{Z}_M^* = \{x \in \mathbb{Z}_M : \gcd(x, M) = 1\}$ the

unity multiplicative group in $\mathbb{Z}_M$. Then, one can define the canonical embedding

$$\begin{aligned} \sigma : \quad & R \mapsto \mathbb{C}^N \\ & m(x) \mapsto (m(\zeta^j))_{j \in \mathbb{Z}_M^*} \end{aligned}$$

where $\zeta = e^{2\pi i/M}$. At this point, we can briefly introduce the decryption and decoding algorithm of a CKKS scheme, given some ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1) \in (R/qR)^2$ (for some modulus q) and some secret decryption key $\mathsf{sk} = (1, s)$:

- **Decryption:** $\tilde{m} \leftarrow \mathsf{Dec}_{sk}(ct)$ outputs the plaintext $\tilde{m} = \langle \mathsf{ct}, \mathsf{sk} \rangle \bmod q = \mathsf{ct}_0 + \mathsf{ct}_1 s \bmod q$.
- **Decoding:** $\mathsf{Decode}(\tilde{m}, \Delta)$. For a plaintext $\tilde{m}$ and its scale factor Δ, the decoding process returns $m = \mathsf{Decode}(\tilde{m}, \Delta) = \sigma(\tilde{m}/\Delta)$.

Hence, we shall build an arithmetic circuit for:

$$\mathsf{ct} \xrightarrow{\mathsf{Dec}_{sk}(\cdot)} \tilde{m} \xrightarrow{\sigma(\cdot/\Delta)} m \ .$$

Consider a LSSS $[\![\mathsf{sk}]\!]$ of the secret key, then $[\![\tilde{m}]\!] = \langle \mathsf{ct}, [\![\mathsf{sk}]\!] \rangle \bmod q$ is a linear operation, and evaluating a shared polynomial in public points has also no multiplicative depth. Therefore, it directly results in a TZKP based on the black-box construction from Section 6.3, without communication required among provers.

Unfortunately, the secret key being a small polynomial in R, assuming that sharing small elements over large ring is efficient (see techniques from Chapter 4), the resulting shares are not compact since polynomials in R have large degree (around $2^{12} - 2^{13}$) and thus we have to pay the number of monomials of a polynomial in R in the proof.

BIBLIOGRAPHY

[AD97] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th ACM STOC*, p. 284–293, El Paso, TX, USA, 1997. ACM Press.

[ADPS16] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In T. Holz and S. Savage, eds, *USENIX Security 2016*, p. 327–343, Austin, TX, USA, 2016. USENIX Association.

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 2087–2104, Dallas, TX, USA, 2017. ACM Press.

[AKP20] B. Applebaum, E. Kachlon, and A. Patra. The round complexity of perfect MPC with active security and optimal resiliency. In S. Irani, ed., *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, p. 1277–1284. IEEE, 2020.

[AL17] G. Asharov and Y. Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptol.*, 30(1):58–151, 2017.

[ALS20] T. Attema, V. Lyubashevsky, and G. Seiler. Practical product proofs for lattice commitments. In D. Micciancio and T. Ristenpart, eds, *CRYPTO 2020, Part II*, vol. 12171 of *LNCS*, p. 470–499, Santa Barbara, CA, USA, 2020. Springer, Heidelberg, Germany.

[ASM10] M. H. Au, W. Susilo, and Y. Mu. Proof-of-knowledge of representation of committed value and its applications. In R. Steinfeld and P. Hawkes, eds, *ACISP 10*, vol. 6168 of *LNCS*, p. 352–369, Sydney, NSW, Australia, 2010. Springer, Heidelberg, Germany.

[Bab85] L. Babai. Trading group theory for randomness. In R. Sedgewick, ed., *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, p. 421–429. ACM, 1985.

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, p. 315–334, San Francisco, CA, USA, 2018. IEEE Computer Society Press.

[BBdSG+23] C. Baum, L. Braun, C. D. de Saint Guilhem, M. Klooß, E. Orsini, L. Roy, and P. Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In H. Handschuh and A. Lysyanskaya, eds, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, vol. 14085 of *Lecture Notes in Computer Science*, p. 581–615. Springer, 2023.

[BBHR18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

[BBM+24] C. Baum, W. Beullens, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. One tree to rule them all: Optimizing GGM trees and owfs for post-quantum signatures. *IACR Cryptol. ePrint Arch.*, pp. 490, 2024.

[BBSS20] X. Bonnetain, R. Bricout, A. Schrottenloher, and Y. Shen. Improved classical and quantum algorithms for subset-sum. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part II*, vol. 12492 of *LNCS*, p. 633–666, Daejeon, South Korea, 2020. Springer, Heidelberg, Germany.

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCD+16] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, eds, *ACM CCS 2016*, p. 1006–1018, Vienna, Austria, 2016. ACM Press.

[BCR13] J. Bi, Q. Cheng, and J. M. Rojas. Sub-linear root detection, and new hardness results, for sparse polynomials over finite fields. In M. Kauers, ed., *International Symposium on Symbolic and Algebraic Computation, ISSAC'13, Boston, MA, USA, June 26-29, 2013*, p. 61–68. ACM, 2013.

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, eds, *EUROCRYPT 2019, Part I*, vol. 11476 of *LNCS*, p. 103–128, Darmstadt, Germany, 2019. Springer, Heidelberg, Germany.

[BD10] R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In D. Micciancio, ed., *TCC 2010*, vol. 5978 of *LNCS*, p. 201–218, Zurich, Switzerland, 2010. Springer, Heidelberg, Germany.

[Bd20] W. Beullens and C. de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In J. Ding and J.-P. Tillich, eds, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, p. 130–150, Paris, France, 2020. Springer, Heidelberg, Germany.

[BDK+18] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, p. 353–367. IEEE, 2018.

[BDL+18] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In D. Catalano and R. De Prisco, eds, *SCN 18*, vol. 11035 of *LNCS*, p. 368–385, Amalfi, Italy, 2018. Springer, Heidelberg, Germany.

[BDLN16] C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In M. Robshaw and J. Katz, eds, *CRYPTO 2016, Part III*, vol. 9816 of *LNCS*, p. 478–498, Santa Barbara, CA, USA, 2016. Springer, Heidelberg, Germany.

[BDO14] C. Baum, I. Damgård, and C. Orlandi. Publicly auditable secure multi-party computation. In M. Abdalla and R. D. Prisco, eds, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, vol. 8642 of *Lecture Notes in Computer Science*, p. 175–196. Springer, 2014.

[BdSG20] W. Beullens and C. D. de Saint Guilhem. Legroast: Efficient post-quantum signatures from the legendre PRF. In J. Ding and J. Tillich, eds, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, vol. 12100 of *Lecture Notes in Computer Science*, p. 130–150. Springer, 2020.

[BdSGK+21] C. Baum, C. D. de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha. Banquet: Short and fast signatures from AES. In J. A. Garay, ed., *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I*, vol. 12710 of *Lecture Notes in Computer Science*, p. 266–297. Springer, 2021.

[Beu20] W. Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In A. Canteaut and Y. Ishai, eds, *EUROCRYPT 2020, Part III*, vol. 12107 of *LNCS*, p. 183–211, Zagreb, Croatia, 2020. Springer, Heidelberg, Germany.

[BFG+24] L. Bidoux, T. Feneuil, P. Gaborit, R. Neveu, and M. Rivain. Dual support decomposition in the head: Shorter signatures from rank SD and minrank. *IACR Cryptol. ePrint Arch.*, pp. 541, 2024.

[BFK+19] W. Beullens, J.-C. Faugère, E. Koussa, G. Macario-Rat, J. Patarin, and L. Perret. PKP-based signature scheme. In F. Hao, S. Ruj, and S. Sen Gupta, eds, *INDOCRYPT 2019*, vol. 11898 of *LNCS*, p. 3–22, Hyderabad, India, 2019. Springer, Heidelberg, Germany.

[BG22] L. Bidoux and P. Gaborit. Compact post-quantum signatures from proofs of knowledge leveraging structure for the pkp, sd and rsd problems. *CoRR*, abs/2204.02915, 2022.

[BGKW90] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes using two prover interactive proofs. In G. Brassard, ed., *CRYPTO'89*, vol. 435 of *LNCS*, p. 498–506, Santa Barbara, CA, USA, 1990. Springer, Heidelberg, Germany.

[BGV14] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In J. Simon, ed., *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, p. 1–10. ACM, 1988.

