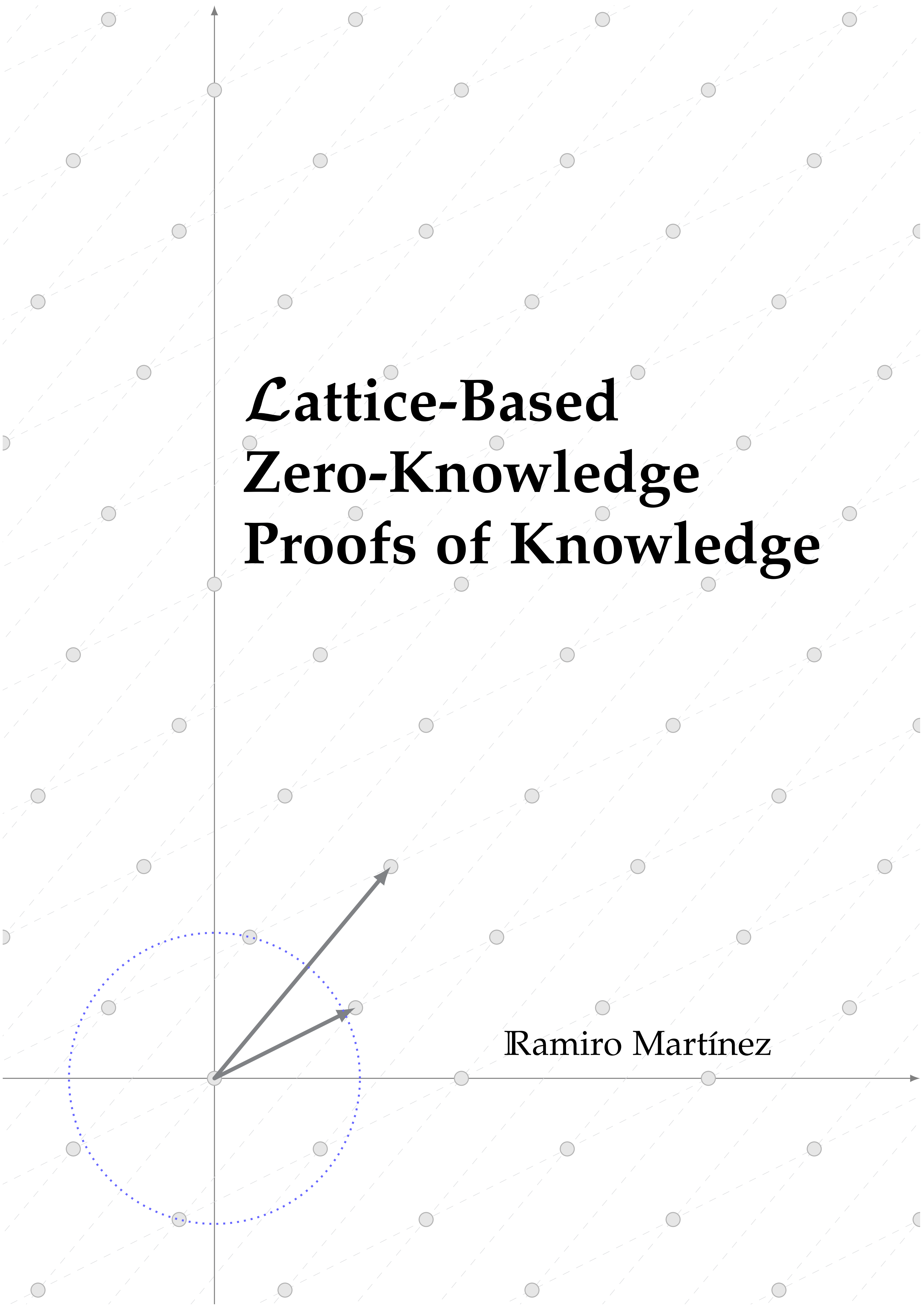


$\mathcal{L}$attice-Based Zero-Knowledge Proofs of Knowledge

$\mathbb{R}$amiro Martínez

Lattice-Based Zero-Knowledge Proofs of Knowledge

A thesis submitted to the Universitat Politècnica de Catalunya
for the degree of Doctor of Philosophy

by

Ramiro Martínez Pinilla

Paz Morillo, Advisor

Department of Mathematics

July 2023

Abstract

The main goal of this dissertation is to develop new lattice-based cryptographic schemes. Most of the cryptographic protocols that each and every one of us use on a daily basis are only secure under the assumption that two mathematical problems, namely the discrete logarithm on elliptic curves and the factorization of products of two primes, are computationally hard. That is believed to be true for classical computers, but quantum computers would be able to solve these problems much more efficiently, demolishing the foundations of plenty of cryptographic constructions. This reveals the importance of post-quantum alternatives, cryptographic schemes whose security relies on different problems intractable for both classical and quantum computers. The most promising family of problems widely believed to be hard for quantum computers are lattice-based problems.

We increase the supply of lattice-based tools providing new Zero-Knowledge Proofs of Knowledge for the Ring Learning With Errors (RLWE) problem, perhaps the most popular lattice-based problem. Zero-knowledge proofs are protocols between a prover and a verifier where the prover convinces the verifier of the validity of certain statements without revealing any additional relevant information. Our proofs extend the literature of Stern-based proofs, following the techniques presented by Jacques Stern in 1994. His original idea involved a code-based problem, but it has been reiteratedly improved and generalized to be used with lattices. We illustrate our proposal defining a variant of the commitment scheme, a cryptographic primitive that allows us to ensure some message was already determined at some point without revealing it until a future time, defined by Benhamouda *et al.* in ESORICS 2015, and proving in zero-knowledge the knowledge of a valid opening. Most importantly we also show how to prove that the message committed in one commitment is a linear combination, with some public coefficients, of the committed messages from two other commitments, again without revealing any further information about the messages. Finally, we also present a zero-knowledge proof analogous to the previous one but for multiplicative relations, something much more involved that allows us to prove any arithmetic circuit. We give first an interactive version of these proofs and then show how to construct a non-interactive one.

We diligently prove that both the commitment and the companion Zero-Knowledge Proofs of Knowledge are secure under the assumption of the hardness of the underlying lattice problems. Furthermore, we specifically develop such proofs so that the arising conditions can be directly used to compute parameters that satisfy

them. This way we provide a general method to instantiate our commitment and proofs with any desired security level. Thanks to this practical approach we have been able to implement all the proposed schemes and benchmark the prototype implementation with actually secure parameters, which allows us to obtain meaningful results and compare its performance with the existing alternatives.

Moreover, provided that multiplication of polynomials in the quotient ring $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$, with p prime and n a power of two, is the most basic operation when working with ideal lattices we comprehensively study what are the necessary and sufficient conditions needed for applying (a generalized version of) the Fast Fourier Transform (FFT) to obtain an efficient multiplication algorithm in quotient rings as $\mathbb{Z}_m[x]/\langle x^n - a\rangle$ (where we consider any positive integer m and generalize the quotient), as we think it is of independent interest. We believe such a theoretical analysis is fundamental to be able to determine when a given generalization can also be applied to design an efficient multiplication algorithm when the FFT is not defined for the ring we are considering. That is the case of the rings used for the commitment and proofs described before, where only a partial FFT is available.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

We start this chapter with Section 1.1, briefly introducing all the concepts behind this work, starting with *cryptography* itself from a historical point of view to be able to define the paradigmatic ideas of *public key cryptography* and *provable security*, in order to motivate the need of new *post-quantum* cryptographic constructions with *concrete security*. Then, all the cryptographic primitives that we are going to use through the dissertation are going to be formally defined in Section 1.2. Additionally, Section 1.3 is going to be devoted to lattices as a mathematical object and the related problems we can use as hardness assumptions for cryptographic constructions. In this introductory chapter we also prove from the very beginning in Section 1.4 some probability lemmas that are going to be useful for simplifying the proofs of security of the cryptographic proposals we are going to latter present. Finally, in Section 1.5 we introduce the main contributions that we are going to present through Chapters 2 to 4.

1.1 Motivation

Humanity has been aiming to keep conversations private for centuries. This goal has been so closely related to writing itself that we can even trace it back to clay tablets with cuneiform writing from the ancient Mesopotamia. One of the first preserved examples is the baked clay tablet now in the collection of the British Museum with number BM 120960, transcribed and translated by Gadd and Thompson in [56]. It was originally found in Tall 'Umar, at the city of Seleucia on the Tigris (current Iraq), and has been dated around the seventeenth century BCE. This tablet is exceptionally unique because it describes the first known recipe for glazed pottery, but, according to Gadd and Thompson, does so with an "artificial obscurity of expression" by means of "artifices of writing which amount to a form of cryptography". The authors of [56] claim that this writing style was specifically intended to conceal the secrets of the recipe because common words as verbs are written straightforwardly while the relevant parts are elaborately “disguised”. The fact that the first civilization for which we have evidence of written language also developed some kind of cryptography exemplifies its ubiquity through history.

However, for a long time cryptography was just a matter of *cryptographers* designing increasingly intricate methods to transform the plain messages, *plaintexts*, into encrypted messages, *ciphertexts*, hoping no smarter *cryptanalyst* was able to hack their techniques. Perhaps the most paradigmatic case is the successful effort from Alan Turing and his team on Bletchley Park cracking the Nazi ciphertexts encrypted with the Enigma machine during the World War II (*Turing's Treatise on the Enigma* is publicly available thanks to the Turing Digital Archive*).

Only during the second half of the last century, when computer science was being born thanks to the formalization of Turing himself and his advisor Alonzo Church, and notions such as computational complexity started to be taken into serious consideration, *modern cryptography* was born embracing the *provable security* paradigm. Applied mathematics intend to address real-life problems by means of reducing them to abstract mathematical problems we already know how to solve. That means the real-life problem is easier because anyone who knows how to solve the abstract problem can also solve the real-life problem by translating it into the former and applying the known method. On the contrary, cryptographers do not want to show that some problems are easy but to ensure some other problems, as decrypting a message or forging a digital signature without knowing the secret key, are indeed really difficult to solve. To do so modern cryptography just follows the reciprocal approach. Encryption and signature schemes are designed with some mathematical problems in mind, involving the same kind of operations, so that breaking its security implies being able to solve the abstract mathematical problem.

We reiterate this foundational idea to develop the discussion on its potential. The point is not that if we knew how to solve a mathematical problem then we could use it to break the security of the cryptographic scheme but we choose to give up because such problem is really hard. If that was just the case we could never have a certain reference for the security level, as there might be a different more clever approach to break it without involving that problem. The good news is that schemes are designed so that reductions work the other way around. If someone devised an efficient method to break its security (forge a signature or decrypt a message addressed to someone else), we know how we could use this method to solve the original mathematical problem we believe to be really hard. That means that, under

*`https://turingarchive.kings.cam.ac.uk/amtc/amt-c-30`

certain assumptions, namely the hardness of the underlying mathematical problems, we can formally prove the security of the cryptographic schemes, defining the realm of provable security "changing this ancient art into a science" [44].

We remark here that these proofs have the computational assumptions as hypothesis, because actually proving the hardness of relevant problems seems a difficult endeavor, exemplified by the fact that we cannot yet prove or disprove the equality of P and NP computational classes (*decisional* problems, with yes or no answers, that we can solve in polynomial time with a Turing Machine or a Non-Deterministic Turing Machine respectively).

Nowadays, this has become a significantly more important matter, provided that our entire lives have been digitalized and personal, professional or financial interactions are conducted over the internet and secured with cryptography. As we have already introduced, mentioning signature schemes, cryptography is much more than concealing secrets through encryption. It also deals with text integrity, authentication and anything communication related that involves some kind of (lack of) *trust*. There are plenty of cryptographic primitives and each needs its formal security models.

The *public key cryptography* paradigm, also called asymmetric cryptography, involves the existence of a secret key with a related public key, so that the secret key belongs to the user and is used for private operations such as decrypting or signing while the public key can be used by anyone for encrypting or verifying a signature. In this paradigm security is only computational. The only alternative to obtain perfect security, from an information-theoretical point of view, is the one-time pad [108], but requires exchanging non-reusable secrets as large as the messages.

The point is to design schemes so that the computational complexity of the attacks escalates much worse than the computational complexity of the honest players. Of course an adversary could always guess the secret key just trying every possible combination of characters, but we can rule out this brute-force *key recovery attack* ensuring the key space is large enough. We can do so if increasing the key size one bit only adds a constant time to the honest encryption or decryption computations, because this increase is perfectly admissible, while doubling the number of keys implies doubling too the brute-force time as the attacker now has twice as many keys to try.

In fact, it is sufficient if the honest computations (encryption and decryption using the key) depend *polynomially* on the size of the keys. As we have been mentioning the big step comes when this approach works not only against this particular attack but against any attempt to break security (recovering or not the keys) because the best known algorithms for solving the underlying mathematical problem (that we know establishes a lower bound on the time required to successfully misbehave) also take an exponential (or at least *superpolynomial*) amount of time on the size of the keys. Then we can use an argument equivalent to the above, as we are going to latter formalize in Section 1.2.

The fact that in the previous example the malicious approach scales exponentially worse than the honest executions means we can hopefully select reasonable parameters that make any attack completely infeasible. Let us grasp how fast is the exponential growth of powers of 2 that we obtain doubling the number of keys with each additional bit. To get an idea of the magnitude of current standard security levels such as 128 security bits, meaning a successful attack would need 2^{128} operations, we just have to mention that the current age of the universe in seconds is less than 2^{59} and according to the TOP500 list* the most powerful supercomputer can now perform 2^{60} floating point operations per second, which amount for a total of 2^{119} if it started running at the Big Bang. That is, it would have not been enough to break a cryptographic scheme with 128 security bits.

Most of the cryptographic standards currently in use rely on the hardness of two well studied problems, the *Discrete Logarithm* (DLog) problem and prime factorization, or related assumptions as *Decisional Diffie-Hellman* (DDH) [26] or *Rivest-Shamir-Adelman* (RSA) [102]. Computing a discrete logarithm on certain groups or factorizing products of large primes are difficult problems for which no efficient algorithm is currently known, and this allows the existence of a myriad of cryptographic applications. But, unfortunately, that statement is only true if we limit ourselves to *classical algorithms*, excluding *quantum algorithms*.

Everyday computers work using binary logic, having a *bit* as the most basic unit of information storing two possible states usually represented as a 0 or a 1. However, taking advantage of the quantum nature of this universe, a completely different model of computation called *quantum computing* can be built having *qubits*, the quantum analogous of a bit, as basic units. Leveraging that quantum mechanics allows a particle to be in superposition of different states a qubit is allowed to be in a superposition of two basic states. Mathematically speaking a qubit is a unitary linear combination of the two states with complex coefficients. When considering more than one qubit (in a Hilbert space of 2^N dimensions) this allows special states, *entangled states*, that are not the product of individual classical states, and from this emerge new possibilities for computation.

It is important to remark that these new operations that are possible with a quantum computer do not add anything new in terms of *computability*. We say a decisional problem is *decidable* if there exists an algorithm guaranteed to compute the

*`https://top500.org`

answer in a finite amount of steps. Every problem decidable with a Quantum Turing Machine (the theoretical model for a quantum computer) is also decidable with a classical Turing Machine (the theoretical model for classical computers). That is the case because a quantum computer can be fully emulated with a classical computer. This means that the Church-Turing thesis (the conjecture stating that any function defined over the natural numbers can be computed by any effective method if and only if it is computable by a Turing Machine, implying that it is the right model for computation) also applies to quantum computation.

The relevant difference comes when we take into account the time-complexity of the algorithms. The possibility of emulating the quantum computations with a classical computer that we have already mentioned comes, as far as we know, at the expense of an exponential slow down (i.e. requiring exponentially many more computations). We already have examples of problems solvable in *polynomial-time* (the amount of operations can be bounded as a polynomial evaluated on the size of the input) with a quantum computer that have no known classical polynomial algorithm. Unfortunately, two of these problems are precisely DLog and integer factorization, because an efficient quantum algorithm for solving these two problems was developed in 1994 by Peter Shor in [109], making most of the cryptographic primitives we still use today quantum vulnerable.

There is a passionate discussion regarding the feasibility of *Cryptographically Relevant Quantum Computers*, that is, quantum computers able to compromise the security of cryptographic primitives based on problems related to the DLog and integer factorization. Even if Shor's algorithm is theoretically efficient we have had a limited success actually building quantum computers able to implement it (it has been recently used to factor $21 = 3 \times 7$ using 5 qubits [112]). It might be the case that the problems regarding scalability involving the number of qubits, the propensity to errors combined with the limited ability to correct them, or the difficulties that arise from keeping the quantum computer running for a prolonged period of time (the quantum states are very delicate and the probability of decoherence, the process from which any disturbance collapses the quantum states into a classical state, increases over time) make it infeasible to build a quantum computer that can be an actual threat to the cryptographic security of the theoretically vulnerable schemes. However, it is a very prolific research field, quantum computers with an increasing number of qubits are presented each year and new methods are developed continuously, for example combining the computational capabilities of a quantum computer and a classical one so that a much less powerful quantum computer already provides an advantage [112].