[BHH01] D. Boneh, S. Halevi, and N. Howgrave-Graham. The modular inversion hidden number problem. In C. Boyd, ed., *ASIACRYPT 2001*, vol. 2248 of *LNCS*, p. 36–51, Gold Coast, Australia, 2001. Springer, Heidelberg, Germany.

[BKK90] J. Boyar, S. A. Kurtz, and M. W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[BKLP15] F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In G. Pernul, P. Y. A. Ryan, and E. R. Weippl, eds, *ESORICS 2015, Part I*, vol. 9326 of *LNCS*, p. 305–325, Vienna, Austria, 2015. Springer, Heidelberg, Germany.

[Blo09] J. Blocki. Direct zero-knowledge proofs. Senior Research Thesis, B.S. in Computer Science, Carnegie Mellon University, 2009.

[BLS19] J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In A. Boldyreva and D. Micciancio, eds, *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, p. 176–202, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[Blu82] M. Blum. Coin flipping by telephone - A protocol for solving impossible problems. In *COMPCON'82, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference, San Francisco, California, USA, February 22-25, 1982*, p. 133–137. IEEE Computer Society, 1982.

[BM97] M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In W. Fumy, ed., *EUROCRYPT'97*, vol. 1233 of *LNCS*, p. 163–192, Konstanz, Germany, 1997. Springer, Heidelberg, Germany.

[BN20a] C. Baum and A. Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, eds, *PKC 2020, Part I*, vol. 12110 of *LNCS*, p. 495–526, Edinburgh, UK, 2020. Springer, Heidelberg, Germany.

[BN20b] C. Baum and A. Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, eds, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*, vol. 12110 of *Lecture Notes in Computer Science*, p. 495–526. Springer, 2020.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pp. 113–131, New York, NY, USA, 1988. Association for Computing Machinery.

[BR60] R. C. Bose and D. K. Ray-Chaudhuri. On A class of error correcting binary group codes. *Inf. Control.*, 3(1):68–79, 1960.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, eds, *CRYPTO 2012*, vol. 7417 of *LNCS*, p. 868–886, Santa Barbara, CA, USA, 2012. Springer, Heidelberg, Germany.

[BTV20] O. Blazy, P. Towa, and D. Vergnaud. Public-key generation with verifiable randomness. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part I*, vol. 12491 of *LNCS*, p. 97–127, Daejeon, South Korea, 2020. Springer, Heidelberg, Germany.

[BVZ12] A. Bauer, D. Vergnaud, and J.-C. Zapalowicz. Inferring sequences produced by nonlinear pseudorandom number generators using Coppersmith's methods. In M. Fischlin, J. Buchmann, and M. Manulis, eds, *PKC 2012*, vol. 7293 of *LNCS*, p. 609–626, Darmstadt, Germany, 2012. Springer, Heidelberg, Germany.

[CCP22] A. Chandramouli, A. Choudhury, and A. Patra. A survey on perfectly secure verifiable secret-sharing. *ACM Comput. Surv.*, 54(11s):232:1–232:36, 2022.

[CDG+17] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 1825–1842, Dallas, TX, USA, 2017. ACM Press.

[CDG+20] M. Chase, D. Derler, S. Goldfeder, J. Katz, V. Kolesnikov, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, X. Wang, and G. Zaverucha. The Picnic Signature Scheme – Design Document. Version 2.2 – 14 April 2020, 2020.

[CG07] S. Canard and A. Gouget. Divisible e-cash systems can be truly anonymous. In M. Naor, ed., *EUROCRYPT 2007*, vol. 4515 of *LNCS*, p. 482–497, Barcelona, Spain, 2007. Springer, Heidelberg, Germany.

[CGGI20] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CGH00] D. Catalano, R. Gennaro, and S. Halevi. Computing inverses over a shared secret modulus. In B. Preneel, ed., *EUROCRYPT 2000*, vol. 1807 of *LNCS*, p. 190–206, Bruges, Belgium, 2000. Springer, Heidelberg, Germany.

[CGM16] M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In M. Robshaw and J. Katz, eds, *CRYPTO 2016, Part III*, vol. 9816 of *LNCS*, p. 499–530, Santa Barbara, CA, USA, 2016. Springer, Heidelberg, Germany.

[CHM+20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. In A. Canteaut and Y. Ishai, eds, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, vol. 12105 of *Lecture Notes in Computer Science*, p. 738–768. Springer, 2020.

[CJL+92] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Comput. Complex.*, 2:111–128, 1992.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, eds, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, vol. 10624 of *Lecture Notes in Computer Science*, p. 409–437. Springer, 2017.

[CKR+20] H. Chen, M. Kim, I. P. Razenshteyn, D. Rotaru, Y. Song, and S. Wagh. Maliciously secure matrix multiplication with applications to private deep learning. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part III*, vol. 12493 of *LNCS*, p. 31–59, Daejeon, South Korea, 2020. Springer, Heidelberg, Germany.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, p. 494–503, Montréal, Québec, Canada, 2002. ACM Press.

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. S. Kaliski Jr., ed., *CRYPTO'97*, vol. 1294 of *LNCS*, p. 410–424, Santa Barbara, CA, USA, 1997. Springer, Heidelberg, Germany.

[CY19] C. Crépeau and N. Yang. Non-locality in interactive proofs. *Electron. Colloquium Comput. Complex.*, TR19-030, 2019.

[Des87] Y. Desmedt. Society and group oriented cryptography: A new concept. In C. Pomerance, ed., *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, vol. 293 of *Lecture Notes in Computer Science*, p. 120–127. Springer, 1987.

[Des94] Y. Desmedt. Threshold cryptography. *Eur. Trans. Telecommun.*, 5(4):449–458, 1994.

[DG23] Q. Dao and P. Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In C. Hazay and M. Stam, eds, *EUROCRYPT 2023, Part II*, vol. 14005 of *Lecture Notes in Computer Science*, p. 531–562. Springer, 2023.

[DGH+21] I. Dinur, S. Goldfeder, T. Halevi, Y. Ishai, M. Kelkar, V. Sharma, and G. Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In T. Malkin and C. Peikert, eds, *CRYPTO 2021, Part IV*, vol. 12828 of *LNCS*, p. 517–547, Virtual Event, 2021. Springer, Heidelberg, Germany.

[DKR+22] C. Dobraunig, D. Kales, C. Rechberger, M. Schofnegger, and G. Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In H. Yin, A. Stavrou, C. Cremers, and E. Shi, eds, *ACM CCS 2022*, p. 843–857, Los Angeles, CA, USA, 2022. ACM Press.

[DKR23] J. Doerner, Y. Kondi, and L. N. Rosenbloom. Sometimes you can't distribute random-oracle-based proofs. *IACR Cryptol. ePrint Arch.*, pp. 1381, 2023.

[DPSZ12] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In R. Safavi-Naini and R. Canetti, eds, *CRYPTO 2012*, vol. 7417 of *LNCS*, p. 643–662, Santa Barbara, CA, USA, 2012. Springer, Heidelberg, Germany.

[ENS20] M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In S. Moriai and H. Wang, eds, *ASIACRYPT 2020, Part II*, vol. 12492 of *LNCS*, p. 259–288, Daejeon, South Korea, 2020. Springer, Heidelberg, Germany.

[Fen24] T. Feneuil. Building mpcith-based signatures from mq, minrank, and rank SD. In C. Pöpper and L. Batina, eds, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*, vol. 14583 of *Lecture Notes in Computer Science*, p. 403–431. Springer, 2024.

[FJR22] T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Y. Dodis and T. Shrimpton, eds, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, vol. 13508 of *Lecture Notes in Computer Science*, p. 541–572. Springer, 2022.

[FJR23] T. Feneuil, A. Joux, and M. Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Des. Codes Cryptogr.*, 91(2):563–608, 2023.

[FMRV22] T. Feneuil, J. Maire, M. Rivain, and D. Vergnaud. Zero-knowledge protocols for the subset sum problem from mpc-in-the-head with rejection. In S. Agrawal and D. Lin, eds, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, vol. 13792 of *Lecture Notes in Computer Science*, p. 371–402. Springer, 2022.

[FR23a] T. Feneuil and M. Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. *IACR Cryptol. ePrint Arch.*, pp. 1573, 2023.