In any case this represents a colossal risk because even if we cannot precisely answer if or when a quantum computer will be able to solve these problems we do know that the consequences could be chaotic unless we have substituted these vulnerable schemes. Furthermore, this issue should be addressed at the earliest convenience. That is the case because signature schemes based on the RSA assumption will become untrustworthy whenever there is a quantum computer that might have forged them, so we have to transition to a resistant scheme when a powerful enough quantum computer becomes available, but encryption schemes have to be replaced now because we expect the encrypted information to remain private for a long period of time and some malicious entity could be collecting now current communications so that they can decrypt them in the future. We can wait for primitives ensuring the integrity of messages or operations, but we have to act now regarding anything related with privacy.

There are to completely different alternatives to face this challenge. *Quantum Cryptography* encompasses the methods that try to base the security of new schemes on the quantum mechanical properties of nature. Quantum random generators and quantum key distribution schemes have been proposed using the non-deterministic nature of quantum states for the former and the properties of quantum measurements for the latter, as these properties allow two users to detect if a third party has measured the individual photons (or whatever quantum particle is used) they have been exchanging and therefore can share a key knowing no eavesdropper has been involved.

The previous approach is then based on a completely different paradigm, and requires specialized expensive hardware to be implemented. The alternative that we follow through this dissertation, called *Post-Quantum Cryptography*, is to keep the approach basing the security on the computational hardness of mathematical problems but using alternative problems that are believed to be hard for both classical and quantum computers. It might be the case that a quantum computer is more efficient than a classical one, but we are fine if the difference between the number of classical computations performed by the honest parties is still significantly smaller than the infeasible amount of computation that would be required for a malicious adversary in order to break the scheme.

There exist multiple alternative candidates, lattice-based, code-based, hash-based and multivariate-based cryptography, each basing its security on a different family of problems. Lattices are the most promising source of problems for post-quantum cryptography (we are going to extensively discuss their advantages in Section 1.3). The greatest effort to standardize post-quantum cryptographic primitives its being conducted by the National Institute of Standards and Technology (NIST) from the United States of America. A competition was launched in 2016 asking for public key encryption and digital signature proposals[*], and from the many applications submitted by teams from all around the globe, after three rounds of submissions three of the four candidates that have been chosen to be standardized are lattice-based proposals (the key encapsulation mechanism Kyber and the digital signatures Dilithium and Falcon) and one is hash-based (the digital signature SPHINCS$^+$).

Besides the existence of asymptotic formal proofs that guarantee that sufficiently large key sizes would make the scheme secure against quantum computers we also aspire, to the best of our knowledge, to obtain *concrete security* and try to quantitatively bound the success probability of an adversary with a specific amount of resources. We believe this is also important if we intend our research to be trusted and yield to practical applications.

1.2 (Post-Quantum) Cryptography

As we have mentioned, cryptography goes far beyond encryption and signature schemes. We introduce in this section all the primitives that we are going to use as examples or to present our contributions. We also introduce here the notation we are going to use regarding probabilistic algorithms.

Given an algorithm $\mathcal{A}$ we use $a \leftarrow \mathcal{A}$ to denote that a is the output of $\mathcal{A}$. If the algorithm $\mathcal{A}$ is probabilistic and the output is not deterministically determined by the input we might then use $a \leftarrow_{\text{R}} \mathcal{A}$ to indicate that some randomness is involved. Given a set X we also use $x \leftarrow_{\text{R}} X$ to denote that x is sampled uniformly at random from X. Finally, given a probability distribution D we analogously use $d \leftarrow_{\text{R}} D$ to denote that d is sampled following the distribution D.

The usual convention is to assume the time-complexity of an algorithm is reasonable if the number of operations it needs to finish is bounded by a polynomial on the size of its input. Given that we use probabilistic algorithms we want this property to be true independently of the internal random coins. This class of algorithms is usually known as *Probabilistic Polynomial-Time* (PPT). We generally consider the possibility of these algorithms being quantum algorithms and not only classical ones.

We want the success probability of adversaries trying to break the security of our schemes to be insignificant. This can be formalized asymptotically with the following two definitions.

Definition 1.1 (Negligible function).

A function f is *negligible* if $|f(n)| \in O\left(n^{-c}\right)$, $\forall c \in \mathbb{Z}_{>0}$.

We denote it writing $f \in \text{negl}(n)$.

[*]`https://csrc.nist.gov/projects/post-quantum-cryptography`

Definition 1.2 (Overwhelming function)**.**
A function f is *overwhelming* if $|f(n) - 1| \in \mathcal{O}(n^{-c})$, $\forall c \in \mathbb{Z}_{>0}$.

The level of security can then be controlled by the variable of these negligible functions, usually known as *security parameter*. Through this dissertation λ is going to be our security parameter in all security proofs.

We might sometimes want to bound the probabilities of the undesired events by some given probability, usually $2^{-\lambda}$ (and not only asymptotically). If this bound is specifically indicated we might also use the word negligible to refer to this specific value because, in particular, $2^{-\lambda}$ is negligible in λ.

1.2.1 Encryption Schemes

In this work we are going to mainly focus on other than encryption primitives, but we start defining a public key encryption scheme because we are going to use them as examples of applications of lattice-based cryptography, and we need this definition to motivate the other primitives.

Through this section we are going to always include all input and output elements necessary to define the involved algorithms. However, in some cases some of these inputs, for example the public parameters that define the length of the messages in the considered message space, are unambiguously implicitly defined from the context. We use a semicolon instead of a comma to separate this kind of parameters that, in the following sections, we might later omit in order to simplify notation when leaving them out does not introduce any possible ambiguity.

Definition 1.3 (Public Key Encryption Scheme)**.** A *public key encryption scheme*, designed to hide a message so that only the designated receiver can then read it, consists on three efficient algorithms:

- Gen: the generator algorithm takes a security parameter 1^λ and outputs a pair of keys, sk and pk, the secret and the public key respectively. It also defines the message space $\mathcal{M}_\lambda$ and the ciphertext space $\mathcal{C}_\lambda$, via some public parameters pp. These public parameters can be seen as part of the public key, but we prefer to treat them separately.
$$(sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda)$$

- Enc: the encryption algorithm takes as input a message $m \in \mathcal{M}_\lambda$, a public key pk and the public parameters pp and produces a ciphertext $c \in \mathcal{C}_\lambda$.
$$c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m; pk, pp)$$

- Dec: the decryption algorithm takes as input a ciphertext c and a secret key sk, possibly together with the public key pk and the public parameters pp, and outputs a message m'.
$$m' \leftarrow \mathsf{Dec}\big(c, sk; pk, pp\big)$$

These algorithms should satisfy the following two properties:

- **Correctness**: knowing the private key the decryption algorithm should recover the encrypted message.
$$\left.\begin{array}{l} (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \\ m \in \mathcal{M}_\lambda \\ c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m; pk, pp) \\ m' \leftarrow_{\mathrm{R}} \mathsf{Dec}\,(c, sk; pk, pp) \end{array}\right\} \implies m' = m.$$
In some occasions we might be content with this property happening except with negligible probability, allowing the possibility of some errors if they happen with sufficiently low probability.
$$\Pr\left[m' \neq m \;\middle|\; \begin{array}{c} (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ m \in \mathcal{M}_\lambda, \\ c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m; pk, pp), \\ m' \leftarrow \mathsf{Dec}(c, sk; pk, pp) \end{array} \right] \in \mathrm{negl}(\lambda).$$

- **Security**: we also expect the encryption algorithm to hide the message m, that should not be recoverable from the ciphertext c and the public key pk alone, without the secret key sk. Depending on how we formalize this notion, rigorously defining what we want to prevent and specifying the capabilities of the adversary, we might obtain different security definitions, involving the security parameter λ.

This definition ensures that, on the one hand, anyone can encrypt messages, as the algorithms and the public key are, indeed, publicly known. On the other hand, people knowing the secret key can recover, always or with overwhelming probability, the secret message that is encrypted in the ciphertext.

The first intuition about the security of a public key encryption scheme tells us that besides this we also need that the original message should not be efficiently obtainable without knowing the secret key. Notice the key generator algorithm takes 1^λ as input because we measure this difficulty as the efficiency of the algorithms with respect to the length of its input, and this way the input has length λ. The same is going to apply for the adversary, so we can define its computational power as a function of λ (notice also that the size of the output of a polynomial algorithm is necessarily polynomial too). This can be formalized with the following definition.

Definition 1.4 (Public Key Encryption One Way Chosen Plaintext Attack Secure – OW-CPA)**.** A public key encryption scheme is said to be *OW-CPA secure* if for all PPT adversaries $\mathcal{A}$ the following holds.

$$\Pr\left[m' = m \;\middle|\; \begin{array}{c} (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ m \leftarrow_{\mathrm{R}} \mathcal{M}_\lambda,\ c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m; pk, pp), \\ m' \leftarrow_{\mathrm{R}} \mathcal{A}(c, pk, pp, 1^\lambda) \end{array} \right] \in \mathrm{negl}(\lambda).$$

However, this is usually not enough. An encryption scheme that only modifies the second half of the message may be OW-CPA secure if it is hard enough to recover this second half, but reveals a lot of information about the message. For that reason the following standard stronger security definition is usually considered.

Definition 1.5 (Public Key Encryption Indistinguishable Chosen Plaintext Attack Secure – IND-CPA)**.** A public key encryption scheme is said to be *IND-CPA secure* if for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ the following holds.

$$\left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1(pk, pp, 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0, 1\},\ c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m_b; pk, pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(c, pk, pp, aux, 1^\lambda) \end{array} \right] - {}^1\!/_2 \right| \in \mathrm{negl}(\lambda).$$

That is, the encryptions of two different messages should not be computationally distinguishable (the advantage over a random guess has to be negligible), even if we allow the adversary to choose those messages. Note that this definition implies that the algorithm has to be probabilistic.

Observe we have defined two separated algorithms for the adversary, in order to separate its two main tasks. However, an adversary trying to learn some information from the encryption could always use their prior knowledge from the moment selecting the two messages. For this reason we include an auxiliary variable *aux* that stores any information $\mathcal{A}_1$ considers that $\mathcal{A}_2$ could use to guess the message.

We prove security of an encryption scheme under certain hypothesis (the hardness of some computational problem) proving the existence of a reduction that implies that a PPT algorithm breaking the security of the encryption could be used to disprove the hypothesis. In particular, we are interested in public key encryption schemes whose security is based in hypothesis believed to be true in a quantum scenario. Some examples are going to be presented in Section 1.3.

1.2.2 Commitment Schemes

Sometimes we want to hide some information not to restrict who should access it but when should it become public. Just publishing it latter might not be an option if the fact that the decision was already made in a prior moment is important.

Let us consider a simple scenario where a group of people votes to choose an option among several possibilities. There is no privacy need and everyone agrees to make their own decision public. However, disclosing their options one by one would make the process unfair, because as more and more options are revealed the following participants have more information and can choose to change their mind in a strategic way (for example voting for their second preferred option if they already consider their first option has a little chance of winning). That would provide more control over the final outcome to the last participants, making the process biased. Enumerating the options and raising hands has similar issues, this time related with the order of the options. In a physical meeting this could be easily solved by asking the participants to write down their option on a piece of paper, so then each participant can reveal their own choice once all the other votes have already been fixed.

Being able to obtain the same functionality with a cryptographic primitive is a very versatile tool, but requires a different security model than encryption. For example, imaging an IND-CPA secure public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that encrypts a single bit for a yes or no vote. A participant can sample a pair of keys and public parameters $(sk, pk; pp) \leftarrow_{\text{R}} \mathsf{Gen}(1^\lambda)$, encrypt their option $b \in \{0, 1\} = \mathcal{M}_\lambda$ with $c \leftarrow_{\text{R}} \mathsf{Enc}(b; pk, pp)$ and reveal c, because the IND-CPA property ensures no adversary can correctly guess the value of b as encryptions of 0 are indistinguishable from encryptions of 1. However, without any further consideration, this would not be enough to later convince others that their vote has already been fixed. Of course the participant could reveal sk so that anyone can check $b \leftarrow \mathsf{Dec}(c, sk; pk, pp)$. But that would be meaningless because, if the participant wishes to change their mind (and choose the other option $\overline{b}$), they can also sample new sk' so that $\overline{b} \leftarrow \mathsf{Dec}(c, sk'; pk, pp)$ (and this is always possible because if sampling different keys we cannot efficiently find one such that $\overline{b} \leftarrow \mathsf{Dec}(c, sk'; pk, pp)$ then we would be able to know the value of b and break the security of the encryption scheme).

A cryptographic primitive that could solve these problems does not only need to guarantee that it *hides* the secret. Whatever is disclosed should be *bound* to the original message and no other, as the user wants to show that they have *committed* to that value. The primitive we need for this scenario is called a commitment scheme.

Definition 1.6 (Commitment Scheme)**.** A *commitment scheme* is a cryptographic primitive with two phases that allows one party to commit to a value that will only be revealed to other parties showing an *opening* in the future. It consists on three efficient algorithms:

- Gen: the generator algorithm takes as input the security parameter 1^λ and outputs a public key pk and the public parameters pp that define the commitment and message spaces and the specifications of the other two algorithms.

$$(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}\left(1^\lambda\right)$$

- Com: the commitment algorithm takes as input a message m and a public key pk, together with the public parameters pp, and outputs both a commitment c and an opening o.

$$(c,o) \leftarrow_{\mathrm{R}} \mathsf{Com}\left(m;pk,pp\right)$$

- Ver: the verification algorithm takes as input a commitment c, a message m, an opening o and the public key pk with the public parameters pp, and accepts or rejects that opening.

$$\mathsf{Ver} : \left\{(c,m,o;pk,pp)\right\} \rightarrow \left\{\text{accept}, \text{reject}\right\}$$

These algorithms have to satisfy the following three properties:

- **Correctness**: if the commitment has been built correctly and the original message and the opening are published then the verification algorithm always accepts.

$$\left.\begin{array}{l}(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}\left(1^\lambda\right)\\ (c,o) \leftarrow_{\mathrm{R}} \mathsf{Com}\left(m;pk,pp\right)\end{array}\right\} \Longrightarrow \text{accept} \leftarrow \mathsf{Ver}(c,m,o;pk,pp).$$

  Analogously as before we might accept correctness except with negligible probability for every message.

$$\Pr\left[\text{reject} \leftarrow \mathsf{Ver}(c,m,o;pk,pp) \;\middle|\; \begin{array}{c}(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda)\\ (c,o) \leftarrow_{\mathrm{R}} \mathsf{Com}(m;pk,pp)\end{array}\right] \in \text{negl}(\lambda).$$

- **Hiding**: a well constructed commitment c does not leak any relevant information about the message m. This property can be perfect or only computational.

  – **Perfectly Hiding**: it is perfectly hiding if this property is absolute and unconditional, meaning that for any commitment to a message there exists a valid opening to any other message.

$$\left.\begin{array}{l}(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \\ (c,o) \leftarrow_{\mathrm{R}} \mathsf{Com}(m;pk,pp)\end{array}\right\} \implies \begin{array}{c}\forall m' \; \exists o' \text{ such that} \\ \text{accept} \leftarrow \mathsf{Ver}(c,m',o';pk,pp).\end{array}$$

– **Computationally Hiding**: for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ the advantage distinguishing two commitments, even if the messages are chosen by the adversary themself, is negligible compared with a random guess.

$$\left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c}(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1(pk,pp,1^\lambda) \\ b \leftarrow_{\mathrm{R}} \{0,1\},\; (c,o) \leftarrow_{\mathrm{R}} \mathsf{Com}(m_b;pk,pp) \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(c,pk,pp,aux,1^\lambda)\end{array} \right] - 1/2 \right| \in \mathrm{negl}(\lambda).$$

- **Binding**: it should be infeasible for the sender to output a commitment that can be opened to two different values. This property can again be perfect or computational.

  – **Perfectly Binding**: a commitment can only be opened to one message.

$$\left.\begin{array}{l}\text{accept} \leftarrow \mathsf{Ver}\,(c,m,o;pk,pp) \\ \text{accept} \leftarrow \mathsf{Ver}\,(c,m',o';pk,pp)\end{array}\right\} \implies m = m'.$$

We might admit again the possibility of this property holding except with negligible probability, in this case in the selection of the public key.

$$\Pr\left[\left.\begin{array}{l}\mathsf{Ver}\,(c,m,o;pk,pp) \\ \mathsf{Ver}\,(c,m',o';pk,pp)\end{array}\right\} \not\Longrightarrow m = m' \;\middle|\; (pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \right] \in \mathrm{negl}(\lambda).$$

– **Computationally Binding**: no PPT adversary $\mathcal{A}$ can output a commitment c and two valid openings o and o' to two different messages $m \neq m'$ except with negligible probability.

$$\Pr\left[\begin{array}{c}\mathsf{Ver}(c,m,o;pk,pp), \\ \mathsf{Ver}(c,m',o';pk,pp), \\ m \neq m'\end{array} \;\middle|\; \begin{array}{c}(pk;pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ (c,m,m',o,o') \leftarrow_{\mathrm{R}} \mathcal{A}(pk,pp,1^\lambda)\end{array} \right] \in \mathrm{negl}(\lambda).$$

Observe the hiding and binding properties cannot be both perfect for the same scheme. In this work we are always going to consider computationally hiding commitment schemes, either perfect or computationally binding.