[FR23b] T. Feneuil and M. Rivain. Threshold linear secret sharing to the rescue of mpc-in-the-head. In J. Guo and R. Steinfeld, eds, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part I*, vol. 14438 of *Lecture Notes in Computer Science*, p. 441–473. Springer, 2023.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, ed., *CRYPTO'86*, vol. 263 of *LNCS*, p. 186–194, Santa Barbara, CA, USA, 1987. Springer, Heidelberg, Germany.

[FV12] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, p. 464–479. IEEE Computer Society, 1984.

[GHM+22] K. Gjøsteen, T. Haines, J. Müller, P. B. Rønne, and T. Silde. Verifiable decryption in the head. In K. Nguyen, G. Yang, F. Guo, and W. Susilo, eds, *Information Security and Privacy - 27th Australasian Conference, ACISP 2022, Wollongong, NSW, Australia, November 28-30, 2022, Proceedings*, vol. 13494 of *Lecture Notes in Computer Science*, p. 355–374. Springer, 2022.

[GIKR01] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, eds, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, p. 580–589. ACM, 2001.

[GMO16] I. Giacomelli, J. Madsen, and C. Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In T. Holz and S. Savage, eds, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, p. 1069–1083. USENIX Association, 2016.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. V. Aho, ed., *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, p. 218–229. ACM, 1987.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol04] O. Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[GOP+22] C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In O. Dunkelman and S. Dziembowski, eds, *EUROCRYPT 2022, Part II*, vol. 13276 of *LNCS*, p. 397–426, Trondheim, Norway, 2022. Springer, Heidelberg, Germany.

[Gou01] L. Goubin. A sound method for switching between Boolean and arithmetic masking. In Çetin Kaya. Koç, D. Naccache, and C. Paar, eds, *CHES 2001*, vol. 2162 of *LNCS*, p. 3–15, Paris, France, 2001. Springer, Heidelberg, Germany.

[GPS12] H. Ghodosi, J. Pieprzyk, and R. Steinfeld. Multi-party computation with conversion of secret sharing. *Des. Codes Cryptogr.*, 62(3):259–272, 2012.

[GQ90] L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, ed., *CRYPTO'88*, vol. 403 of *LNCS*, p. 216–231, Santa Barbara, CA, USA, 1990. Springer, Heidelberg, Germany.

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, p. 212–219, Philadephia, PA, USA, 1996. ACM Press.

[Gro16a] J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, eds, *EUROCRYPT 2016, Part II*, vol. 9666 of *LNCS*, p. 305–326, Vienna, Austria, 2016. Springer, Heidelberg, Germany.

[Gro16b] J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J. Coron, eds, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, vol. 9666 of *Lecture Notes in Computer Science*, p. 305–326. Springer, 2016.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, pp. 953, 2019.

[HILL99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HJ10] N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, ed., *EUROCRYPT 2010*, vol. 6110 of *LNCS*, p. 235–256, French Riviera, 2010. Springer, Heidelberg, Germany.

[HS74] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974.

[IKOS07] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In D. S. Johnson and U. Feige, eds, *39th ACM STOC*, p. 21–30, San Diego, CA, USA, 2007. ACM Press.

[IKOS09] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[IN96] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

[JKPT12] A. Jain, S. Krenn, K. Pietrzak, and A. Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In X. Wang and K. Sako, eds, *ASIACRYPT 2012*, vol. 7658 of *LNCS*, p. 663–680, Beijing, China, 2012. Springer, Heidelberg, Germany.

[Jou23] A. Joux. MPC in the head for isomorphisms and group actions. *IACR Cryptol. ePrint Arch.*, pp. 664, 2023.

[Joy21] M. Joye. Guide to fully homomorphic encryption over the [discretized] torus. Cryptology ePrint Archive, Report 2021/1402, 2021.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, eds, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, p. 85–103. Plenum Press, New York, 1972.

[Kel16] Z. Kelley. Roots of sparse polynomials over a finite field. *LMS Journal of Computation and Mathematics*, 19(A):196–204, 2016.

[Kil89] J. Kilian. *Uses of randomness in algorithms and protocols*. PhD thesis, Massachusetts Institute of Technology, 1989.

[KKK09] J. Katz, C. Koo, and R. Kumaresan. Improving the round complexity of VSS in point-to-point networks. *Inf. Comput.*, 207(8):889–899, 2009.

[KKW18] J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In D. Lie, M. Mannan, M. Backes, and X. Wang, eds, *ACM CCS 2018*, p. 525–537, Toronto, ON, Canada, 2018. ACM Press.

[KMR12] M. Keller, G. L. Mikkelsen, and A. Rupp. Efficient threshold zero-knowledge with applications to user-centric protocols. In A. D. Smith, ed., *Information Theoretic Security - 6th International Conference, ICITS 2012, Montreal, QC, Canada, August 15-17, 2012. Proceedings*, vol. 7412 of *Lecture Notes in Computer Science*, p. 147–166. Springer, 2012.

[KPR18] M. Keller, V. Pastro, and D. Rotaru. Overdrive: Making SPDZ great again. In J. B. Nielsen and V. Rijmen, eds, *EUROCRYPT 2018, Part III*, vol. 10822 of *LNCS*, p. 158–189, Tel Aviv, Israel, 2018. Springer, Heidelberg, Germany.

[KZ20] D. Kales and G. Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In S. Krenn, H. Shulman, and S. Vaudenay, eds, *CANS 20*, vol. 12579 of *LNCS*, p. 3–22, Vienna, Austria, 2020. Springer, Heidelberg, Germany.

[KZ22] D. Kales and G. Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Paper 2022/588, 2022.

[KZGM21] S. Kanjalkar, Y. Zhang, S. Gandlur, and A. Miller. Publicly auditable mpc-as-a-service with succinct verification and universal setup. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, p. 386–411. IEEE, 2021.

[LN17] Y. Lindell and A. Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds, *ACM CCS 2017*, p. 259–276, Dallas, TX, USA, 2017. ACM Press.

[LNS21] V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In J. Garay, ed., *PKC 2021, Part I*, vol. 12710 of *LNCS*, p. 215–241, Virtual Event, 2021. Springer, Heidelberg, Germany.

[LNSW13] S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In K. Kurosawa and G. Hanaoka, eds, *PKC 2013*, vol. 7778 of *LNCS*, p. 107–124, Nara, Japan, 2013. Springer, Heidelberg, Germany.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, ed., *EUROCRYPT 2010*, vol. 6110 of *LNCS*, p. 1–23, French Riviera, 2010. Springer, Heidelberg, Germany.

[LPS10] V. Lyubashevsky, A. Palacio, and G. Segev. Public-key cryptographic primitives provably as secure as subset sum. In D. Micciancio, ed., *TCC 2010*, vol. 5978 of *LNCS*, p. 382–400, Zurich, Switzerland, 2010. Springer, Heidelberg, Germany.

[LSSW12] S. Ling, I. E. Shparlinski, R. Steinfeld, and H. Wang. On the modular inversion hidden number problem. *J. Symb. Comput.*, 47(4):358–367, 2012.

[Lyu08] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In R. Cramer, ed., *PKC 2008*, vol. 4939 of *LNCS*, p. 162–179, Barcelona, Spain, 2008. Springer, Heidelberg, Germany.

[Lyu09] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, ed., *ASIACRYPT 2009*, vol. 5912 of *LNCS*, p. 598–616, Tokyo, Japan, 2009. Springer, Heidelberg, Germany.

[MGH+23] C. A. Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. The return of the sdith. In C. Hazay and M. Stam, eds, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, vol. 14008 of *Lecture Notes in Computer Science*, p. 564–596. Springer, 2023.

[MH78] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inf. Theory*, 24(5):525–530, 1978.

[MS81] R. J. McEliece and D. V. Sarwate. On sharing secrets and reed-solomon codes. *Commun. ACM*, 24(9):583–584, 1981.

[MV23a] J. Maire and D. Vergnaud. Commitments with efficient zero-knowledge arguments from subset sum problems. In G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, eds, *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*, vol. 14344 of *Lecture Notes in Computer Science*, p. 189–208. Springer, 2023.

[MV23b] J. Maire and D. Vergnaud. Efficient zero-knowledge arguments and digital signatures via sharing conversion in the head. In G. Tsudik, M. Conti, K. Liang, and G. Smaragdakis, eds, *Computer Security - ESORICS 2023 - 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25-29, 2023, Proceedings, Part I*, vol. 14344 of *Lecture Notes in Computer Science*, p. 435–454. Springer, 2023.

[MV24] J. Maire and D. Vergnaud. Secure multi-party linear algebra with perfect correctness. *IACR Commun. Cryptol.*, 1(1):29, 2024.