We sometimes abuse notation to include the randomness of the commitment algorithm as another input, writing $(c, o) \leftarrow \mathsf{Com}(m, r; pk, pp)$ to denote that we have used randomness r. Alternatively, if everything else is clear from the context, we might further simplify the notation and just write $c \leftarrow_{\mathrm{R}} \mathsf{Com}(m)$. In some occasions we also simply call opening to the pair of message and opening (m, o).

We identify the possible outputs of the verification algorithm accept and reject with true and false. Then we might say that "$\mathsf{Ver}(c, m, o; pk, pp)$" to indicate that the output is accept (as we have done when describing the binding property), or write "not $\mathsf{Ver}(c, m, o; pk, pp)$" to indicate that the output is reject. When someone has to verify that the output is accepting we use the notation $\overset{?}{\mathsf{Ver}}(c, m, o; pk, pp)$.

This primitive allows the realization of many interesting protocols. It can be used just as a way of preserving anonymity. For example, a commitment to the identity of the author can be used as a pseudonym in a blindly reviewed literature context so that only the winner has to reveal themself, and the same strategy can apply in a call for tender. But it is far more useful for occasions where we need to ensure that several participants that do not trust each other cannot make strategic decisions because they have prior knowledge of the choices of the others. That would be the case on a sealed bid auction, where all bids are sent in advance. Using a commitment scheme would allow the highest bidder to reveal their offer (the organizer would start descending from a maximum value until one of the participants opens their commitment to the current value and wins the auction), while keeping the rest of bids completely secret. As a classical example it is usually said that commitment schemes allows one to play rock/paper/scissors over the internet (again none of the players should know the rival's option before choosing their own, and it can be solved exchanging first a commitment to your option and then sharing valid openings).

This last example seems just a game, but randomly choosing an option among many participants that do not trust each other is an important building block of multiparty computation. Two-party coin tossing involves two players trying to decide a binary output (heads or tails with the coin metaphor) without trusting the other participant's coin. In the physical world this can be solved allowing both of the participants to toss their own coin each, previously determining that we proceed with one option if both coins have the same outcome (two heads or two tails) or the other otherwise. This approach guarantees that, as long as one of the coins is unbiased, both options have the same probability, so each participant has to trust only their own coin. However, the procedure could be gamed if one sees the outcome of the other first and changes their coin accordingly. To prevent that, commitment schemes can be used again.

With more advanced protocols we constantly have to consider this kind of events.

When we talk about the difficulty of a problem we refer to the difficulty of solving a random instance of that problem, and that might need different elements to be selected independently by different parties. An adversary knowing one of the elements beforehand could use that information to maliciously choose their own contribution so that the statement is no longer uniformly distributed and the problem is no longer hard for such adversary. This type of issues can again be trivially solved using commitments as a fundamental piece of interactive protocols.

1.2.3 Interactive Proof Systems

Beyond concealing information, cryptography offers solutions to verify that some procedures have been conducted properly, providing evidence that can be validated by everyone.

Consider a cloud provider that solves computationally expensive problems as a service. It could be directly used if the user is able to verify that the answer they receive solves indeed the initial problem. However, if verifying the fact that the answer is indeed a solution is itself a computational task too expensive for the constrained device of the final user, we need additional tools in order to avoid just trusting the company.

Mathematically speaking we formalize this saying that a *prover*, usually denoted by $\mathcal{P}$, wants to convince a verifier, usually denoted by $\mathcal{V}$, that some statement x belongs to a formal language of strings L (a set of strings over some alphabet, encoding some class of statements that we call valid).

Definition 1.7 (Interactive Proof System)**.** An *Interactive Proof System* is a two-party protocol between a (potentially computationally unlimited) prover $\mathcal{P}$ and a verifier $\mathcal{V}$ where the prover tries to convince the verifier of the fact that $x \in L$ sequentially exchanging messages until the verifier makes a decision and accepts or rejects depending on the conversation.

We denote this interactive proof by $\langle \mathcal{P}, \mathcal{V} \rangle$ and define the decision finally taken by the verifier when the protocol is executed for some statement x as $\langle \mathcal{P}, \mathcal{V} \rangle(x) \in \{\text{accept}, \text{reject}\}$.

It satisfies two properties:

- **Completeness**: if both the prover and the verifier follow the protocol for an $x \in L$ then the verifier always accepts.

$$x \in L \implies \langle \mathcal{P}, \mathcal{V} \rangle(x) = \text{accept}.$$

- **Soundness**: if $x \notin L$ then the verifier rejects (except with some small probability

ϵ, that we call the *soundness error* of the protocol).

$$\Pr\left[\langle \mathcal{P}, \mathcal{V} \rangle(x) = \text{accept} \,\middle|\, x \notin L\right] \leq \epsilon.$$

If we also allow completeness to hold except with some small probability then the class of languages for which there exists an interactive proof system is the computational class IP, which has theoretical interest on its own.

1.2.4 Zero-Knowledge Proofs

Interactive proofs can be useful not only if executing the protocol is faster than verifying the statement. We can also add additional properties that ensure the prover can take advantage of some secret information they use to verify that $x \in L$ without revealing such information. This way the prover can prove knowledge of some secret, which can be used to authenticate a user asking them to prove knowledge of the secret key related to a public key. More complex protocols can benefit from this tool too, as the prover can convince the verifier that they have not misbehave regarding the fact that some encrypted or committed value satisfies the requested properties without revealing it.

We can formalize these ideas using the complexity class of NP problems, decisional problems solvable in polynomial time with a non-deterministic Turing Machine. These problems can also be characterized from the existence of certificates, that is, for every $x \in L$ there exists a polynomial certificate y (another string whose length is bounded by a polynomial applied to the length of x) so that a specific polynomial algorithm that only depends on L (usually characterized as a Turing Machine) that takes as input x and y outputs accept, while no such y exists if $x \notin L$. This class of problems encompass all decisional problems that ask whether there exists some mathematical object satisfying some properties, no matter how difficult or easy is to find that object, if once we have found it verifying that it satisfies the property can be done in polynomial time.

Equivalently we can define $\mathfrak{R}$ the binary relation where a *statement* x is related to a *witness* w (another way of calling the certificate), with size polynomially bounded by the size of x, if the previously defined polynomial algorithm outputs accept. Then the NP language L can be described as the set of x for which there exists some polynomial w such that $(x, w) \in \mathfrak{R}$. We sometimes abuse notation identifying L with $\mathfrak{R}$ and write $x \in \mathfrak{R}$ to say that there exists a witness w such that $(x, w) \in \mathfrak{R}$ (and say x is *true*), and write $x \notin \mathfrak{R}$ (and say x is *false*) otherwise. $\mathfrak{R}$ is then a poly-time verifiable polynomial binary relation.

Our goal is to prove the truthiness of some statement, the fact that $x \in \mathfrak{R}$, without revealing anything else, particularly any information about the w that certifies $(x, w) \in \mathfrak{R}$, besides what can be efficiently deduced from the fact that the statement is indeed true. To do so we introduce the notion of Zero-Knowledge Proofs (of Knowledge).

Definition 1.8 (Zero-Knowledge Proof)**.** A $(2\mu+1)$-*move Honest-Verifier Zero-Knowledge Proof* is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which, given an x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that there exists a witness w such that $(x, w) \in \mathfrak{R}$. We use the notation $\mathrm{ZKP}\left[\, w \;\middle|\; (x, w) \in \mathfrak{R} \,\right]$ when we refer to a proof emphasizing the relation $\mathfrak{R}$ and $\langle \mathcal{P}, \mathcal{V} \rangle$ when we have already defined the role of the prover and the verifier. We might also mention the information each of the participants know writing $\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$.

$\mathcal{P}$ and $\mathcal{V}$ engage in an interaction where $\mathcal{P}$ starts sending an initial message a consecutively answered by $\mathcal{V}$ with a random challenge c_i followed with another response b_i by $\mathcal{P}$ for i from 1 to μ. Finally, $\mathcal{V}$ accepts or rejects the proof checking the conversation $\left(x, a, \{c_i\}_{i=1}^{\mu}, \{b_i\}_{i=1}^{\mu}\right)$.

Analogously as before we denote the output of the interaction for a given statement x with $\langle \mathcal{P}, \mathcal{V} \rangle(x)$. We ask that the protocol has the following three properties:

- **Completeness**: if an honest prover $\mathcal{P}$ knows a valid witness w such that $(x, w) \in \mathfrak{R}$ and follows the protocol, then an honest verifier $\mathcal{V}$ always accepts the conversation.

$$(x, w) \in \mathfrak{R} \implies \langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle(x) = \text{accept}.$$

- **Soundness**: an interactive proof $\langle \mathcal{P}, \mathcal{V} \rangle$ for relation $\mathfrak{R}$ is *sound* with *soundness error* $\epsilon(x)$ if for every malicious prover $\mathcal{P}^*$ and every statement $x \notin \mathfrak{R}$ we have

$$\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = \text{accept}\right] \leq \epsilon(x).$$

- **Honest-Verifier Zero-Knowledge**: there exists a polynomial-time simulator $\mathcal{S}$ that takes as input a statement x, samples μ challenges $\{c_i\}_i$ and outputs an accepted conversation $(x, a, \{c_i\}_i, \{b_i\}_i)$ with the same probability distribution as conversations between honest $\mathcal{P}$ and honest $\mathcal{V}$.

$$\mathcal{S}(x, \{c_i\}_i) \sim \langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle.$$

We might also accept the fact that these two distributions are computationally indistinguishable.

Observe that only the fact that the protocol is executed in order prevents the prover from cheating. If the challenges were known in advance computing a valid conversation might be straightforward, provided we know there exists a simulator doing precisely that. Moreover, the fact that simulated conversations follow the same distribution as real conversations between an honest prover and an honest verifier means that the conversations themselves do not reveal any information on its own, as the simulator is able to compute them without knowing the secret witness. Several transformations exist to ensure that the zero-knowledge property still holds against a malicious verifier that deviates from the protocol. We just focus on Honest-Verifier proofs because that is sufficient to latter obtain secure non-interactive proofs.

Some alternative definitions additionally require the protocol to be public-coin. We define it separately but anyway consider only public-coin protocols.

Definition 1.9 (Public-Coin)**.** An interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ is *public-coin* if all of $\mathcal{V}$'s random choices are made public.

In this case we can assume the challenges from $\mathcal{V}$ are directly uniform samples from some challenge spaces $c_i \leftarrow_{\text{R}} C_i$. The structure of a public-coin *Zero-Knowledge Proof of Knowledge* (ZKPoK) can be seen in the example Protocol 1.1.

With this structure the honest-verifier zero-knowledge property can be proven showing how to simulate a conversation for each possible set of challenges $\{c_i\}_i$.

There are scenarios where the soundness definition from Definition 1.8 is not enough, and we need a stronger one, ensuring that the prover not only can guarantee that x is valid but also knows a witness w of $(x, w) \in \mathfrak{R}$. More formally we say that there is an extractor that, given oracle access to a successful enough prover, can be used to efficiently compute a witness with sufficient probability.

Definition 1.10 (Knowledge-Soundness as in [11])**.** An interactive proof $\langle \mathcal{P}, \mathcal{V} \rangle$ for relation $\mathfrak{R}$ is *knowledge sound* with *knowledge error* $\kappa(x)$ if there exists a positive polynomial p and an algorithm $\mathcal{E}$ with the following properties. The extractor $\mathcal{E}$, given input x and rewindable oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, is expected to run in polynomial time in $|x|$ and outputs a witness w such that $(x, w) \in \mathfrak{R}$ with probability

$$\Pr\left[(x, w) \in \mathfrak{R} \,\middle|\, w \leftarrow_{\text{R}} \mathcal{E}^{\mathcal{P}^*}(x)\right] \geq \frac{\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = \text{accept}\right] - \kappa(x)}{p(|x|)}.$$

A zero-knowledge proof with knowledge-soundness is called a ZKPoK.

That is the meaningful definition in scenarios like proving knowledge of the DLog, where the relation is always satisfied and the interesting fact to prove is the

Protocol 1.1 PUBLIC-COIN ZKPOK EXAMPLE

	$\mathcal{P}\,(x;w)$		$\mathcal{V}\,(x)$
	$a \leftarrow_{\text{R}} \mathcal{P}(x,w)$		
1:		$\xrightarrow{a}$	
			$c_1 \leftarrow_{\text{R}} C_1$
2:		$\xleftarrow{c_1}$	
	$b_1 \leftarrow_{\text{R}} \mathcal{P}(x,w,a,c_1)$		
3:		$\xrightarrow{b_1}$	
$\vdots$		$\vdots$	
			$c_i \leftarrow_{\text{R}} C_i$
$2i$:		$\xleftarrow{c_i}$	
	$b_i \leftarrow_{\text{R}} \mathcal{P}\left(x,w,a,\{c_j\}_{j=1}^{i},\{b_j\}_{j=1}^{i-1}\right)$		
$2i+1$:		$\xrightarrow{b_i}$	
$\vdots$		$\vdots$	
			$c_\mu \leftarrow_{\text{R}} C_\mu$
2μ:		$\xleftarrow{c_\mu}$	
	$b_\mu \leftarrow_{\text{R}} \mathcal{P}\left(x,w,a,\{c_j\}_{j=1}^{\mu},\{b_j\}_{j=1}^{\mu-1}\right)$		
$2\mu+1$:		$\xrightarrow{b_\mu}$	
			$\mathcal{V}\left(x,a,\{c_j\}_{j=1}^{\mu},\{b_j\}_{j=1}^{\mu}\right)$ decides to accept or reject

knowledge of the witness. We however are interested in adapting the interactive proof to a non-interactive scenario where the same proof can be shared by someone different from the one that produced it, even if they do not know the witness, but in any case the definition is relevant to this analysis because it implies the previous soundness definition (if $\kappa < \epsilon$ then a prover with a success probability greater than ϵ can be used by the extractor to compute a witness with a positive probability, and therefore x is valid) and can usually be more directly proved.

More specific soundness properties are going to be latter defined in Section 3.4 when proving soundness of our protocols.

Whenever the soundness property holds only under certain computational assumptions proofs are sometimes called *arguments*. We are going to equally use the term proof.

In order to prove that the soundness and zero-knowledge properties hold one has to be very careful with the notation used, as what we define is the protocol that has to be followed by honest provers and verifiers, but a malicious prover can arbitrary deviate. For that matter when in a protocol an honest prover tries to convince a verifier that they know some element a satisfying some relations we denote by $\widetilde{a}$ the element disclosed by a possibly malicious prover, because $\widetilde{a}$ might not satisfy the same relations the honestly computed a does. We can only use whatever relation is actually verified by the verifier. An honest prover will always use $\widetilde{a} = a$.

Equivalently, to prove that the conversations between a prover and a verifier do not reveal any relevant information we will show the existence of a simulator that can output conversations indistinguishable from the real ones. If an original conversation contains an element a, for the same reasons as before, we will call $\widehat{a}$ to the element alleged to play the same role in the simulated conversation. This $\widehat{a}$ could be computed differently, and we have to check that it follows the same distribution as the original a.

In the particular case with $\mu = 1$, i.e. only 3 moves, we can define *special soundness* as the property meaning that a valid witness can be computed from 2 accepting conversations, which directly implies soundness. Such a protocol with 3 moves and special soundness is called a Σ-protocol. Provided that we have defined a more general interactive ZKPoK we have to work with generalized versions of properties usually defined for Σ-protocols.

The most direct use of an interactive ZKPoK is as authentication mechanism working as an identification scheme. An instance of a difficult problem can be used as a public key and only the one who generated it, knowing the solution, can use it as a secret key and execute the protocol to convince anyone that they are who they claim to be.

The main drawback of this approach for other scenarios is that it requires interaction, because only the fact that the conversation has been produced in the right order with the challenges selected by the verifier convinces the verifier of the truthiness of the statement. Anyone else who does not necessarily trust the verifier would then not have any reason to trust the proof (as it could be simulated).

To solve these issues when we need a universally verifiable proof the notion of non-interactive proofs is then introduced. There are different approaches to obtain this goal, but we focus on proofs that come from an interactive version where the role of the verifier has been substituted by a random oracle $\mathcal{O}$ (formally defined in the following subsection), i.e. with the Fiat-Shamir transform [52].

Definition 1.11 (Non-Interactive Zero-Knowledge Proof of Knowledge)**.** A *Non-Interactive Zero-Knowledge Proof of Knowledge* (NIZKPoK), denoted as $(\mathcal{P}, \mathcal{V})$, allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a certain statement x is true because they know some secret information, a witness w, that satisfies a given relation $(x, w) \in \mathfrak{R}$, while preserving confidentiality of such secret. Unlike the interactive variant each participant executes their part on their own, $\mathcal{P}$ takes x and w as input and produces a proof $\Pi \leftarrow_{\mathrm{R}} \mathcal{P}(x, w)$, while $\mathcal{V}$ accepts or rejects the proof by looking at x and Π. We denote the response given by the verifier as $\mathcal{V}(x, \Pi)$. Every NIZKPoK system must satisfy the following properties:

- **Completeness**: if an honest prover knows the witness, then an honest verifier will always accept the proof provided that the protocol was followed.

- **(Knowledge) Soundness**: a protocol is said to be *sound* if no computationally bounded (for example with a limited number of oracle queries) dishonest adversary can produce a valid proof for a false statement ($x \notin \mathfrak{R}$), except with negligible probability.