[OB22] A. Ozdemir and D. Boneh. Experimenting with collaborative zk-snarks: Zero-knowledge proofs for distributed secrets. In K. R. B. Butler and K. Thomas, eds, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, p. 4291–4308. USENIX Association, 2022.

[Odl90] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. Cryptology and computational number theory, Lect. Notes AMS Short Course, Boulder/CO (USA) 1989, Proc. Symp. Appl. Math. 42, 75-88 (1990)., 1990.

[Ped92] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, ed., *CRYPTO'91*, vol. 576 of *LNCS*, p. 129–140, Santa Barbara, CA, USA, 1992. Springer, Heidelberg, Germany.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PS04] C. Padró and G. Sáez. Correction to "secret sharing schemes with bipartite access structure". *IEEE Trans. Inf. Theory*, 50(6):1373, 2004.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, eds, *37th ACM STOC*, p. 84–93, Baltimore, MA, USA, 2005. ACM Press.

[Rot11] R. Rothblum. Homomorphic encryption: From private-key to public-key. In Y. Ishai, ed., *TCC 2011*, vol. 6597 of *LNCS*, p. 219–234, Providence, RI, USA, 2011. Springer, Heidelberg, Germany.

[RRRK22] M. Rivinius, P. Reisert, D. Rausch, and R. Küsters. Publicly accountable robust multi-party computation. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, p. 2430–2449. IEEE, 2022.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sha86] A. Shamir. A zero-knowledge proof for knapsacks. presented at a workshop on Probabilistic Algorithms, Marseille, 1986.

[Sha90] A. Shamir. An efficient identification scheme based on permuted kernels (extended abstract) (rump session). In G. Brassard, ed., *CRYPTO'89*, vol. 435 of *LNCS*, p. 606–609, Santa Barbara, CA, USA, 1990. Springer, Heidelberg, Germany.

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, p. 124–134, Santa Fe, NM, USA, 1994. IEEE Computer Society Press.

[Sim91] G. Simmons. Identification of data, devices, documents and individuals. In *Proceedings. 25th Annual 1991 IEEE International Carnahan Conference on Security Technology*, p. 197–218, 1991.

[SPW06] R. Steinfeld, J. Pieprzyk, and H. Wang. On the provable security of an efficient RSA-based pseudorandom generator. In X. Lai and K. Chen, eds, *ASIACRYPT 2006*, vol. 4284 of *LNCS*, p. 194–209, Shanghai, China, 2006. Springer, Heidelberg, Germany.

[SS81] R. Schroeppel and A. Shamir. A T=O($2^{n/2}$), S=O($2^{n/4}$) algorithm for certain NP-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981.

[Sta96] M. Stadler. Publicly verifiable secret sharing. In U. M. Maurer, ed., *EUROCRYPT'96*, vol. 1070 of *LNCS*, p. 190–199, Saragossa, Spain, 1996. Springer, Heidelberg, Germany.

[Ste94a] J. Stern. Designing identification schemes with keys of short size. In Y. Desmedt, ed., *CRYPTO'94*, vol. 839 of *LNCS*, p. 164–173, Santa Barbara, CA, USA, 1994. Springer, Heidelberg, Germany.

[Ste94b] J. Stern. A new identification scheme based on syndrome decoding. In D. R. Stinson, ed., *CRYPTO'93*, vol. 773 of *LNCS*, p. 13–21, Santa Barbara, CA, USA, 1994. Springer, Heidelberg, Germany.

[TD09] T. Tassa and N. Dyn. Multipartite secret sharing by bivariate interpolation. *J. Cryptol.*, 22(2):227–258, 2009.

[Win84] R. S. Winternitz. A secure one-way hash function built from DES. In *Proceedings of the 1984 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 29 - May 2, 1984*, p. 88–90. IEEE Computer Society, 1984.

[XSH+19] J. Xu, S. Sarkar, L. Hu, H. Wang, and Y. Pan. New results on modular inversion hidden number problem and inversive congruential generator. In A. Boldyreva and D. Micciancio, eds, *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, p. 297–321, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[Yao82] A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, p. 160–164. IEEE Computer Society, 1982.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, p. 162–167, Toronto, Ontario, Canada, 1986. IEEE Computer Society Press.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In E. W. Ng, ed., *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, vol. 72 of *Lecture Notes in Computer Science*, p. 216–226. Springer, 1979.

APPENDIX

A. THE 3-ROUND VARIANT OF PROTOCOL 6

Prover P		Verifier V
$x \in \{0,1\}^n$		
$w \in \mathbb{Z}_q^n, t = \langle w, x \rangle$		w, t
$\mathsf{mseed}^{[0]} \xleftarrow{\$} \{0,1\}^\lambda$		
$(\mathsf{mseed}^{[e]})_{j\in[1,M]} \leftarrow \mathrm{PRG}(\mathsf{mseed}^{[0]})$		
For each $e \in [1, M]$:		
$r^{[e]}, \leftarrow \mathrm{PRG}(\mathsf{mseed}^{[e]})$		$\triangleright\ r^{[e]} \in \{0,1\}^n$
$(\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\in[1,N]} \leftarrow \mathrm{PRG}(\mathsf{mseed}^{[e]})$		
For each $i \in [1, N]$:		
$[\![r^{[e]}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i^{[e]})$		$\triangleright\ [\![r^{[e]}]\!]_i \in [0, A-1]^n$
$\mathsf{com}_i^{[e]} = \mathrm{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$		
$\Delta r^{[e]} = r^{[e]} - \sum_i [\![r^{[e]}]\!]_i$		
$h_j = \mathcal{H}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$		
$\tilde{x}^{[e]} = x \oplus r^{[e]}$		$\triangleright\ \oplus$ is the XOR operation ($\tilde{x} \in \{0,1\}^n$)
The parties locally set		
$[\![x^{[e]}]\!] = (1 - \tilde{x}^{[e]}) \circ [\![r^{[e]}]\!]$		
$+\tilde{x}^{[e]} \circ (1 - [\![r^{[e]}]\!])$		
and they set $[\![t^{[e]}]\!] = \langle w, [\![x^{[e]}]\!] \rangle$.		
$h'_e = \mathcal{H}_3(\tilde{x}^{[e]}, [\![t^{[e]}]\!])$		
$h' = \mathrm{Merkle}(h'_1, \ldots, h'_M)$		
$h = \mathcal{H}_2(h_1, \ldots, h_M, h')$	$\xrightarrow{h}$	$J \xleftarrow{\$} \{J \subset [1, M]\ ;\ \lvert J\rvert = \tau\}$
	$\xleftarrow{L}$	$L = \{\ell_e\}_{e\in J} \xleftarrow{\$} [1, N]^\tau$
If there exists $(e, j) \in J \times [1, n]$ such that:		
- either $[\![r_j^{[e]}]\!]_{\ell_e} = 0$ with $r_j = 1$		
- or $[\![r_j^{[e]}]\!]_{\ell_e} = A - 1$ with $r_j = 0$, then abort.		
$\mathsf{auth}^{\mathrm{Merkle}} := \mathsf{auth}((h'_1, \ldots, h'_M), J)$		
$\sigma = \mathsf{auth}^{\mathrm{Merkle}} \mid (\mathsf{mseed}^{[e]})_{j\in[1,M]\setminus J}$		
$\sigma = \sigma \mid \begin{pmatrix} (\mathsf{seed}_i^{[e]}, \rho_i^{[e]})_{i\neq\ell_e} \\ r^{[e]} - [\![r^{[e]}]\!]_{\ell_e} \\ \tilde{x}^{[e]},\ \mathsf{com}_{\ell_e} \end{pmatrix}_{e\in J}$	$\xrightarrow{\sigma}$	
		For each $e \notin J$:
		Compute h_e using $\mathsf{mseed}^{[e]}$
		For each $e \in J$:
		For all $i \neq \ell_e$
		$\mathsf{com}_i^{[e]} = \mathrm{Com}(\mathsf{seed}_i^{[e]}; \rho_i^{[e]})$
		Rerun the party i
		as the prover to get $[\![t^{[e]}]\!]_i$
		$\Delta r^{[e]} = (r^{[e]} - [\![r^{[e]}]\!]_{\ell_e}) - \sum_{i\neq\ell_j} [\![r^{[e]}]\!]$
		$h_e = \mathcal{H}_1(\Delta r^{[e]}, \mathsf{com}_1^{[e]}, \ldots, \mathsf{com}_n^{[e]})$
		$[\![t^{[e]}]\!] = t - \Delta t^{[e]} - \sum_{i\neq\ell_j} [\![t^{[e]}]\!]_i$
		$h'_e = \mathcal{H}_3(\tilde{x}^{[e]}, [\![t^{[e]}]\!])$
		Using $\mathsf{auth}^{\mathrm{Merkle}}$, check that $\{h'_e\}_{e\in J}$ are consistent and deduce the Merkle root h'.
		Check $h = \mathcal{H}_2(h_1, \ldots, h_M, h')$
		Return 1

Protocol 14: Zero-knowledge argument (3-round variant) for Subset Sum Problem via MPC-in-the-Head paradigm with rejection, using cut-and-choose strategy to prove binarity.

B. SIGNATURE SCHEMES WITH SUBSET SUM PROBLEM

The Fiat-Shamir heuristic [FS87] is a method to convert Σ-protocols (a specific class of ZK proofs) into non-interactive ZK proofs and hence can be used to build signature. Using this heuristic we can transform our two protocols into signature schemes. For each of them, we explain how to apply the Fiat-Shamir transform and how to evaluate the obtained security.

Signature from Protocol 5. We compute the challenges $\{\varepsilon^{[e]}\}_{e\in[1,\tau]}$ and $\{i^{*[e]}\}_{e\in[1,\tau]}$ for τ executions as:

$$\{\varepsilon^{[e]}\}_{e\in[1,\tau]} := \mathcal{H}'_1(m,h)$$

and

$$\{i^{*[e]}\}_{e\in[1,\tau]} := \mathcal{H}'_2(m,h,h')$$

where m is the input message, $\mathcal{H}'_1$ and $\mathcal{H}'_2$ are some hash functions, and h (resp. h') is the hash value corresponding to the merged inputs of $\mathcal{H}_1$ (resp. $\mathcal{H}_2$) from the τ executions.

Since the protocol has 5 rounds, we must take into account the forgery attack described in [KZ20] to estimate the security of the resulting signature. When we adapt the attack for Protocol 5, its cost is given by

$$\mathrm{cost}_{\mathrm{forge}} = \min_{\tau_1,\tau_2:\tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \mathrm{PMF}(i,\tau,\frac{1}{q'})} + \frac{1}{\sum_{i=0}^{\eta} \mathrm{PMF}(i,\tau_2,1-\frac{1}{N})} \right\},$$

with $\mathrm{PMF}(i,\tau,p) := \binom{\tau}{i} p^i (1-p)^{\tau-i}$. When selecting the signature parameters, we must choose τ such that $\mathrm{cost}_{\mathrm{forge}} \geq 2^{\lambda}$.

Signature from Protocol 6. The challenges J and L are computed as

$$J := \mathcal{H}'_1(m,h)$$

and

$$L := \mathcal{H}'_2(m,h,h',(\mathsf{mseed}^{[j]})_{j\in[1,M]\setminus J})$$

where m is the input message and where $\mathcal{H}'_1$ and $\mathcal{H}'_2$ are some hash functions.

Since the protocol has 5 rounds, the security of the resulting signature scheme is given by the attack of [KZ20] which has, in the context of the Protocol 6, a forgery cost of

$$\mathrm{cost}_{\mathrm{forge}} = \min_{M-\tau\leq k\leq M} \left\{ \frac{\binom{M}{M-\tau}}{\binom{k}{M-\tau}} + \frac{1}{\sum_{i=0}^{\eta} \mathrm{PMF}(i,k-M+\tau,1-\frac{1}{N})} \right\}.$$

Another approach consists in turning the 5-round protocol into a 3-round protocol (before applying the Fiat-Shamir). We refer to [KKW18, FJR23] for the details of such an approach. We provide a formal description of the 3-round variant of the protocol in Appendix A. The soundness error of this variant is the same as for the original protocol (see Theorem 9). When we apply the Fiat-Shamir to this variant, the security of the obtained signature scheme is equal to the soundness error of the protocol (since the protocol has now only 3 rounds) and its size (in bits) is

$$\textsc{Size}_{\eta} = 4\lambda + \eta 4\lambda + 3\lambda\tau\log_2\frac{M}{\tau} + (\tau-\eta)\left[n\log_2(A-1) + n + \lambda\log_2 N + 2\lambda\right].$$

Performances. We selected some parameter sets to instantiate the resulting signature schemes while targeting a security of 128 bits and a rejection rate of 0.01. We obtained the performances of Table B.1.

Signature	Parameters					Proof size	Rej. rate	Security
	τ	η	N	A	M			
Protocol 5 (batching)	29	2	256	2^{14}	-	28.1 KB	0.010	129 bits
Protocol 5 (batching)	42	3	32	2^{14}	-	38.7 KB	0.004	128 bits
Protocol 6 (C&C), 5 rounds	46	3	256	2^{14}	993	30.3 KB	0.006	128 bits
Protocol 6 (C&C), 5 rounds	71	3	32	2^{14}	452	42.5 KB	0.025	128 bits
Protocol 6 (C&C), 3 rounds	28	2	64	2^{14}	514	21.1 KB	0.009	128 bits
Protocol 6 (C&C), 3 rounds	53	3	8	2^{14}	253	33.2 KB	0.009	128 bits

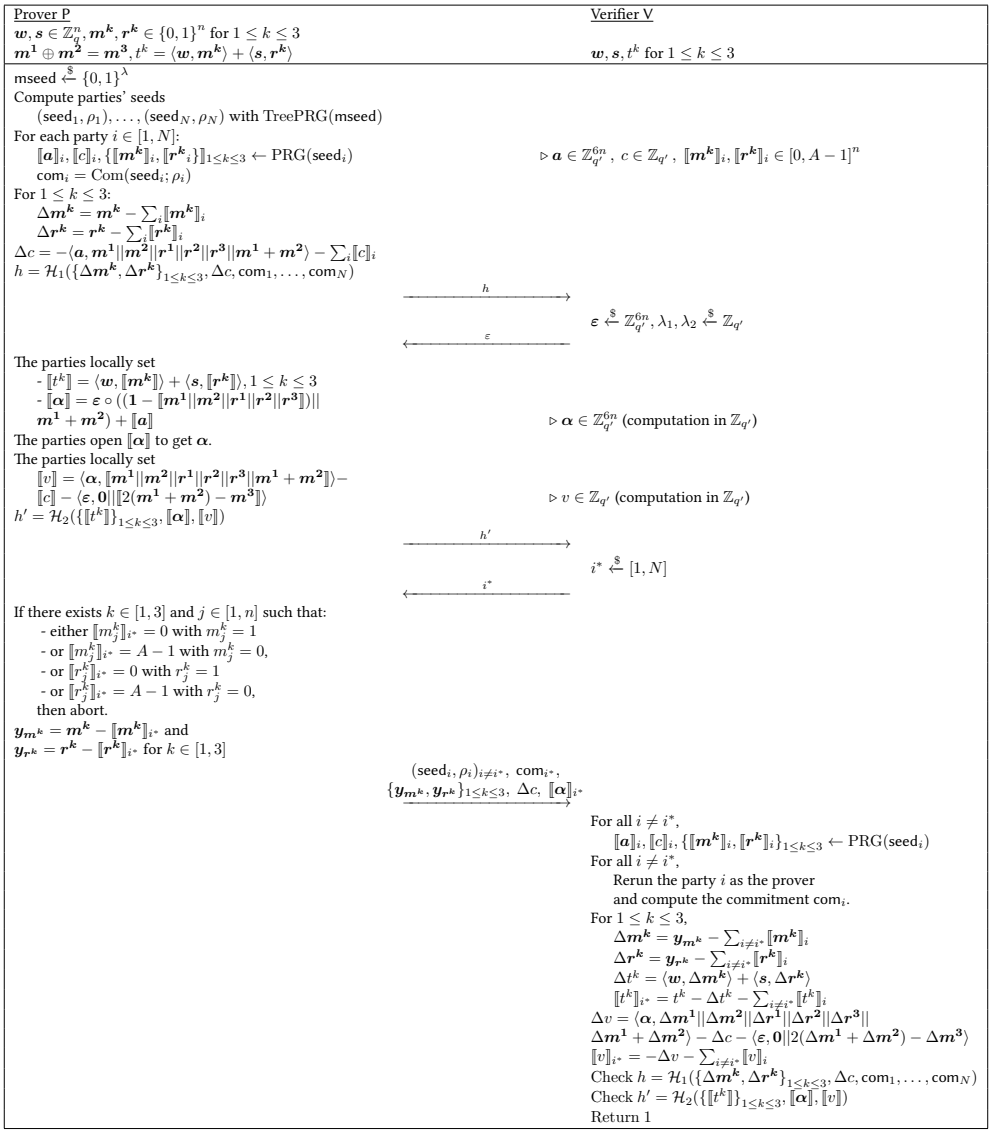
Tab. B.1: Performance of the obtained signatures