  A protocol is said to have the stronger property of *Knowledge-Soundness* if there is an extractor such that provided $\mathcal{P}^*$ is an adversary that produces valid proofs for a statement x then the extractor is able to use it to obtain a valid witness such that $(x, w) \in \mathfrak{R}$ with a similar success probability.

- **Zero-Knowledge**: the proof itself should not reveal any additional information about the witness besides the fact that it exists.

In this case we are going to extensibly discuss formal ways to achieve these properties in Chapter 4. Particularly Definitions 4.4 and 4.5 formalize the concepts of soundness and knowledge-soundness for NIZKPoKs that come from interactive proofs, where we use the number of oracle calls as a measure of the computational power of the adversary. As the transform and the particular details are going to be introduced in this chapter we leave there these definitions.

1.2.5 Pseudorandomness

Pseudorandom generators, deterministic algorithms that expand short seeds into apparently random longer bit sequences, are fundamental for cryptography. We ask these pseudorandom sequences to be computationally indistinguishable from truly random sequences by efficient algorithms, and then call them *Cryptographically Secure Pseudorandom Number Generator*s (CSPRNGs).

Definition 1.12 (Hash Function)**.** A *hash function* is a function that maps a message of arbitrary length to a fixed length *hash* or *digest*.

$$H : \{0,1\}^* \mapsto \{0,1\}^s$$

We usually want hash functions to satisfy additional properties, as collision resistance, preimage resistance or second preimage resistance (it should be computationally infeasible to find two messages with the same image, compute a preimage or compute an additional preimage of a known message hash).

In our implementation in Chapter 4, we use the SHA-3 family of hash functions.

Definition 1.13 (eXtendable Output Function)**.** An *eXtendable Output Function* (XOF) is a generalization of a hash function. While a hash function produces an output of fixed length, an XOF can produce a digest of arbitrary length. Moreover, the extension of the length of the output (requesting more digest bits) only costs the generation of these extra bits. Abusing notation we can write the following.

$$\mathsf{XOF} : (\{0,1\}^*, n) \mapsto \{0,1\}^n$$

In Chapter 4 we use an XOF (SHAKE-128) this time to obtain an appropriate length buffer to store the cryptographically secure pseudorandom bytes used to sample an integer uniformly at random from a given interval.

Sometimes we need a perfect source of pseudorandomness to be able to formalize the security proofs. This is modeled as a random oracle and the approach is called the *Random Oracle Model* (ROM).

Definition 1.14 (Random Oracle)**.** A *random oracle* is an ideal oracle $\mathcal{O}$ that can be queried with arbitrary messages and answers a uniformly random element from the output domain if the message has not been queried before or returns the previous answer otherwise.

A *Quantum Random Oracle* follows the same idea regarding quantum computation, and for that matter queries are allowed to be a quantum superposition of different messages, answering too with a quantum superposition of the classical answers.

We can use the ROM to define a simple commitment scheme, that we then use as auxiliary commitment scheme to design ZKPoKs of more versatile commitments. Let aGen, aCom and aVer be the following algorithms:

- aGen: on input a security parameter 1^λ output as a public key a random oracle $\mathcal{O}$ with $\{0,1\}^\lambda$ as output domain, and the security parameter λ as public parameter.
$$(pk := \mathcal{O}; pp := \lambda) \leftarrow_{\mathrm{R}} \mathsf{aGen}(1^\lambda)$$
- aCom: given a message $m \in \{0,1\}^*$ sample a random bit string $r \leftarrow_{\mathrm{R}} \{0,1\}^\lambda$ and query the random oracle with the concatenation of the message and the

randomness $m\|r$. The answer $c \leftarrow \mathcal{O}(m\|r)$ would work as commitment and the randomness as opening.

$$(c := \mathcal{O}(m\|r), o := r) \leftarrow_{\mathrm{R}} \mathsf{aCom}(m; \mathcal{O}, \lambda)$$

- aVer: given a commitment c, a message m and an opening $o := r$, together with the public key and the public parameters, the verifier checks if $r \overset{?}{\in} \{0,1\}^{\lambda}$ and if c is the answer from the random oracle to the concatenation of the message and the opening.

$$\mathsf{aVer}(c, m, r; \mathcal{O}, \lambda) \text{ outputs } \begin{cases} \text{accept} & \text{if } \begin{cases} r \overset{?}{\in} \{0,1\}^{\lambda} \\ c \overset{?}{\leftarrow} \mathcal{O}(m\|r) \end{cases} \\ \text{reject} & \text{otherwise} \end{cases}$$

This scheme is a well known construction, we can informally check that it satisfies the properties needed to be a secure commitment scheme. Correctness property holds by construction, because the random oracle always returns the same answer when queried with the same message. It is computationally hiding because the oracle answer of $m\|r$ is independent of $m\|r'$ if $r \neq r'$, and the randomness comes from a sufficiently large space. It is also computationally binding, because the fact that the answers from the oracle are uniformly distributed in an exponentially large space makes it infeasible to find a collision.

1.3 Lattices

Lattice-based cryptography is currently a prolific research topic as it allows building many cryptographic primitives from post-quantum assumptions, as lattice problems are widely believed to be computationally hard even for quantum computers. Besides that, an important property of the lattice-problems used for cryptography is that they enjoy worst-case to average-case reductions. That means we know how to solve the most difficult instance of a lattice problem just being able to break with sufficient probability a random instance of our cryptographic construction. This is important because it implies that the hardness assumption works on average, and not only in some special cases, which is a desirable but uncommon property.

Another reason explaining why the cryptographic community has great confidence on lattice assumptions is the fact that lattices have long been used for cryptographic purposes, being used as a tool for cryptanalysis against RSA schemes before lattice-based cryptography itself was first proposed [43, 27].

Lattices have also become a celebrated alternative because they permit the realization of *Fully Homomorphic Encryption* (FHE), allowing anyone to perform computations on encrypted data. However, sometimes we do not need such a strong feature, and we are going to see how to prove in zero-knowledge that some element is the result of applying an arithmetic circuit to some secret inputs, which might still be useful for many applications.

In any case we describe in this section the basic notions we need to define the lattice problems of interest and start discussing the main challenges we face when trying to build ZKPoKs for such problems.

1.3.1 Notation

We usually follow the standard notation in which column vectors are denoted by lower-case bold-faced roman letters, $\boldsymbol{a}$ or $\boldsymbol{b}$, and row vectors are denoted as transposed column vectors $\boldsymbol{a}^\top$. Unless it is specifically mentioned otherwise upper-case bold-faced roman letters, $\boldsymbol{M}$ or $\boldsymbol{A}$, are going to represent matrices. We denote the binary logarithm simply as log, and use ln for the natural logarithm. We denote by $\lfloor x \rfloor$ the largest integer not greater than x and by $\lfloor x \rceil$ the closest integer to x, with ties broken upwards.

Regarding lattices, we mostly follow the notation from [96]. Throughout the rest of the dissertation we will mainly work over the quotient ring of polynomials $R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$, where n is a power of two and q is an odd prime. For the sake of being able to define norms of polynomials and vectors of polynomials, we choose $\{-\lfloor q/2 \rfloor, \dots, \lfloor q/2 \rfloor\}$ as representatives for $\mathbb{Z}_q$. We denote the representative of $a \in \mathbb{Z}_q$ as $a \ \text{ rem } q \in \mathbb{Z}$.

It is important to remark the need for unique representatives. Intermediate results of operations in $\mathbb{Z}_q$ can be computed using any equivalent integer from $\mathbb{Z}$, as long as we always use the same representatives for the final outcome. That is important because even if we have a theoretical proof ensuring that some element follows a uniform distribution in $\mathbb{Z}_q$, independent of any secret, a bad implementation could be leaking relevant information if the representative does depend on the secret.

This way given a vector $\boldsymbol{v} \in \mathbb{Z}_q^n$ we define the infinity norm as $\|\boldsymbol{v}\|_\infty := \max_{1 \le i \le n} |v_i|$, where v_i are the coordinates of vector $\boldsymbol{v}$ taking the mentioned elements as representatives. We identify a polynomial $p \in R_q$ with the vector $\boldsymbol{p} \in \mathbb{Z}_q^n$ that has its coefficients as elements, and define the norm of a polynomial as the norm of the related vector $\|p\|_\infty := \|\boldsymbol{p}\|_\infty$. The infinity norm of a vector of polynomials is analogously defined. We denote the Hamming weight of a vector $\boldsymbol{v}$, the number of non-zero coordinates, as $\|\boldsymbol{v}\|_{\mathsf{H}}$.

Lattice-based cryptography deals with noisy equations, where the noise terms come from an error distribution χ. The tightest security guarantees are obtained choosing these errors following Gaussian distributions.

The *discrete Gaussian distribution* of parameter σ over the integers, denoted as D_σ, assigns to $x \in \mathbb{Z}$ a probability which is proportional to

$$\rho_\sigma(x) = \exp\left(-\frac{1}{2}x^2/\sigma^2\right).$$

Sometimes we need to ensure that such errors are small enough, and for that reason we truncate the distribution disregarding the tails so that the errors are indeed bounded. We denote by $D_{\sigma,B}$ the *truncated discrete Gaussian distribution* over $\mathbb{Z}$ obtained by sampling an element from a discrete Gaussian D_σ conditioned to be in the interval $[-B, B)$, where the bound B is usually going to be another power of two. We define the interval like this for convenience, as it characterizes errors that can be described with a fixed number of bits. We sometimes abuse notation writing $\|p\|_\infty \leq B$ while what we are going to check is whether all the coefficients from p belong to the interval $[-B, B)$.

1.3.2 Basic Definitions

Definition 1.15 (Lattice)**.** A *lattice* $\mathcal{L}$ is a set of points in an n-dimensional space, usually $\mathbb{R}^n$, with a periodic structure. That is, the following two conditions hold:

- It is an additive subgroup: $0 \in \mathcal{L}$ and $\forall x, y \in \mathcal{L}$ we have $-x, x + y \in \mathcal{L}$.
- It is discrete: $\forall x \in \mathcal{L}$ there exists a neighborhood of x in $\mathbb{R}^n$ such that x is the only point of the lattice.

Usually a lattice is defined by a basis of vectors. An example of a 2-dimensional lattice is presented in Figure 1.1.

Definition 1.16 (Generated lattice)**.** Given k linearly independent vectors $\boldsymbol{b}_1, \ldots,$ $\boldsymbol{b}_k \in \mathbb{R}^n$, the *lattice generated* by them is the set defined as follows.

$$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k) := \left\{ \sum_{i=1}^{k} z_i \boldsymbol{b}_i \;\middle|\; z_i \in \mathbb{Z} \right\} = \left\{ \boldsymbol{B}\boldsymbol{z} \;\middle|\; \boldsymbol{z} \in \mathbb{Z}^k \right\} = \mathcal{L}(\boldsymbol{B}).$$

We have called $\boldsymbol{B}$ to the matrix whose columns are vectors $\boldsymbol{b}_i$. We say $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ form a *basis* of the lattice $\mathcal{L}(\boldsymbol{B})$. We additionally say the lattice is *full-rank* if $k = n$.

Definition 1.17 (Minimum λ_i)**.** We define the *minimum* $\lambda_i(\mathcal{L})$ as the radius of the smallest hypersphere centered in the origin that contains at least i linearly independent points of the lattice $\mathcal{L}$.

As an example we have drawn λ_2 for the previously presented lattice in Figure 1.2.

1.3.3 Fundamental Problems

Now we can define some of the problems whose hardness will be used as hypothesis for our cryptographic constructions.

Definition 1.18 (Approximate Shortest Vector Problem – γ-SVP)**.** Given a basis $\boldsymbol{B}$ of a lattice $\mathcal{L}(\boldsymbol{B})$ the *Approximate Shortest Vector Problem* (γ-SVP) consists on finding a non-zero vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\boldsymbol{B}))$.

If $\gamma = 1$ it is called the *Shortest Vector Problem* (SVP). We represent the solution of the SVP for the lattice generated by $\boldsymbol{B}$ in Figure 1.3. The difficulty of this problem depends on the approximating factor γ. Our problems involve a $\gamma(n)$ that is a polynomial of the dimension of the lattice. It has been proven to be an NP-hard problem in its exact version and also for some subpolynomial approximations [5]. The best algorithms for the approximated versions have exponential cost [57], and it is believed that no successful PPT algorithm exists.

Definition 1.19 (Approximate Closest Vector Problem – γ-CVP)**.** Given a basis $\boldsymbol{B}$ of a lattice $\mathcal{L}(\boldsymbol{B})$ and a target vector $\boldsymbol{t} \in \mathbb{R}^n$, the *Approximate Closest Vector Problem* (γ-CVP) consists on finding a vector $\boldsymbol{u} \in \mathcal{L}(\boldsymbol{B})$ such that $\|\boldsymbol{u} - \boldsymbol{t}\| \leq \gamma \min_{\boldsymbol{w} \in \mathcal{L}(\boldsymbol{B})} \|\boldsymbol{w} - \boldsymbol{t}\|$.

Analogously as before it is called *Closest Vector Problem* (CVP) if the approximation factor is 1. An example of a solution for a CVP can be found in Figure 1.4.

1.3.4 Lattice Basis Reduction

A lattice can be defined by different basis. Multiplying $\boldsymbol{B}$ by a unimodular matrix $\boldsymbol{U}$ gives us another basis for the same lattice. In fact, all basis defining the same lattice can be obtained this way.

Theorem 1.20.
$\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B}')$ if and only if there exist a unimodular matrix $\boldsymbol{U}$ such that $\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{U}$.

Proof. Assume $\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{U}$. As $\boldsymbol{U}$ is unimodular its inverse $\boldsymbol{U}^{-1}$ is unimodular too. In particular, they are both integer matrices, and we also have $\boldsymbol{B} = \boldsymbol{B}'\boldsymbol{U}^{-1}$. Columns of $\boldsymbol{B}'$ are integer combinations of columns of $\boldsymbol{B}$ (therefore $\mathcal{L}(\boldsymbol{B}') \subset \mathcal{L}(\boldsymbol{B})$) and columns of $\boldsymbol{B}$ are integer combinations of columns of $\boldsymbol{B}'$ (therefore $\mathcal{L}(\boldsymbol{B}) \subset \mathcal{L}(\boldsymbol{B}')$). Consequently, $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B}')$.

Assume $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{B}')$. Then each column $\boldsymbol{b}'_i$ of $\boldsymbol{B}'$ is a point of the lattice $\mathcal{L}(\boldsymbol{B})$, generated by the columns of $\boldsymbol{B}$. Then $\boldsymbol{b}'_i = \boldsymbol{B}\boldsymbol{u}_i$, where $\boldsymbol{u}_i \in \mathbb{Z}^k$ are the corresponding

Figure 1.1 Lattice generated by $\boldsymbol{B}$

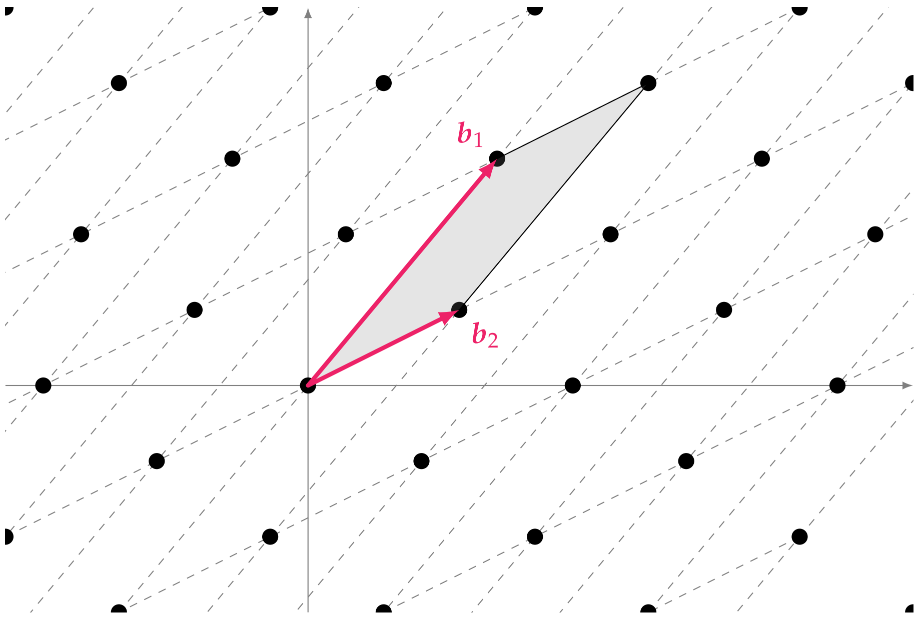


Figure 1.2 Minimum λ_2 for the lattice generated by $\boldsymbol{B}$

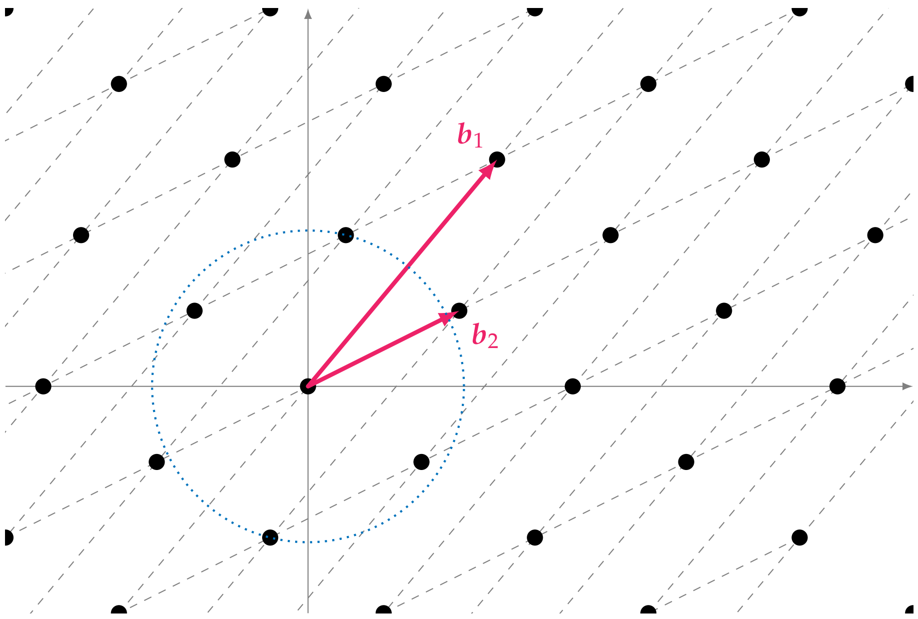


Figure 1.3 Solution of the SVP for the lattice generated by $\boldsymbol{B}$

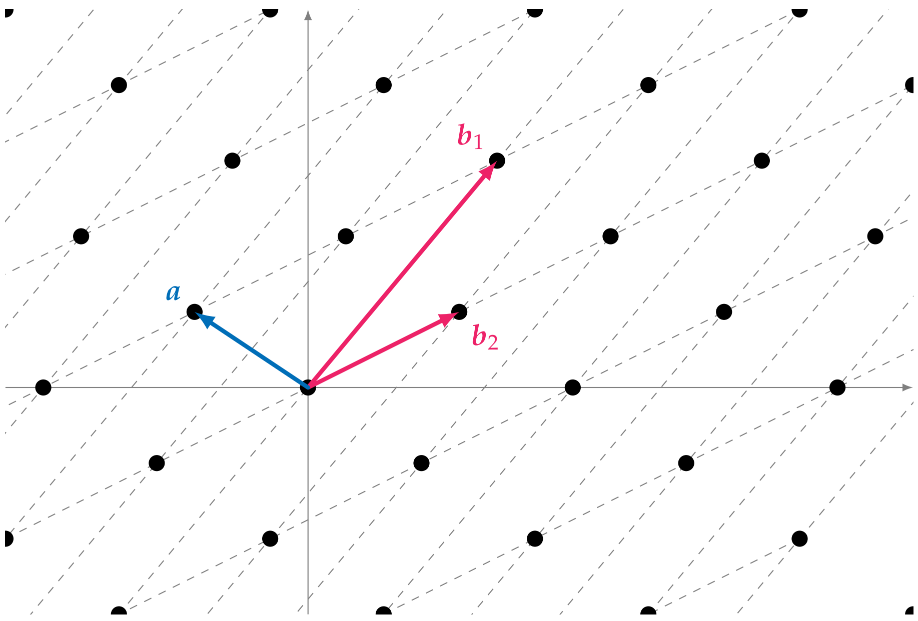
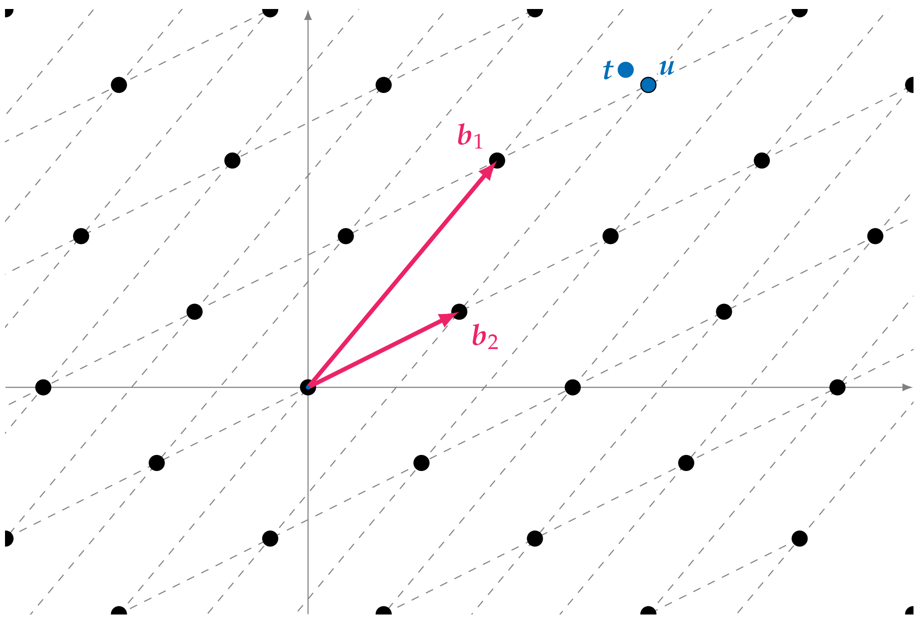


Figure 1.4 Solution of the CVP for a lattice generated by $\boldsymbol{B}$



coefficients of the columns of $\boldsymbol{B}$. We have $\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{U}$ where $\boldsymbol{U}$ is an integer matrix. And by the same argument $\boldsymbol{B} = \boldsymbol{B}'\boldsymbol{V}$, where $\boldsymbol{V}$ is another integer matrix. Combining the two expressions we get $\boldsymbol{B}' = \boldsymbol{B}'\boldsymbol{V}\boldsymbol{U}$. Then $\boldsymbol{B}'(\boldsymbol{V}\boldsymbol{U} - \boldsymbol{Id}) = \boldsymbol{0}$. Since $\boldsymbol{B}'$ is non-singular $\boldsymbol{V}\boldsymbol{U} = \boldsymbol{Id}$ and $\boldsymbol{U}$ is unimodular. □

Corollary 1.21. *The* lattice determinant *of a full-rank lattice, defined as the absolute value of the determinant of a basis,* $\det(\mathcal{L}(\boldsymbol{B})) := |\det(\boldsymbol{B})|$, *is well-defined and a lattice invariant.*

We can use this invariant to find bounds for λ_1. It is important to find invariants as some problems over lattices are hard or easy depending on some properties of the basis defining the lattice that is given as input. Informally, a basis is said to be *good* if it has short highly orthogonal vectors, and *bad* if it has low orthogonality.

We can attempt to solve the CVP by just rounding the coordinates of a point $\boldsymbol{t}$, given in basis $\boldsymbol{B}$, to get a lattice point $\boldsymbol{u}$. As we can see in Figures 1.5 and 1.6 the space is then divided in regions that would go to the same lattice point, and the fact that we are using the vectors from $\boldsymbol{B}$ to define the coordinates means that these regions are shaped into n-dimensional parallelepipeds defined by $\boldsymbol{B}$. Given a target $\boldsymbol{t}$ the point $\boldsymbol{u}$ that we obtain would be the lattice point in the center of the parallelepiped containing $\boldsymbol{t}$, gray colored in the figures. This parallelepiped is called the *fundamental parallelepiped*, and has volume $\det \mathcal{L}(\boldsymbol{B})$, independent of the basis. Nevertheless, following this approach with a *bad* basis (as in Figure 1.5) yields worse results than with a *good* basis (as in Figure 1.6). That is because all points in the gray area would go to the same $\boldsymbol{u}$ at its center when rounding their coordinates, but a more skewed parallelepiped means that we are rounding to $\boldsymbol{u}$ points that are much further apart.

It is known that better results are obtained if before starting this procedure the basis is orthogonalized and the space is divided into boxes. This approach is followed by Babai's Nearest Plane algorithm for solving the approximated CVP, but we see in Figures 1.7 and 1.8 that the quality of the outcome heavily depends too on the *quality* of the basis.

We can formalize this notion of *good* and *bad* basis. Some algorithms work well with highly orthogonal basis, but have a very low probability of success (or can only guarantee a worse approximation factor) if the basis has low orthogonality. We also say that a highly orthogonal basis has a low orthogonality defect and vice versa.

Definition 1.22 (Orthogonality defect)**.**

The *orthogonality defect* of a full-rank lattice basis $\boldsymbol{B}$ is given by

$$\delta(\boldsymbol{B}) := \frac{\prod_{i=1}^{n} \|\boldsymbol{b}_i\|}{\det(\boldsymbol{B})}.$$

Figure 1.5 Rounding for the CVP (*bad* basis)

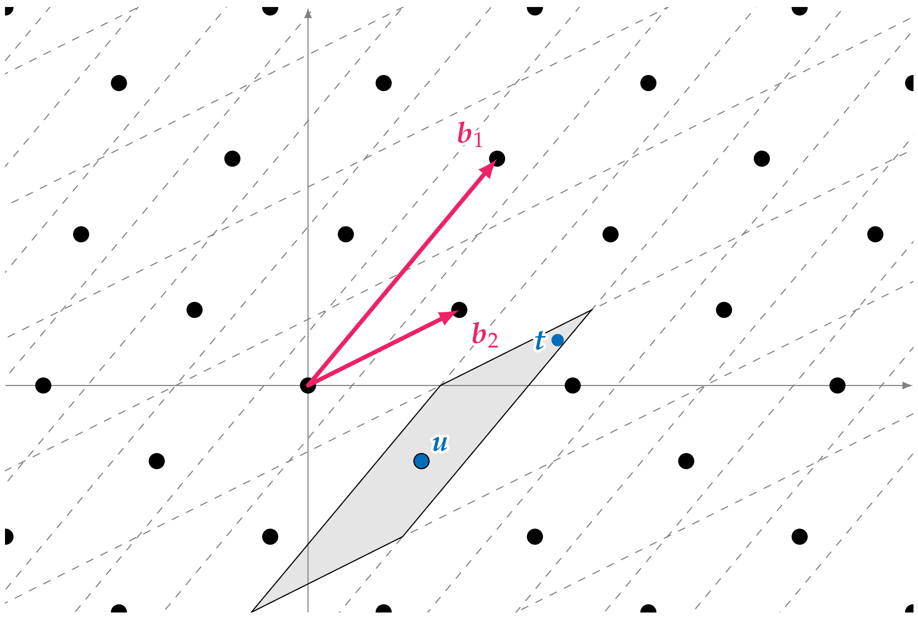


Figure 1.6 Rounding for the CVP (*good* basis)

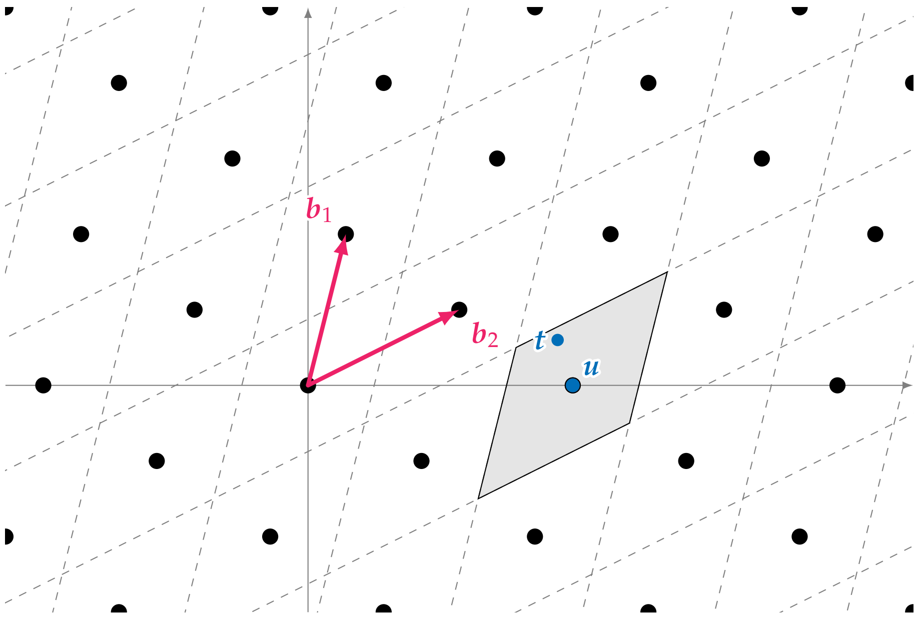


Figure 1.7 Gram-Smith Orthogonalization for the CVP (*bad* basis)

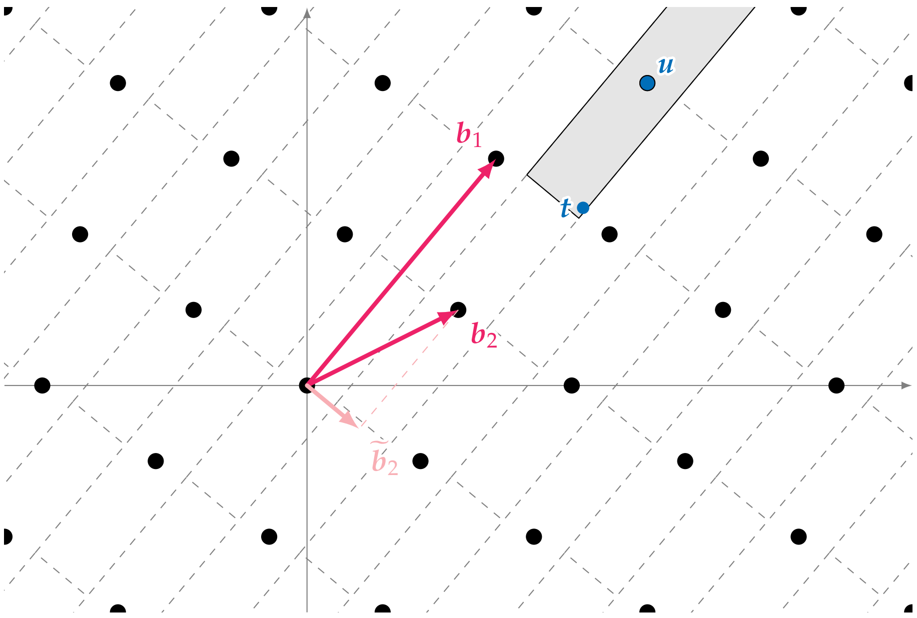
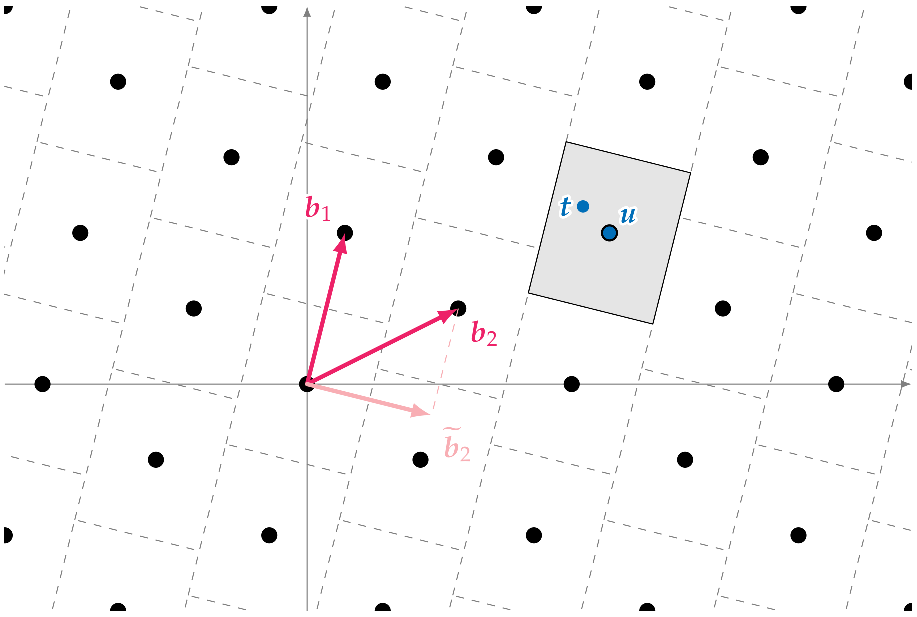


Figure 1.8 Gram-Smith Orthogonalization for the CVP (*good* basis)



It can also be normalized taking the n-root, giving $\sqrt[n]{\delta(\boldsymbol{B})}$.

Nevertheless, transforming a *bad* basis into a *good* basis, with short orthogonal vectors, is a difficult problem. In a two-dimensional lattice it can be done performing Gaussian elimination. This can be generalized to an n-dimensional lattice as it is done in the *Lenstra-Lenstra-Lovasz* (LLL) algorithm [73], performing Gaussian elimination on the elements of the basis two by two. However, the obtained basis has vectors whose length is still exponentially far from optimal.

This can be improved working with blocks of k vectors, instead of pairs of vectors, as it is done in the *Blockwise Korkine-Zolotarev* (BKZ) reduction [106]. However, this algorithm has to find the shortest vector of a k-dimensional lattice as a subroutine, which is again a hard problem even approximately (it is precisely the SVP on a slightly smaller dimension, and the recursion saves some computations but is still far from efficient).