C. ZERO-KNOWLEDGE ARGUMENT FOR BONEH-HALEVI-HOWGRAVE-GRAHAM PRF

Prover P		Verifier V
$x \in \{0,1\}^n$		
$(z_1, y_1), \ldots, (z_t, y_t)$		$y_1, \ldots, y_t$
$\mathsf{mseed} \xleftarrow{\$} \{0,1\}^\lambda$		
Compute parties' seeds		
$(\mathsf{seed}_1, \rho_1), \ldots, (\mathsf{seed}_N, \rho_N)$ with $\mathrm{TreePRG}(\mathsf{mseed})$		
For each party $i \in [1, N]$:		
$[\![x]\!]_i, [\![a]\!]_i, [\![c]\!]_i, [\![\boldsymbol{z}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$		$\triangleright\ [\![x]\!], [\![a]\!]_i, [\![c]\!]_i \in \mathbb{Z}_p, [\![\boldsymbol{z}]\!]_i \in [0, A-1]^t$
$\mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$		
$\Delta x = x - \sum_i [\![x]\!]_i$		
$\Delta c = ax - \sum_i [\![c]\!]_i$		
$\Delta \boldsymbol{z} = \boldsymbol{z} - \sum_i [\![\boldsymbol{z}]\!]_i$		
$h = \mathcal{H}_1(\Delta x, \Delta c, \Delta \boldsymbol{z}, \mathsf{com}_1, \ldots, \mathsf{com}_N)$		
	$\xrightarrow{h}$	
		$\gamma_1, \ldots, \gamma_t, \varepsilon \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{\boldsymbol{\gamma}, \varepsilon}$	
The parties locally set		
- $[\![\alpha]\!] = \varepsilon \langle \boldsymbol{\gamma}, [\![\boldsymbol{z}]\!] \rangle + [\![a]\!] \bmod p$		
- $[\![r]\!] =$ "right part of Equation 4.15"		
The parties open $[\![\alpha]\!]$ to get α.		
The parties locally set		
$[\![v]\!] = \varepsilon [\![r]\!] - \alpha [\![x]\!] + [\![c]\!] \bmod p$		
$h' = \mathcal{H}_2([\![\alpha]\!], [\![v]\!])$		
	$\xrightarrow{h'}$	
		$i^* \xleftarrow{\$} [1, N]$
	$\xleftarrow{i^*}$	
$\boldsymbol{\mu} = \boldsymbol{z} - [\![\boldsymbol{z}]\!]_{i^*}$		
If there exists $j \in [t]$ such that:		
- either $\mu_j \geq 1$		
- or $\mu_j \leq -A + B - 1$,		
then abort.		
	$(\mathsf{seed}_i, \rho_i)_{i \neq i^*}, \mathsf{com}_{i^*},$ $\boldsymbol{\mu}, \Delta c, [\![\alpha]\!]_{i^*}$ $\xrightarrow{\quad}$	
		For all $i \neq i^*$,
		$[\![x]\!]_i, [\![a]\!]_i, [\![c]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		$[\![\boldsymbol{z}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		$\Delta \boldsymbol{z} = \boldsymbol{\mu} - \sum_{i \neq i^*} [\![\boldsymbol{z}]\!]_i$
		$\Delta \alpha = \varepsilon \cdot \langle \boldsymbol{\gamma}, \Delta \boldsymbol{z} \rangle$
		For all $i \neq i^*$,
		Rerun the party i as the prover
		and compute the commitment com_i.
		$\Delta r =$ deduced from the right part of Eq. 4.15
		$\Delta v = \varepsilon \Delta r - \alpha \Delta x - \Delta c$
		$[\![v]\!]_{i^*} = -\Delta v - \sum_{i \neq i^*} [\![v]\!]_i$
		Check $h = \mathcal{H}_1(\Delta x, \Delta c, \Delta \boldsymbol{z}, \mathsf{com}_1, \ldots, \mathsf{com}_N)$
		Check $h' = \mathcal{H}_2([\![\alpha]\!], [\![v]\!])$
		Return 1

Protocol 15: Relaxed zero-knowledge argument for Boneh et al's PRF.

D. DESCRIPTION OF PROTOCOLS 16, 17, AND 18

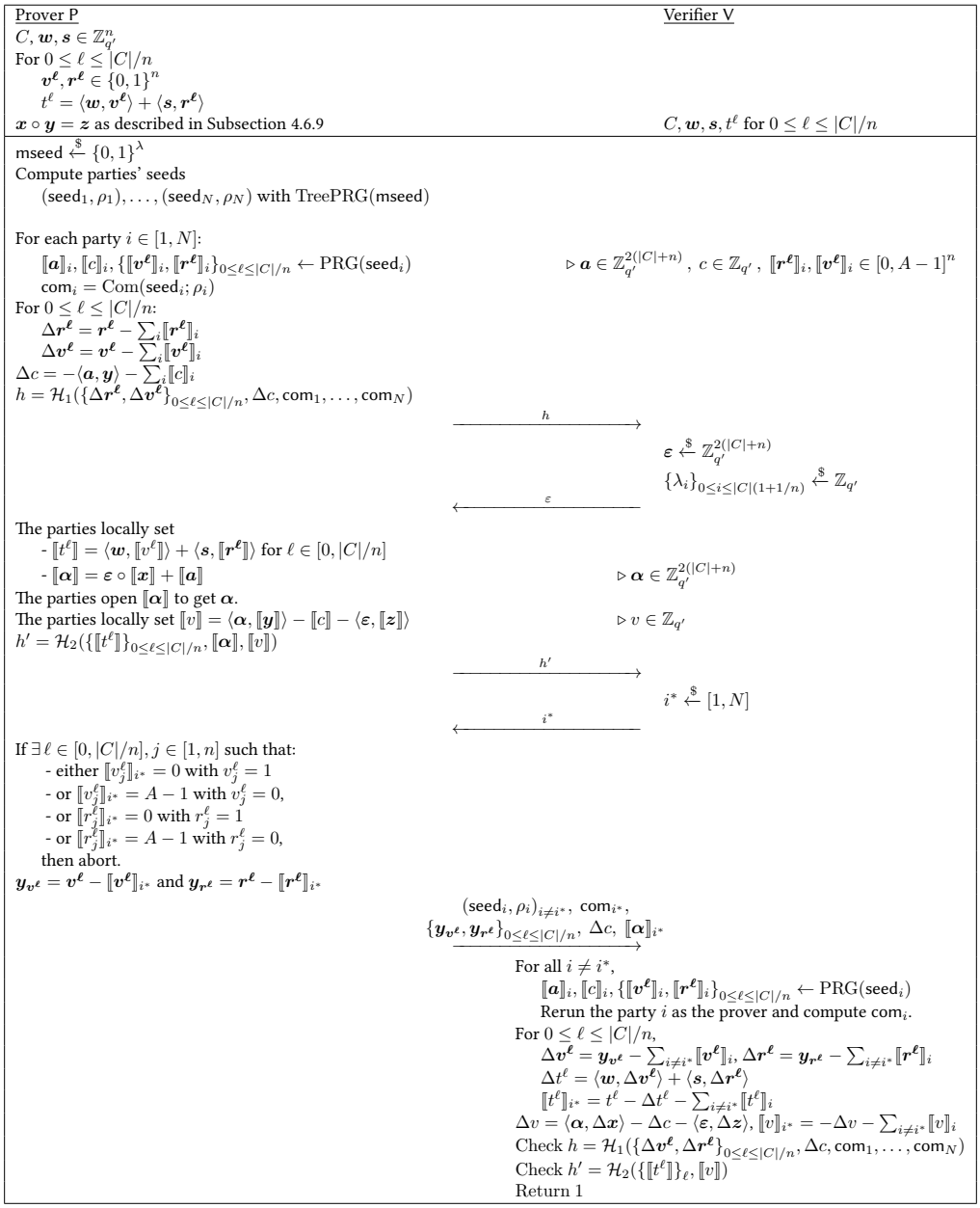


Protocol 16: Zero-knowledge argument for XOR gate

In order to describe the circuit during Protocol 17, we set $S \leftarrow \emptyset$. Then construct S as follows: if $m_{x_k}^{\ell} \wedge m_{y_k}^{\ell_k} = m_{z_k}^{\ell'_k}$ for $k \in [1, M], \{\ell, \ell_k, \ell'_k\} \in [1, L]^3, \{x_k, y_k, z_k\} \in [1, n]^3$, then $S = S \cup \{(\ell, x_k; \ell_k, y_k; \ell'_k, z_k)\}$.