The last improvements on finding algorithms able to obtain better basis focus on this k-dimensional SVP, following two different approaches. Enumeration techniques [54, 66, 58], as depicted in Figure 1.9, try to do an exhaustive search in a region of space. We can choose this region so that it surely contains the shortest vector, as it is done in Figure 1.9 where the whole hypersphere with radius the length of the shortest vector of the basis is used. More efficient algorithms can be obtained cleverly reducing its search space, leveraging this trade-off to achieve better efficiency without decreasing so much its success probability. Sieving techniques [92, 122], as depicted in Figure 1.10, sample random points of the lattice (colored blue in Figure 1.10) using small random integer linear combinations of the vectors of the basis and try to find pairs of them that are close (if two vectors are close its difference is a short vector of the lattice). We have drawn in Figure 1.10 the shortest difference among the randomly selected points. With these shorter vectors a new basis is defined, and the procedure is repeated. In order to improve efficiency not all differences are computed, the space is divided into several regions and only sampled lattice points from the same region are compared. These regions can be delimited by hyperplanes, cones centered at the origin, hyperspheres centered at randomly chosen lattice points, etc. Different choices yield up to a great variety of algorithms with different performance and computational cost (regarding time and space).

However, despite all these recent techniques and improvements their computational cost is still exponential in the lattice dimension n, and some of the algorithms that have better asymptotic cost due to a smaller constant in the exponent also have worse behavior for smaller dimensions, and the asymptotic improvement only dominates for dimensions far above what it is used for cryptography.

Figure 1.9 Enumeration

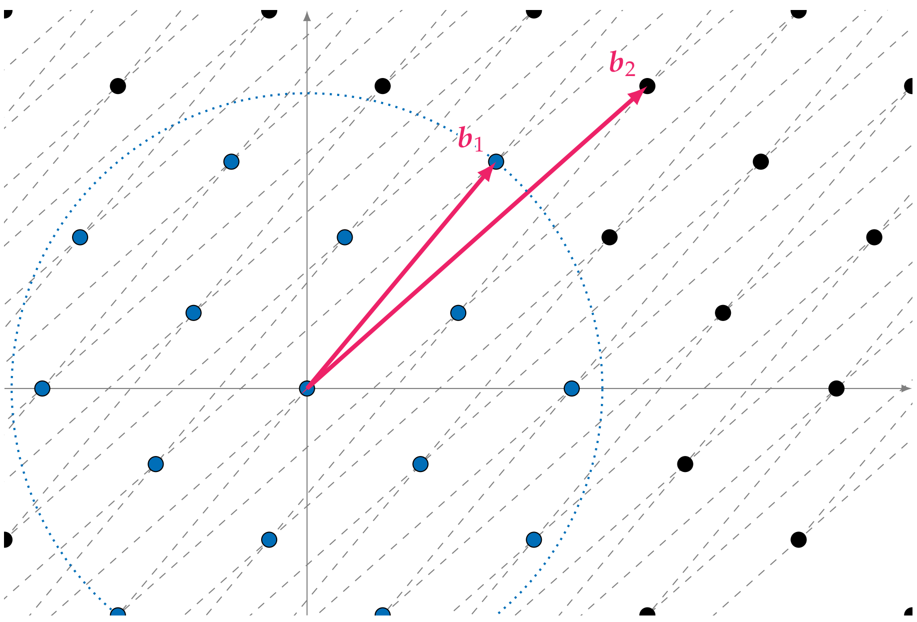
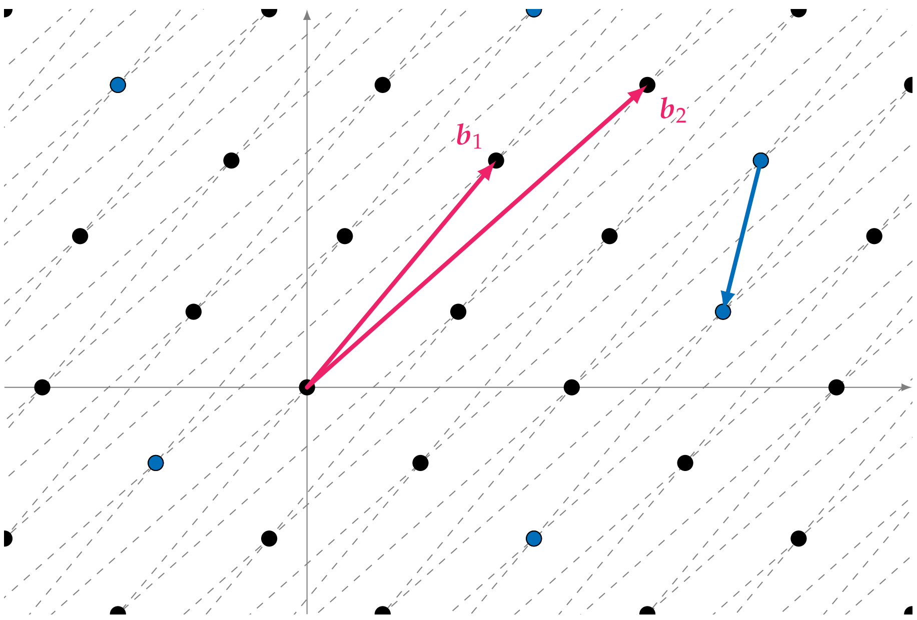


Figure 1.10 Sieving



1.3.5 Additional Definitions

In order to work with lattices in a computer it is preferable to work modulo a prime q. A q-ary lattice is an integer lattice for which whether a point x belongs or not to the lattice is determined by $x \bmod q$.

Definition 1.23 (q-ary lattices)**.**
A lattice $\mathcal{L} \subset \mathbb{R}^n$ is said to be *q-ary* if $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, for an integer q.

Given a lattice we can also define its dual, which is again a lattice.

Definition 1.24 (Dual Lattice)**.**
The *dual lattice* of $\mathcal{L} \subset \mathbb{R}^n$ is a lattice $\mathcal{L}^* := \left\{ \boldsymbol{w} \;\middle|\; \langle \boldsymbol{w}, \mathcal{L} \rangle \subseteq \mathbb{Z} \right\}$.

There are two usual ways of representing a q-ary lattice of dimension m given a full-rank matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$, with $m \geq n$. The first one is called the Λ_q form.

$$\Lambda_q(\boldsymbol{A}) := \left\{ \boldsymbol{y} \in \mathbb{Z}^m \;\middle|\; \boldsymbol{y} = \boldsymbol{A}\boldsymbol{z} \mod q,\ \boldsymbol{z} \in \mathbb{Z}^n \right\} \subset \mathbb{R}^m.$$

And the other is called the orthogonal Λ_q form.

$$\Lambda_q^{\perp}(\boldsymbol{A}) := \left\{ \boldsymbol{y} \in \mathbb{Z}^m \;\middle|\; \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} = \boldsymbol{0} \mod q \right\} \subset \mathbb{R}^m.$$

Proposition 1.25. *Lattices $\Lambda_q(\boldsymbol{A})$ and $\Lambda_q^{\perp}(\boldsymbol{A})$ are dual of each other, up to normalization.*

Proof. We can verify that $\Lambda_q^{\perp}(\boldsymbol{A}) = q\Lambda_q(\boldsymbol{A})^*$ and that $\Lambda_q(\boldsymbol{A}) = q\Lambda_q^{\perp}(\boldsymbol{A})^*$. Let us prove the first equality seeing that a vector $\boldsymbol{y}$ belonging to the first lattice also belongs to the second and vice versa.

$$\begin{aligned}
&\Lambda_q^{\perp}(\boldsymbol{A}) \subseteq q\Lambda_q(\boldsymbol{A})^* : \\
&\quad \boldsymbol{y} \in \Lambda_q^{\perp}(\boldsymbol{A}) \subseteq \mathbb{Z}^m \\
&\quad \boldsymbol{y}^{\mathsf{T}}\boldsymbol{A} = \boldsymbol{0}^{\mathsf{T}} \mod q \\
&\quad \boldsymbol{y}^{\mathsf{T}}\boldsymbol{A} = q\boldsymbol{a}^{\mathsf{T}}, \quad \boldsymbol{a} \in \mathbb{Z}^n \\
&\quad (q^{-1}\boldsymbol{y})^{\mathsf{T}}\boldsymbol{A} = \boldsymbol{a}^{\mathsf{T}} \\
&\quad (q^{-1}\boldsymbol{y})^{\mathsf{T}}(\boldsymbol{A}\boldsymbol{z}) = \boldsymbol{a}^{\mathsf{T}}\boldsymbol{z} \in \mathbb{Z}, \quad \boldsymbol{z} \in \mathbb{Z}^n \\
&\quad (q^{-1}\boldsymbol{y}) \in \Lambda_q(\boldsymbol{A})^* \\
&\quad \boldsymbol{y} \in q\Lambda_q(\boldsymbol{A})^* \\
&\qquad \implies \Lambda_q^{\perp}(\boldsymbol{A}) \subseteq q\Lambda_q(\boldsymbol{A})^*
\end{aligned}$$

$$\begin{aligned}
&\Lambda_q^{\perp}(\boldsymbol{A}) \supseteq q\Lambda_q(\boldsymbol{A})^* : \\
&\quad \boldsymbol{y} \in q\Lambda_q(\boldsymbol{A})^* \\
&\quad \boldsymbol{y} = q\boldsymbol{y}', \quad \boldsymbol{y}' \in \Lambda_q(\boldsymbol{A})^* \\
&\quad \boldsymbol{y}^{\mathsf{T}}\boldsymbol{A} = q\boldsymbol{y}'^{\mathsf{T}}\boldsymbol{A} \\
&\quad \boldsymbol{y}^{\mathsf{T}}\boldsymbol{A} = q\boldsymbol{a}^{\mathsf{T}}, \quad \boldsymbol{a} \in \mathbb{Z}^n \\
&\quad \boldsymbol{y}^{\mathsf{T}}\boldsymbol{A} = \boldsymbol{0}^{\mathsf{T}} \mod q \\
&\quad \boldsymbol{y} \in \Lambda_q^{\perp}(\boldsymbol{A}) \\
&\qquad \implies q\Lambda_q(\boldsymbol{A})^* \subseteq \Lambda_q^{\perp}(\boldsymbol{A})
\end{aligned}$$

From the definition of the dual lattice it is easy to see that the dual of a dual lattice is the original lattice itself $(\mathcal{L}^*)^* = \mathcal{L}$ (this can be proved finding a basis of the dual lattice in terms of a basis of the original lattice) and that $(q\mathcal{L})^* = q^{-1}\mathcal{L}^*$ (direct consequence of the definition).

Applying these two properties to the first equality $\Lambda_q^\perp(\boldsymbol{A}) = q\Lambda_q(\boldsymbol{A})^*$ we can get the second one $\Lambda_q(\boldsymbol{A}) = q\Lambda_q^\perp(\boldsymbol{A})^*$.

$$q\left(\Lambda_q^\perp(\boldsymbol{A})\right)^* = q\left(q\Lambda_q(\boldsymbol{A})^*\right)^* = q \cdot q^{-1}\left(\Lambda_q(\boldsymbol{A})^*\right)^* = \Lambda_q(\boldsymbol{A}). \qquad \square$$

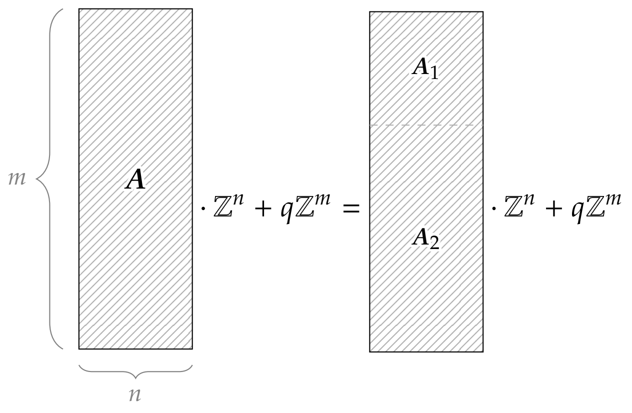
Proposition 1.26. *Let $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ be a full-rank matrix. Then $\Lambda_q(\boldsymbol{A})$ is a full-rank lattice of dimension m and determinant q^{m-n}.*

Proof.

This can be proved quite straightforwardly manipulating the matrix that generates the lattice, but the steps are somehow delicate because we have to choose at each step if we consider the elements in $\mathbb{Z}$ or in $\mathbb{Z}_q$. To help describe the steps of the proof we include visual representations of the matrices involved.

Elements from $\Lambda_q(\boldsymbol{A})$ are equivalent modulo q to an integer linear combination of the columns of $\boldsymbol{A}$. We can describe this as an integer linear combination of the columns of $\boldsymbol{A}$ considered as vectors in $\mathbb{Z}^m$ plus any vector of integers in $\mathbb{Z}^m$ multiple of q. We represent it as a matrix-vector product $\boldsymbol{A} \cdot \mathbb{Z}^n$ plus $q\mathbb{Z}^m$.

Provided that $\boldsymbol{A}$ is a full-rank matrix it has n independent rows. Without loss of generality, reordering the rows of $\boldsymbol{A}$ (which just means reordering the coordinates of the lattice points), we can assume the first n rows of $\boldsymbol{A}$, that we are going to call $\boldsymbol{A}_1$, are already independent as vectors of $\mathbb{Z}_q^n$.

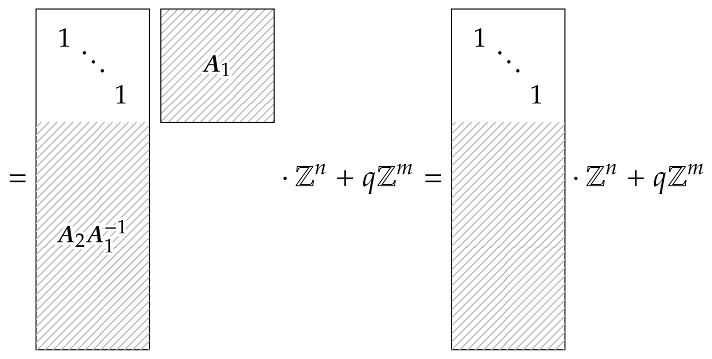


m
$\boldsymbol{A}$
n
$\cdot \mathbb{Z}^n + q\mathbb{Z}^m =$
$\boldsymbol{A}_1$
$\boldsymbol{A}_2$
$\cdot \mathbb{Z}^n + q\mathbb{Z}^m$

We can then factor out the contribution of $\boldsymbol{A}_1$. Let ${\boldsymbol{A}_1}^{-1}$ be a representative of the inverse matrix of $\boldsymbol{A}$ in $\mathbb{Z}_q^{n\times n}$. This means that the product ${\boldsymbol{A}_1}^{-1}\boldsymbol{A}_1$, when considered in $\mathbb{Z}^{n\times n}$ is going to be the identity matrix plus some multiple of q matrix.

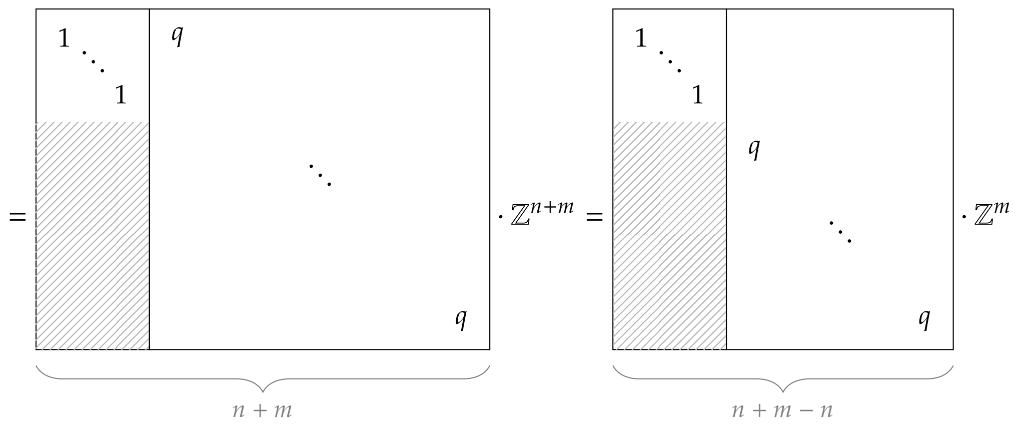
We can safely ignore this second part because its contribution has been already taken into account by the $q\mathbb{Z}^m$ term.

Notice that, since $\boldsymbol{A}_1$ is non-singular, integer combinations of its columns can be any integer vector of $\mathbb{Z}^n$. The final conclusion is that we can assume the first block of the matrix is just an $n \times n$ identity matrix.


$=$
$1 \ddots 1$
$\boldsymbol{A}_2\boldsymbol{A}_1^{-1}$
$\boldsymbol{A}_1$
$\cdot\, \mathbb{Z}^n + q\mathbb{Z}^m =$
$1 \ddots 1$
$\cdot\, \mathbb{Z}^n + q\mathbb{Z}^m$

We can consider that the multiples of q part is a linear integer combination of the columns of a q scalar matrix. Observe the leftmost n columns with a q in the first n rows are integer linear combinations of the other columns, and can be omitted.

That is the case because the q's and the zeroes in the first n elements of any of these columns can be obtained multiplying q by the corresponding column from the first n ones that has a 1 in the same position. The bottom $m - n$ elements obtained multiplying this column by q would be non-zero, but, as all of them would be multiples of q, we can subtract the linear integer combination of the rightmost $m - n$ columns that equals to this $m - n$ vector.



We finally obtain that a q-ary lattice in its Λ form is generated by a full-rank lower triangular matrix of dimension m, and we can directly compute its determinant because it is the product of its diagonal, that is, $\det(\Lambda_q(\boldsymbol{A})) = q^{m-n}$. □

The fact that we have a deep understanding of this particular class of lattices makes them suitable for our purposes. For example, as we have mentioned, there are known results relating bounds for $\lambda_1(\mathcal{L})$ with $\det(\mathcal{L})$, and in this case we have seen that $\det(\Lambda_q(\boldsymbol{A}))$ does only depend on the dimensions of $\boldsymbol{A}$.

1.3.6 Lattice Problems and Cryptography

Short Integer Solution

Definition 1.27 (Short Integer Solution – $\mathrm{SIS}_{n\ q\ m\ \beta}$)**.** Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ be a uniformly random matrix. The *Short Integer Solution* (SIS) problem consists on finding a non-zero vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\|\boldsymbol{x}\| \leq \beta$ and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$.

Observe we can see this as finding short vectors in the lattice $\Lambda_q^{\perp}(\boldsymbol{A}^{\mathsf{T}})$. The problem is usually defined with the Euclidean norm $\|\boldsymbol{x}\|_2$, but can also be defined with the infinity norm $\|\boldsymbol{x}\|_\infty$, possibly adapting the bound, provided these two norms are equivalent and $\|\boldsymbol{x}\|_2 \leq \sqrt{m}\,\|\boldsymbol{x}\|_\infty$.

We can also define an inhomogeneous version of the previous problem.

Definition 1.28 (Inhomogeneous Short Integer Solution – $\mathrm{ISIS}_{n\ q\ m\ \beta}$)**.** Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ be a uniformly random matrix and $\boldsymbol{y} \in \mathbb{Z}_q^n$ a vector. The *Inhomogeneous Short Integer Solution* (ISIS) problem consists on finding a vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\|\boldsymbol{x}\| \leq \beta$ and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$.

Notice that finding two different solutions to the ISIS problem implies finding a solution to the SIS problem, satisfying a slightly larger bound, given by the difference of the solutions.

The main point to remark about this problem is that there exists a reduction from any instance of the approximated SVP on any lattice $\mathcal{L} \subseteq \mathbb{R}^n$ to a random instance of the SIS problem on a $\Lambda_q^{\perp}(\boldsymbol{A})$ lattice, as was proven by Ajtai in [4] for a specific polynomial approximation factor that has been later improved. This worse-case to average case reduction we already mentioned significantly simplifies the task of selecting secure public keys.

Recall we classify an algorithm as efficient when we can bound the number of operations it takes until finishing with a polynomial evaluated at the size of its input. Then, this bound is going to be determined by the running time of the worst-case instance of each input size. The fact that we can not find any polynomial bound only means that some instances are going to be hard, but does not guarantee the hardness of the others, not even on average. Only the existence of a reduction from the worst case of another problem to a random instance of the one we use guarantees a hardness level on average (under the assumption that the original problem was hard on the worst-case, which is a much milder and reasonable assumption).

Learning With Errors

Definition 1.29 (Learning With Errors – $\mathrm{LWE}_{n\,q\,m\,\chi}$)**.** Let n, q be integers (q usually prime), χ a discrete probability distribution in $\mathbb{Z}$ (usually a discrete Gaussian distribution) and $\boldsymbol{s}$ a secret vector from $\mathbb{Z}_q^n$.

We denote $\mathcal{L}_{\boldsymbol{s},\chi}$ the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow_{\mathrm{R}} \chi$ and considering it in $\mathbb{Z}_q$ and finally calculating $(\boldsymbol{a}, b \coloneqq \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. There are two variants of the *Learning With Errors* (LWE) problem.

The *Decisional Learning With Errors* problem, Decisional-$\mathrm{LWE}_{n\,q\,\chi}$, consists on deciding if pairs $(\boldsymbol{a}, b)$ are samples of $\mathcal{L}_{\boldsymbol{s},\chi}$ or come from the uniform distribution of $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The *Search Learning With Errors* problem, Search-$\mathrm{LWE}_{n\,q\,\chi}$, consists on recovering $\boldsymbol{s}$ from samples $(\boldsymbol{a}, b)$ obtained from $\mathcal{L}_{\boldsymbol{s},\chi}$.

We generally consider a polynomial adversary has access to polynomially many samples. If we restrict the problem to a fixed number of samples m we denote it as $\mathrm{LWE}_{n\,q\,m\,\chi}$.

This problem, introduced by Regev in [101], is again as hard as the approximated SVP [84], even if the secret $\boldsymbol{s}$ is chosen component by component from the error distribution χ [103]. Some of these reductions involve continuous error terms over the reals, and not discrete Gaussians. We encourage any reader interested in the non-trivial relations among the LWE variants when defined with continuous, rounded or discrete Gaussians, to read the interesting discussions from [95, 59].

It is clear that we can reduce the decisional version to the search version. It is also possible to reduce the search version to the decisional version [101, 94].

We can observe the direct relation of these problems with lattices. Considering m samples from $\mathcal{L}_{\boldsymbol{s},\chi}$ we can stack vectors $\boldsymbol{a}_i^\top$ as rows in a matrix $\boldsymbol{A}$ and the problem can be defined as solving a linear system of equations with noise $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$. If we see $\boldsymbol{A}$ as a matrix that defines $\Lambda_q(\boldsymbol{A})$ then the search-LWE problem becomes recovering the coordinates of a lattice point in its Λ_q form after adding some error $\boldsymbol{e}$, while the decisional-LWE problem is to distinguish uniformly random points in $\mathbb{Z}^m$

from perturbed lattice points.

Cryptographic Applications

As a first example we can see how a good basis with short vectors allows us to build a trapdoor function. Let $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$ be a random matrix and $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ a full rank matrix such that all columns of $\boldsymbol{T}$ have a small norm and $\boldsymbol{A}^\top \boldsymbol{T} = \boldsymbol{0}$. That is, $\boldsymbol{T}$ is a good basis of $\Lambda^\perp(\boldsymbol{A})$.

Notice that we require $\boldsymbol{T}$ to be full-rank and at the same time we impose that $\boldsymbol{A}$ belongs to the cokernel of $\boldsymbol{T}$, which seems contradictory. However, this second property works modulo q, and the first condition we need is $\boldsymbol{T}$ to be full-rank over the integers, where the lattice really lives.

The matrix $\boldsymbol{A}$ is public and everybody can compute LWE samples $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \bmod q$ with its own secret $\boldsymbol{s}$. The LWE hardness assumption tells us that in general it is difficult to recover $\boldsymbol{s}$. However, using the good basis we could compute $\boldsymbol{T}^\top \boldsymbol{b} = \boldsymbol{T}^\top \boldsymbol{A}\boldsymbol{s} + \boldsymbol{T}^\top \boldsymbol{e} = \boldsymbol{T}^\top \boldsymbol{e} \in \mathbb{Z}_q^m$.

Usually we would be stuck here since we know that $\boldsymbol{T}$ is invertible over the integers but not over $\mathbb{Z}_q$. However, if elements of $\boldsymbol{T}$ and $\boldsymbol{e}$ are small enough then the coefficients of $\boldsymbol{T}^\top \boldsymbol{e} \in \mathbb{Z}^m$ would be much smaller than q, and therefore taking the representatives we obtain the equality $\boldsymbol{T}^\top \boldsymbol{b} \;\operatorname{rem}\; q = \boldsymbol{T}^\top \boldsymbol{e} \in \mathbb{Z}^m$ over the integers. Once we have the equation over the integers we can invert $\boldsymbol{T}^\top$, recover $\boldsymbol{e}$ and finally solve the overdetermined system of equations $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b} - \boldsymbol{e} \mod q$.

It is possible to build more advanced public key cryptographic primitives using a good basis as a secret key, but we can also directly use the secret of the LWE problem as secret key.

Proposition 1.30. *The following folklore encryption scheme is secure under the assumption of the hardness of the decisional-LWE problem.*

- *Gen: the generator algorithm takes as input the security parameter 1^λ and defines the public parameters $pp := (n, q, \chi)$. It then chooses a matrix $\boldsymbol{A} \leftarrow_R \mathbb{Z}_q^{n\times n}$, and small vectors $\boldsymbol{s}, \boldsymbol{e} \leftarrow_R \chi^n$. The public key is $pk := (\boldsymbol{A}, \boldsymbol{y}^\top := \boldsymbol{s}^\top \boldsymbol{A} + \boldsymbol{e}^\top)$ and the secret key is $sk := \boldsymbol{s}$.*

$$(sk := \boldsymbol{s}, pk := (\boldsymbol{A}, \boldsymbol{y}^\top := \boldsymbol{s}^\top \boldsymbol{A} + \boldsymbol{e}^\top); pp := (n, q, \chi)) \leftarrow_R \mathsf{Gen}(1^\lambda)$$

- *Enc: in order to encrypt a single bit $m \in \{0, 1\}$ we sample $\boldsymbol{r}, \boldsymbol{x} \leftarrow_R \chi^n$, $x' \leftarrow_R \chi$ and compute a vector of LWE samples $\boldsymbol{a} := \boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}$ and an integer that hides the secret message bit $a := \boldsymbol{y}^\top \boldsymbol{r} + x' + m \lfloor q/2 \rceil \;\operatorname{rem}\; q$.*

$$c := (\boldsymbol{a} := \boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}, a := \boldsymbol{y}^\top \boldsymbol{r} + x' + m \lfloor q/2 \rceil \quad \operatorname{rem}\; q) \leftarrow_R \mathsf{Enc}(m; (\boldsymbol{A}, \boldsymbol{y}^\top), pp)$$

- *Dec: the decryption method recovers the message computing $a - \boldsymbol{s}^\top \boldsymbol{a} \mod q$, as it should be equal to $m \lfloor q/2 \rceil$ plus some errors that only involve small elements. For that matter it outputs 0 if the result is closer to 0 or 1 if it is closer to $q/2$ (taking into account that we are working in $\mathbb{Z}_q$).*

$$\mathsf{Dec}((\boldsymbol{a}, a), \boldsymbol{s}; pp) = \begin{cases} 0 & \text{if } \left| a - \boldsymbol{s}^\top \boldsymbol{a} \mod q \right| < q/4 \\ 1 & \text{otherwise} \end{cases}$$

Since we are rounding elements from $\mathbb{Z}_q$ to 0 or $q/2$ we denote this operation as $\lfloor \cdot \rceil_{q/2}$ and write

$$\left\lfloor a - \boldsymbol{s}^\top \boldsymbol{a} \right\rceil_{q/2} \leftarrow \mathsf{Dec}((\boldsymbol{a}, a), \boldsymbol{s}; pp).$$

Proof. Correctness follows if χ is bounded by a sufficiently small bound, since $a - \boldsymbol{s}^\top \boldsymbol{a} = (\boldsymbol{s}^\top \boldsymbol{A} + \boldsymbol{e}^\top)\boldsymbol{r} + x' + m \lfloor q/2 \rceil - \boldsymbol{s}^\top(\boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}) = \boldsymbol{e}^\top \boldsymbol{r} + x' - \boldsymbol{s}^\top \boldsymbol{x} + m \lfloor q/2 \rceil \approx m \lfloor q/2 \rceil$. If the bound only holds except with a negligible probability we would then obtain a scheme with correctness except with a negligible probability.

Security as defined in Definition 1.5 follows from the LWE hardness assumption. We can prove it following the sequence of games approach, which we will further formalize in Section 1.4. We consider a sequence of games starting from the IND-CPA challenge and slightly modify it in each step in a way that we can ensure the difference between the success probability of the adversary compared with success probability of a random guess is not significantly different among two consecutive games. We proceed this way until we end up with a game for which we can ensure any adversary has exactly the same success probability than a random guess.

- **Game 0**: the adversary $\mathcal{A}$ plays the IND-CPA game and tries to guess which message has been encrypted. We write every step of the encryption algorithm because we are going to later modify that part of the game.

$$\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (\boldsymbol{s}, (\boldsymbol{A}, \boldsymbol{y}^\top); (n, q, \chi)) \leftarrow_{\text{R}} \mathsf{Gen}(1^\lambda), \\ (m_0, m_1, aux) \leftarrow_{\text{R}} \mathcal{A}_1((\boldsymbol{A}, \boldsymbol{y}^\top), (n, q, \chi), 1^\lambda), \\ b \leftarrow_{\text{R}} \{0,1\}, \\ \boldsymbol{r}, \boldsymbol{x} \leftarrow_{\text{R}} \chi^n, \; x' \leftarrow_{\text{R}} \chi, \\ \boldsymbol{a} := \boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}, \; a := \boldsymbol{y}^\top \boldsymbol{r} + x' + m_b \lfloor q/2 \rceil \mod q, \\ b_{\mathcal{A}} \leftarrow_{\text{R}} \mathcal{A}_2((\boldsymbol{a}, a), (\boldsymbol{A}, \boldsymbol{y}^\top), (n, q, \chi), aux, 1^\lambda) \end{array} \right]$$

- **Game 1**: we substitute the public key by a fake one that is just a uniformly sampled matrix and vector pair $(\boldsymbol{A}, \boldsymbol{y}^\top) \leftarrow_{\text{R}} \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$. Let $\widetilde{\mathsf{Gen}}$ be a new generator algorithm that executes the original Gen and just outputs the public parameters.

$$\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (n,q,\chi) \leftarrow_{\mathrm{R}} \widetilde{\mathsf{Gen}}(1^\lambda),\ (\boldsymbol{A},\boldsymbol{y}^\top) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n, \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1((\boldsymbol{A},\boldsymbol{y}^\top),(n,q,\chi),1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ \boldsymbol{r},\boldsymbol{x} \leftarrow_{\mathrm{R}} \chi^n,\ x' \leftarrow_{\mathrm{R}} \chi, \\ \boldsymbol{a} := \boldsymbol{A}\boldsymbol{r} + \boldsymbol{x},\ a := \boldsymbol{y}^\top\boldsymbol{r} + x' + m_b \lfloor q/2 \rceil \quad \mathrm{rem}\ q, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2((\boldsymbol{a},a),(\boldsymbol{A},\boldsymbol{y}^\top),(n,q,\chi),aux,1^\lambda) \end{array} \right]$$

The success probability of the adversary in game 1 is at negligible distance from the success probability of the adversary in game 0. Otherwise, we could use it to break the decisional LWE assumption for n samples, as this adversary would correctly guess b with a significantly different probability when $(\boldsymbol{A}^\top, \boldsymbol{y})$ comes from a LWE distribution or when both elements are uniformly and independently distributed, and we could use it to distinguish the two scenarios with a probability greater than a random guess by a non-negligible amount.

- **Game 2**: the adversary $\mathcal{A}$ receives both a fake public key $(\boldsymbol{A},\boldsymbol{y}^\top) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$ and a partially fake encryption $(\boldsymbol{a}, a)$, where $\boldsymbol{a} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^n$ and $a := a' + m \lfloor q/2 \rceil$ rem q with $a' \leftarrow_{\mathrm{R}} \mathbb{Z}_q$.

$$\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (n,q,\chi) \leftarrow_{\mathrm{R}} \widetilde{\mathsf{Gen}}(1^\lambda),\ (\boldsymbol{A},\boldsymbol{y}^\top) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n, \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1((\boldsymbol{A},\boldsymbol{y}^\top),(n,q,\chi),1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ \boldsymbol{a} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^n,\ a' \leftarrow_{\mathrm{R}} \mathbb{Z}_q, \\ a := a' + m_b \lfloor q/2 \rceil \quad \mathrm{mod}\ q, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2((\boldsymbol{a},a),(\boldsymbol{A},\boldsymbol{y}^\top),(n,q,\chi),aux,1^\lambda) \end{array} \right]$$

Notice that in game 2 the vector $\boldsymbol{y}$ was already uniformly random, so the element that plays the role of the encryption,

$$\begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{r} + \begin{pmatrix} \boldsymbol{x} \\ x' \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ m_b \lfloor q/2 \rceil \end{pmatrix} \quad \mathrm{rem}\ q,$$

was constructed from $n+1$ LWE samples, while in game 3 it involves a uniformly random vector from $\mathbb{Z}^{n+1}$,

$$\begin{pmatrix} \boldsymbol{a} \\ a' \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ m_b \lfloor q/2 \rceil \end{pmatrix} \quad \mathrm{rem}\ q.$$

Analogously as before, under the assumption that the LWE problem with $n+1$

samples is hard, the difference between the success probability in games 2 and 3 is negligible.

- **Game 3**: the adversary $\mathcal{A}$ receives a fake $(\boldsymbol{A}, \boldsymbol{y}) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$ and a completely random fake encryption $(\boldsymbol{a}, a) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^n \times \mathbb{Z}_q$.

$$\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (n, q, \chi) \leftarrow_{\mathrm{R}} \widetilde{\mathsf{Gen}}(1^\lambda),\ (\boldsymbol{A}, \boldsymbol{y}^\intercal) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n, \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1((\boldsymbol{A}, \boldsymbol{y}^\intercal), (n, q, \chi), 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0, 1\}, \\ \boldsymbol{a} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^n,\ a \leftarrow_{\mathrm{R}} \mathbb{Z}_q, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2((\boldsymbol{a}, a), (\boldsymbol{A}, \boldsymbol{y}^\intercal), (n, q, \chi), aux, 1^\lambda) \end{array} \right]$$

It is immediate to check that the success probability of the adversary is the same in games 2 and 3 since anything plus a uniformly random distribution follows a uniformly random distribution, and therefore the adversary obtains the same information.