Prover P		Verifier V
$\boldsymbol{w}, \boldsymbol{s} \in \mathbb{Z}_q^n, S$		
For $1 \le \ell \le L$,		
$\boldsymbol{m}^{\boldsymbol{\ell}}, \boldsymbol{r}^{\boldsymbol{\ell}} \in \{0,1\}^n$		
$t^\ell = \langle \boldsymbol{w}, \boldsymbol{m}^{\boldsymbol{\ell}} \rangle + \langle \boldsymbol{s}, \boldsymbol{r}^{\boldsymbol{\ell}} \rangle \in \mathbb{Z}_q$		t^ℓ for $1 \le \ell \le L$
$\boldsymbol{x} \circ \boldsymbol{y} = \boldsymbol{z}$ as described in Subsection 4.6.6		$S, \boldsymbol{w}, \boldsymbol{s}$
mseed $\xleftarrow{\$} \{0,1\}^\lambda$		
Compute parties' seeds		
$(\mathsf{seed}_1, \rho_1), \ldots, (\mathsf{seed}_N, \rho_N)$ with TreePRG(mseed)		
For each party $i \in [1, N]$:		
$[\![\boldsymbol{a}]\!]_i, [\![c]\!]_i, \{[\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_i, [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_i\}_{1 \le \ell \le L} \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$		$\triangleright\ \boldsymbol{a} \in \mathbb{Z}_{q'}^{2Ln},\ c \in \mathbb{Z}_{q'},\ [\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_i, [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_i \in [0, A-1]^n$
$\mathsf{com}_i = \mathrm{Com}(\mathsf{seed}_i; \rho_i)$		
For $1 \le \ell \le L$:		
$\Delta \boldsymbol{m}^{\boldsymbol{\ell}} = \boldsymbol{m}^{\boldsymbol{\ell}} - \sum_i [\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_i$		
$\Delta \boldsymbol{r}^{\boldsymbol{\ell}} = \boldsymbol{r}^{\boldsymbol{\ell}} - \sum_i [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_i$		
$\Delta c = -\langle \boldsymbol{a}, \boldsymbol{y} \rangle - \sum_i [\![c]\!]_i$		
$h = \mathcal{H}_1(\{\Delta \boldsymbol{m}^{\boldsymbol{\ell}}, \Delta \boldsymbol{r}^{\boldsymbol{\ell}}\}_{1 \le \ell \le L}, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$		
	$\xrightarrow{h}$	
		$\boldsymbol{\varepsilon} \xleftarrow{\$} \mathbb{Z}_{q'}^{2Ln}, \{\lambda_i\}_{1 \le i \le L+K} \xleftarrow{\$} \mathbb{Z}_{q'}$
	$\xleftarrow{\varepsilon}$	
The parties locally set		
- $[\![t^\ell]\!] = \langle \boldsymbol{w}, [\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!] \rangle + \langle \boldsymbol{s}, [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!] \rangle$ for $\ell \in [1, L]$		
- $[\![\boldsymbol{\alpha}]\!] = \boldsymbol{\varepsilon} \circ [\![\boldsymbol{x}]\!] + [\![\boldsymbol{a}]\!]$		$\triangleright\ \boldsymbol{\alpha} \in \mathbb{Z}_{q'}^{2Ln}$
The parties open $[\![\boldsymbol{\alpha}]\!]$ to get $\boldsymbol{\alpha}$.		
The parties locally set		
$[\![v]\!] = \langle \boldsymbol{\alpha}, [\![\boldsymbol{y}]\!] \rangle - [\![c]\!] - \langle \boldsymbol{\varepsilon}, \boldsymbol{z} \rangle$		$\triangleright\ v \in \mathbb{Z}_{q'}$
$h' = \mathcal{H}_2(\{[\![t^\ell]\!]\}_{1 \le \ell \le L}, [\![\boldsymbol{\alpha}]\!], [\![v]\!])$		
	$\xrightarrow{h'}$	
		$i^* \xleftarrow{\$} [1, N]$
	$\xleftarrow{i^*}$	
If $\exists\, \ell \in [1, L]$ and $j \in [1, n]$ such that:		
- either $[\![m_j^\ell]\!]_{i^*} = 0$ with $m_j^\ell = 1$		
- or $[\![m_j^\ell]\!]_{i^*} = A - 1$ with $m_j^\ell = 0$,		
- or $[\![r_j^\ell]\!]_{i^*} = 0$ with $r_j^\ell = 1$		
- or $[\![r_j^\ell]\!]_{i^*} = A - 1$ with $r_j^\ell = 0$,		
then abort.		
$\boldsymbol{y}_{\boldsymbol{m}^{\boldsymbol{\ell}}} = \boldsymbol{m}^{\boldsymbol{\ell}} - [\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_{i^*}$ and $\boldsymbol{y}_{\boldsymbol{r}^{\boldsymbol{\ell}}} = \boldsymbol{r}^{\boldsymbol{\ell}} - [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_{i^*}$ for $\ell \in [1, L]$		
	$(\mathsf{seed}_i, \rho_i)_{i \ne i^*},\ \mathsf{com}_{i^*},$ $\{\boldsymbol{y}_{\boldsymbol{m}^{\boldsymbol{\ell}}}, \boldsymbol{y}_{\boldsymbol{r}^{\boldsymbol{\ell}}}\}_{1 \le \ell \le L},\ \Delta c,\ [\![\boldsymbol{\alpha}]\!]_{i^*}$ $\xrightarrow{\qquad\qquad\qquad\qquad\qquad\qquad}$	
		For all $i \ne i^*$:
		$[\![\boldsymbol{a}]\!]_i, [\![c]\!]_i, \{[\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_i, [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_i\}_{1 \le \ell \le L} \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$
		Rerun the party i as the prover and compute com_i.
		For $\ell \in [1, L]$:
		$\Delta \boldsymbol{m}^{\boldsymbol{\ell}} = \boldsymbol{y}_{\boldsymbol{m}^{\boldsymbol{\ell}}} - \sum_{i \ne i^*} [\![\boldsymbol{m}^{\boldsymbol{\ell}}]\!]_i$
		$\Delta \boldsymbol{r}^{\boldsymbol{\ell}} = \boldsymbol{y}_{\boldsymbol{r}^{\boldsymbol{\ell}}} - \sum_{i \ne i^*} [\![\boldsymbol{r}^{\boldsymbol{\ell}}]\!]_i$
		$\Delta t^k = \langle \boldsymbol{w}, \Delta \boldsymbol{m}^{\boldsymbol{\ell}} \rangle + \langle \boldsymbol{s}, \Delta \boldsymbol{r}^{\boldsymbol{\ell}} \rangle$
		$[\![t^\ell]\!]_{i^*} = t^\ell - \Delta t^\ell - \sum_{i \ne i^*} [\![t^\ell]\!]_i$
		$\Delta v = \langle \boldsymbol{\alpha}, \Delta \boldsymbol{x} \rangle - \Delta c - \langle \boldsymbol{\varepsilon}, \Delta \boldsymbol{z} \rangle$
		$[\![v]\!]_{i^*} = -\Delta v - \sum_{i \ne i^*} [\![v]\!]_i$
		Check $h = \mathcal{H}_1(\{\Delta \boldsymbol{m}^{\boldsymbol{\ell}}, \Delta \boldsymbol{r}^{\boldsymbol{\ell}}\}_{1 \le \ell \le L}, \Delta c, \mathsf{com}_1, \ldots, \mathsf{com}_N)$
		Check $h' = \mathcal{H}_2(\{[\![t^\ell]\!]\}_{1 \le \ell \le L}, [\![\boldsymbol{\alpha}]\!], [\![v]\!])$
		Return 1

Protocol 17: Zero-knowledge argument for arbitrary AND gates



Protocol 18: Zero-knowledge argument for circuit satisfiability.

E. DESCRIPTION OF THE ACCESS STRUCTURE FOR OUR TZKP

Essentially, secret sharing schemes can be classified according to their *hierarchical access structure*. Obviously, any user with a secret share from an additive/multiplicative/Shamir LSSS is at the same hierarchy level as any other user. However, we could consider LSSS with a more involved access structure, and this notion of *hierarchical access structure* aims to embrace scenarios with a hierarchy in the set of users.

An access structure on a finite set $\mathcal{U}$ of users is a monotone increasing family $\Gamma \subseteq 2^{\mathcal{U}}$ (which denotes all the possible subsets of $\mathcal{U}$), meaning that if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$ and $\mathcal{A} \in \Gamma$, then $\mathcal{B} \in \Gamma$.

Definition 20 (Hierarchical access structures)**.** *Let Γ be an access structure on $\mathcal{U}$. We say that the user $u \in \mathcal{U}$ is hierarchically superior to the user $v \in \mathcal{U}$, and we write $v \preceq u$, if for every subset $\mathcal{A} \subseteq \mathcal{U} \setminus \{u, v\}$ with $\mathcal{A} \cup \{v\} \in \Gamma$, we have $\mathcal{A} \cup \{u\} \in \Gamma$. An access structure is said to be hierarchical if all users are hierarchically comparable (i.e., for every couple of users $\{u, v\} \in \mathcal{U}$, either $v \preceq u$ or $u \preceq v$).*

In the literature, these kinds of schemes are often based on multivariate Lagrange interpolation or Birkoff interpolation [PS04, TD09], to achieve weighted threshold access structures or multilevel access structures.

Hierarchical access structure for our black-box construction of Section 6.3. To provide a clearer picture of the construction, we give its access structure. The two layers of MPC naturally define two levels of hierarchy. Any prover is hierarchically superior to any party, and we adapt the general definitions of hierarchical threshold access structures to our context. In particular, it is captured with the following hierarchical access structure:

$$\Gamma = \{\mathcal{A} \subset \mathcal{U} : |\{\mathbb{1}_{|\mathcal{A} \cap \mathcal{H}_i| > N-1}\}_{i \in [1,k]}| > t\} \tag{E.1}$$

where $\mathcal{U}$ denotes the set of the Nk parties (N parties emulated by each of the k users) with the rating abuse $\mathsf{P}_i \in \mathcal{U} \iff \{\mathsf{P}_{i,\ell}\}_{\ell \in [1,N]} \subset \mathcal{U}$, and $\{\mathcal{H}_1, \ldots, \mathcal{H}_k\}$ denotes the k sets of parties (one for each user). Then, given this access structure, for any subset $\mathcal{A}$ of the Nk parties,

$$\mathcal{A} \subset \Gamma \iff \text{the parties in } \mathcal{A} \text{ can recover the secret.}$$

We have chosen the threshold $(N-1, N)$ for the parties in the head of provers, but the previous access structure can be generalized to other thresholds (*e.g.* those for Shamir secret sharing).

F. VERIFIABLE SECRET SHARING SCHEME

We consider the optimized 5-round version [GIKR01], the original 7 rounds version have been introduced with the BGW protocol [BGW88]. It has communication complexity $O(n^2 \log |\mathbb{F}|(1+bc))$, where bc denotes the communication complexity over the broadcast channel. Then, several other schemes have been proposed based on Shamir's secret sharing scheme achieving better round complexity [GIKR01, KKK09, AKP20]. But none of them achieves strictly better asymptotic communication complexity.

We present the next VSS protocol in the context of Section 6.4. Eventually, we can adapt this protocol (with heavy notations) so that the output of each user P_i is $(\!|[\![s]\!]_i|\!)$, where $(\!|\cdot|\!)$ denotes the sharing scheme in the head of provers.

Verifiable secret sharing protocol
Input: The dealer D holds $s \in \mathbb{F}$
Output: Each user $\mathsf{P}_i \in \{\mathsf{P}_1, \ldots, \mathsf{P}_k\}$ gets $[\![s]\!]_i$, where $[\![\cdot]\!]$ denotes the Shamir secret sharing of degree t.

- **Round I:**
  The dealer D generates a random bivariate polynomial $S(x, y) \in \mathbb{F}[x, y]$ of degree t in both variables such that $S(0, 0) = s$. For every $i \in [1, k]$, D sends to each user P_i $S(0, i)$, $F_i(x) := S(x, i)$ and $G_i(x) := S(i, y)$.

- **Round II:**
  For every $j \in [1, k]$, P_i sends $\{F_i(j), G_i(j)\}$ to P_j.

- **Round III:**
  For every $j \in [1, k]$, let $\{u_{j,i}, v_{j,i}\}$ denote the values received by P_i from P_j in Round II. Prover P_i computes $u_i = (G_i(1) - u_{1,i}, \ldots, G_i(k) - u_{k,i})$, and $v_i = (F_i(1) - v_{1,i}, \ldots, F_i(k) - v_{k,i})$, and broadcasts it. For each $j \in [1, k]$, if the j-th component of u_i or of v_i is not equal to 0, then P_i broadcasts a complaint $\textsc{complaint}(i, j, F_i(j), G_i(j))$. If no user broadcasts a COMPLAINT, then every user P_i outputs $F_i(0)$ and halts.

- **Round IV:**
  For every $\textsc{complaint}(i, j, u, v)$ broadcast by P_i, either $u = S(j, i)$ and $v = S(i, j)$, or D reveals $(i, F_i(x), G_i(y))$.

- **Round V:**

  For every $j \neq k$, user P_i marks (j, k) as a JOINT COMPLAINT if it viewed $\textsc{complaint}(k, j, u_1, v_1)$ and $\textsc{complaint}(j, k, u_2, v_2)$ broadcast by P_k and P_j, such that $u_1 \neq v_2$ or $v_1 \neq u_2$. If there exists a joint complaint (j, k) for which the dealer did not broadcast $(j, F_j(x), G_j(y))$ nor $(k, F_k(x), G_k(y))$, then go to output decision step (and do not broadcast CONSISTENT). Otherwise:

  Consider the set of revealed $(j, F_j(x), G_j(y))$ messages sent by D. If there exists a message in the set with $j = i$ then reset the polynomials $F_i(x)$ and $G_i(y)$ to the new polynomials that were received, and go to the output decision step (without broadcasting CONSISTENT). If there exists a message in the set with $j \neq i$ and for which $F_i(j) \neq G_j(i)$ or $G_i(j) \neq F_j(i)$, then go to output decision step (without broadcasting CONSISTENT). If the set of reveal messages does not contain a message that fulfills either one of the above conditions, then proceed to the next step.

  Broadcast the message CONSISTENT.

- **Output decision:** If at least $k-t$ users broadcast CONSISTENT, P_i outputs $F_i(0)$. Otherwise, P_i outputs $\perp$.

Fig. F.1: Verifiable secret sharing protocol

Abstract

This thesis aims to study zero-knowledge arguments, a cryptographic primitive that allows to prove a statement while yielding nothing beyond its truth (we may call it proof instead of argument depending on the security model). Specifically, we focus on a family of arguments whose construction is based on secure multiparty computation. It is well-known that, given any functionality, there exists a secure multiparty protocol computing it. Let us take a generic one-way function f, and a secure multiparty protocol computing f, then it has been shown seventeen years ago that we can build a zero-knowledge argument for the $\mathcal{NP}$-problem of finding a pre-image of f. This construction was considered only theoretical until a few years ago, and this thesis contributes to the emergence of new techniques as well as efficient applications.

As an appetizer, we develop simple zero-knowledge protocols that significantly improve the state-of-the-art communication complexity for some well-known problems. Our first substantial contribution, with a desire to share small elements over large fields, is the introduction of a sharing over the integers that is securely embedded in our protocols with some artificial abortion. Applications are manifold, eventually in the post-quantum regime. In the line with our sharing over the integers, we propose a cryptographic string commitment scheme based on subset sum problems. In particular, it enables efficient arguments for circuit satisfiability. Then, we present a proof construction employing conversion between additive and multiplicative secret sharings, leading to efficient proofs of linear and multiplicative relations. The applications are again manifold when designing arguments and digital signatures. Finally, leaving aside protocols conception, we explore cryptography foundations with multi-prover zero-knowledge proofs, a framework for distributing the prover's computation of interactive zero-knowledge proofs. To capture the full interest of this dispatching, we add to the literature a fundamental result for threshold zero-knowledge proofs for generic $\mathcal{NP}$-statement.