Since, except for a negligible difference, the adversary has the same advantage on game 0 than on game 3, where the inputs they get are completely independent of b, we can conclude that the presented encryption scheme has semantic security, that is, is IND-CPA secure, as long as the Gen algorithm outputs parameters that make the LWE problem hard. □

1.3.7 Ideal Lattices

The main downside of the LWE problem is that matrix $\boldsymbol{A}$ has a size quadratic in the dimension n, and this yields to a high communication cost for parameters that make the problem difficult enough. The good news is that this cost can be reduced to something linear in n if we restrict ourselves to a particular class of lattices.

Fixed a vector $\boldsymbol{f} = (f_0, \dots, f_{n-1})^\intercal \in \mathbb{Z}^n$ we define the transformation matrix $\boldsymbol{F}$ as

$$\left[\begin{array}{ccc|c} 0 & \cdots & 0 & -f_0 \\ \ddots & & & -f_1 \\ & \boldsymbol{Id}_{n-1} & & \vdots \\ & & \ddots & -f_{n-1} \end{array}\right].$$

Definition 1.31 (Ideal Lattice)**.** An *ideal lattice* is a lattice $\mathcal{L}(\boldsymbol{A})$ generated by a block matrix $\boldsymbol{A} = \left[\boldsymbol{A}^{(1)}\middle|\dots\middle|\boldsymbol{A}^{(m)}\right]$ whose blocks $\boldsymbol{A}^{(i)}$ are constructed from a vector $\boldsymbol{a}^{(i)} \in \mathbb{Z}^n$

and a transformation matrix $\boldsymbol{F}$ in the following way.

$$\boldsymbol{A}^{(i)} = \left[\boldsymbol{a}^{(i)}, \boldsymbol{F}\boldsymbol{a}^{(i)}, \ldots, \boldsymbol{F}^{n-1}\boldsymbol{a}^{(i)}\right].$$

These lattices are called ideal lattices as they can be seen as principal ideals in the polynomial ring $R = \mathbb{Z}[x]/\langle f(x)\rangle$, where $f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in \mathbb{Z}[x]$ is the monic polynomial given by the vector of the transformation matrix. As we said, we identify any other vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{Z}^n$ with a polynomial $v = v_1 + v_2x + \cdots + v_nx^{n-1} \in R$. It can be easily checked that, by construction, multiplying two polynomials a and b in the ring R is equivalent to multiply the matrix $\boldsymbol{A}$ constructed from the vector $\boldsymbol{a}$ with the vector $\boldsymbol{b}$. Then, vectors $\{\boldsymbol{v} \mid \boldsymbol{v} = \boldsymbol{A}\boldsymbol{b}\}$ are, when considered as polynomials, elements of the principal ideal $\langle a\rangle \subset R$.

We choose f to be $(1, 0, \ldots, 0)$, so that $f(x) = x^n + 1$, with n a power of 2, and work with the q-ary lattices $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, as it gives us nice security reductions [84].

Using this particular polynomial each block of the matrix $\boldsymbol{A}$ is an anti-cyclic integer matrix.

$$\boldsymbol{A}^{(i)} = \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & -a_n & \cdots & -a_3 \\ a_3 & a_2 & a_1 & \cdots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 \end{pmatrix}.$$

No algorithm is known to be able to take significant advantage of the structure in ideal lattices of this family to efficiently solve the problems presented before when restricted to their ideal cases [84].

Ring Short Integer Solution

Definition 1.32 (Ring Short Integer Solution – Ring-SIS$_{n\,q\,m\,\beta}$)**.** Let $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$ be a uniformly random vector of polynomials. The *Ring Short Integer Solution* (Ring-SIS) problem consists on finding a short not-zero vector of polynomials $\boldsymbol{e} = (e_1, \ldots, e_m) \in R_q^m$ such that $\|\boldsymbol{e}\| \leq \beta$ and $\langle \boldsymbol{a}, \boldsymbol{e}\rangle = \sum_{i=1}^m a_ie_i = 0$.

Definition 1.33 (Ring Inhomogeneous Short Integer Solution – Ring-ISIS$_{n\,q\,m\,\beta}$)**.** Let $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$ be a uniformly random vector of polynomials and $y \in R_q$. The *Ring Inhomogeneous Short Integer Solution* (Ring-ISIS) problem consists on finding a short not-zero vector of polynomials $\boldsymbol{e} = (e_1, \ldots, e_m) \in R_q^m$ such that $\|\boldsymbol{e}\| \leq \beta$ and $\langle \boldsymbol{a}, \boldsymbol{e}\rangle = \sum_{i=1}^m a_ie_i = y$.

There are again efficient reductions from the ideal version of the SVP to Ring-SIS [97].

Ring Learning With Errors

The LWE problem can also be particularized to a ring version, called *Ring Learning With Errors* (RLWE) [84].

Definition 1.34 (Ring Learning With Errors – $\mathrm{RLWE}_{n\,q\,m\,\chi}$)**.** Let n, q be integers (q usually prime), χ a discrete probability distribution in $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ (usually a discrete Gaussian distribution) and s a secret polynomial from R_q.

The RLWE distribution $\mathcal{R}_{s,\chi}$ over $R_q \times R_q$ is sampled choosing $a \leftarrow_{\mathrm{R}} R_q$ uniformly at random, a small error term $e \leftarrow_{\mathrm{R}} \chi$ from the noise distribution, and outputting the pair of polynomials $(a, b := a \cdot s + e) \in R_q \times R_q$.

Analogously to LWE, the goal of the *decisional-RLWE* problem will be to distinguish random linear equations, perturbed by a small amount of noise $\{(a_i, a_i \cdot s + e_i)\} \leftarrow_{\mathrm{R}} \mathcal{R}_{s,\chi}$, from truly uniform pairs $\{(a_i, u_i)\} \leftarrow_{\mathrm{R}} R_q \times R_q$. The goal of the *search-RLWE* problem will be to recover the secret $s \in R_q$ from arbitrarily (or depending on the version polynomially or finitely) many noisy products $\{(a_i, a_i \cdot s + e_i)\} \leftarrow_{\mathrm{R}} \mathcal{R}_{s,\chi}$. If the number of available samples is limited to m we denote the problem as $\mathrm{RLWE}_{\mathrm{n\,q\,m}\,\chi}$. Usually the error distribution χ is a discrete Gaussian distribution on $\mathbb{Z}^n$, that is $\chi = D_\sigma^n$, for a given parameter σ (recall we identify vectors with polynomials, so $e \in R_q$ sampled from D_σ^n would just mean that we have its coefficients independently sampled from a discrete Gaussian D_σ and then considered modulo q).

Certain instantiations of RLWE are supported by worst-case hardness theorems [83], related to the ideal version of the SVP. Taking as error distribution a Gaussian with parameter $\sigma \geq \omega(\sqrt{\log n})$, and for any ring, there exist a quantum reduction from the $\gamma(n)$-SVP to the RLWE problem to within $\gamma(n) = \mathcal{O}(\sqrt{n} \cdot q/\sigma)$. Additionally, the RLWE problem becomes no easier to solve even if the secret s is chosen from the error distribution, rather than uniformly [83].

Additional variants of the LWE exist, as the *Module Learning With Errors* (MLWE) problem, where the LWE problem is generalized over modules (problems are described using matrices whose elements are polynomials), showed to be polynomially equivalent to the RLWE problem in [7].

Cryptographic Applications of Ideal Lattices

Proposition 1.35. *Lyubashevsky, Peikert and Regev proposed in [84] the following encryption scheme based on the hardness of the RLWE problem.*

- Gen: *the key generator algorithm takes the security parameter* 1^λ *as input and outputs as public parameters a prime* q, *a power of two* n *and a parameter* σ *so that the* $\text{RLWE}_{n\,q\,D_\sigma^n}$ *problem is hard enough and errors are small with sufficient probability. Then a RLWE sample is defined as a public key, with* $a \leftarrow_R R_q$ *obtained uniformly at random,* $s, e \leftarrow_R D_\sigma^n$ *small elements obtained from the error distribution and* b *computed as* $b := a \cdot s + e$. *The secret key is therefore the secret* s.

$$(sk := s, pk := (a, b := a \cdot s + e); pp := (n, q, \sigma)) \leftarrow_R \mathsf{Gen}(1^\lambda)$$

- Enc: *the encryption algorithm takes as input a message* $m \in \{0,1\}^n$ *encoded as a polynomial in* R_q *with* 0 *or* 1 *coefficients, a public key* (a, b), *and the public parameters* pp. *It chooses small random elements* $r, e_u, e_v \leftarrow_R D_\sigma^n$ *and computes the ciphertext as two polynomials following the structure of RLWE samples, adding the escalated message to the second one,* $(u, v) := (a \cdot r + e_u, b \cdot r + e_v + \lfloor q/2 \rceil\, m) \in R_q \times R_q$.

$$(u := a \cdot r + e_u, v := b \cdot r + e_v + \lfloor q/2 \rceil\, m) \leftarrow_R \mathsf{Enc}(m; (a, b), pp)$$

- Dec: *the decryption algorithm takes as input a ciphertext* (u, v) *and uses the secret key* s *to compute a noisy message* $v - s \cdot u$ *that can then be rounded to the original message.*

$$\lfloor v - s \cdot u \rceil_{q/2} \leftarrow \mathsf{Dec}((u, v), s; pp)$$

The above cryptosystem is IND-CPA secure assuming the hardness of the decisional-RLWE problem.

Proof. Correctness comes from the fact that a and b are not random independent polynomials, which would make $a \cdot r + e_u$ and $b \cdot r + e_v$ two samples from $\mathcal{R}_{r,D_\sigma^n}$ and the message irrecoverable. Provided that (a, b) is a RLWE sample and the secret is the secret key we can remove the polynomials masking the message, except for some small error terms.

$$\begin{aligned} v - s \cdot u &= b \cdot r + e_v + \left\lfloor \frac{q}{2} \right\rceil m - s(a \cdot r + e_u) \\ &= s \cdot a \cdot r + e \cdot r + e_v + \left\lfloor \frac{q}{2} \right\rceil m - s \cdot a \cdot r - s \cdot e_u \\ &= e \cdot r + e_v - s \cdot e_u + \left\lfloor \frac{q}{2} \right\rceil m \\ &\approx \left\lfloor \frac{q}{2} \right\rceil m. \end{aligned}$$

If σ is small enough that $\|e \cdot r + e_v - s \cdot e_u\|_\infty < q/4$, except with negligible probability, when we round we recover the original message, again except with negligible probability. As an alternative we can also use a bounded error distribution so that the probability of a decryption error is identically zero.

In order to demonstrate that the scheme is IND-CPA secure in this case we are going to prove it by contradiction. Assume $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a PPT adversary such that the following quantity is non-negligible in λ.

$$\left| \Pr\left[z_{\mathcal{A}} = z \;\middle|\; \begin{array}{c} (s,(a,b);(n,q,\sigma)) \leftarrow_{\mathrm{R}} \mathsf{Gen}\left(1^{\lambda}\right), \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1\left((a,b),(n,q,\sigma),1^{\lambda}\right), \\ z \leftarrow_{\mathrm{R}} \{0,1\},\ (u,v) \leftarrow_{\mathrm{R}} \mathsf{Enc}\left(m_z;(a,b),(n,q,\sigma)\right), \\ z_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2\left((u,v),(a,b),(n,q,\sigma),aux,1^{\lambda}\right) \end{array} \right] - {}^1\!/_2 \right| \notin \mathrm{negl}(\lambda)$$

Let $(a_1, c_1), (a_2, c_2) \in R_q \times R_q$ be two pairs that can be uniformly random pairs or RLWE samples. If they are random polynomials then

$$\left| \Pr\left[z_{\mathcal{A}} = z \;\middle|\; \begin{array}{c} (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1\left((a_1,a_2),(n,q,\sigma),1^{\lambda}\right), \\ z \leftarrow_{\mathrm{R}} \{0,1\}, \\ u := c_1,\ v := c_2 + \lfloor q/2 \rceil m_z, \\ z_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2\left((u,v),(a_1,a_2),(n,q,\sigma),aux,1^{\lambda}\right) \end{array} \right] - {}^1\!/_2 \right| = 0.$$

As the uniformly random element c_2 completely masks the chosen message the output $z_{\mathcal{A}}$ is independent of z, and the probability of correctly guessing the bit is exactly ${}^1\!/_2$.

If (a_1, c_1) and (a_2, c_2) are real RLWE samples there are two possibilities. If the difference with ${}^1\!/_2$ is non-negligible we can use this adversary to solve the decisional-RLWE with non-negligible probability, because the event $z = z_{\mathcal{A}}$ happens with a significantly different probability in both scenarios. If the probability is at a negligible distance from ${}^1\!/_2$ it means that the adversary $\mathcal{A}$ worked well when the public key (a, b) was a real RLWE sample, but not when the elements (a_1, a_2) playing the same role are two random independent polynomials, which also gives us a way to distinguish random pairs from RLWE samples playing the same game this time choosing a_2 to be the element that can be a uniformly random polynomial or a RLWE sample of the lattice generated by a_1.

Notice that we do not know a priori which of those possibilities would be true, but all of them would allow us to distinguish RLWE samples from random elements, which contradicts the hardness assumption and concludes the proof. □

Observation 1.36. Notice the rounding step is an important part of the decryption process. Someone wanting to prove that the decryption of a ciphertext is a given value without revealing their own secret key cannot omit the rounding revealing

$v - s \cdot u$ and proving in zero-knowledge the fact that is has been computed from a secret s that corresponds to their public key. Even if the ZKPoK would reveal no information itself the error terms $e \cdot r + e_v - s \cdot e_u$ do depend on the secret key s and would partially leak the secret. Observe that these error terms also depend on e, which has to remain secret too because knowing e we could directly compute s from a and b. The fact that this additional non-linear rounding operation is an important part of the decryption process makes the realization of correct decryption proofs (or the generalization to threshold decryption) more cumbersome.

Besides encryption schemes the RLWE assumption can also be directly used to design lattice based commitment schemes, again hiding a message by adding a RLWE sample. Nevertheless, provided that one of the main contributions of this dissertation is precisely a new lattice-based commitment scheme using the RLWE problem as hardness assumption, we omit here that discussion and leave the details for Chapter 3.

1.3.8 Zero-Knowledge Proofs of Knowledge with Lattices

As we have seen one particular characteristic of lattice-based cryptography is the important role small norm elements play. Many ZKPoKs require to prove that some random element satisfying some relation exists, and working with lattice-based cryptography we usually need to prove that some random element satisfying some relations exists and has small norm. In order to achieve this, several strategies have been used in the literature, and we are going to discuss the most relevant two.

Consider an instance of the ISIS problem. Given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\boldsymbol{y} \in \mathbb{Z}_q^n$ we want to prove knowledge of $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ and $\|\boldsymbol{x}\|_\infty \leq \beta$.

Let us present in Protocol 1.2 a first attempt following the structure of a standard Σ-protocol, a 3-move interactive proof with special soundness, to see what issues arise. To do so we sample another vector $\widehat{\boldsymbol{x}} \in \mathbb{Z}_q^m$ to hide the original secret, ask for a challenge and finally reveal a combination of this masking element and the secret, mimicking the Schnorr protocol for the DLog problem [105].

However, this simple approach fails as a proof of knowledge for the ISIS problem. From two accepted answers to different challenges we would get a vector $(\alpha - \alpha')^{-1}(\boldsymbol{z} - \boldsymbol{z}')$ that is a preimage of $\boldsymbol{y}$, but it might not have a small norm. We could try to ensure that the norm of $\boldsymbol{z}$ is small, but to do so the masking element $\widehat{\boldsymbol{x}}$ and the challenge α should also be small. Then $\boldsymbol{z}$ would not be uniformly distributed, and would reveal information about $\boldsymbol{x}$. Besides that, if we ensure $\boldsymbol{z} - \boldsymbol{z}'$ has a small norm we would be proving knowledge of a short preimage of $(\alpha - \alpha')\boldsymbol{y}$ instead of $\boldsymbol{y}$, and we could no longer bound the norm of the preimage when multiplying by $(\alpha - \alpha')^{-1}$.

To solve at least some of these issues a more involved protocol is needed.

Protocol 1.2 Attempt of an ISIS protocol

```
   $\mathcal{P}(A, y; x)$                                              $\mathcal{V}(A, y)$
1: $\widehat{x} \leftarrow_R \mathbb{Z}_q^m$
2: $\widehat{y} = A\widehat{x}$
                                  $\xrightarrow{\widehat{y}}$
3:                                                                  $\alpha \leftarrow_R \mathbb{Z}_q$
                                  $\xleftarrow{\alpha}$
4: $z = \widehat{x} + \alpha x$
                                  $\xrightarrow{z}$
5:                                                                  $Az \overset{?}{=} \widehat{y} + \alpha y$
```

Fiat-Shamir with Aborts

In order to avoid some of these problems, the prover can just use a small masking element even if adding it does not completely randomize the secret as would be the case with a uniformly random masking element, ask for a small α, and abort whenever they are going to reveal too much information. This strategy is called *rejection sampling*.

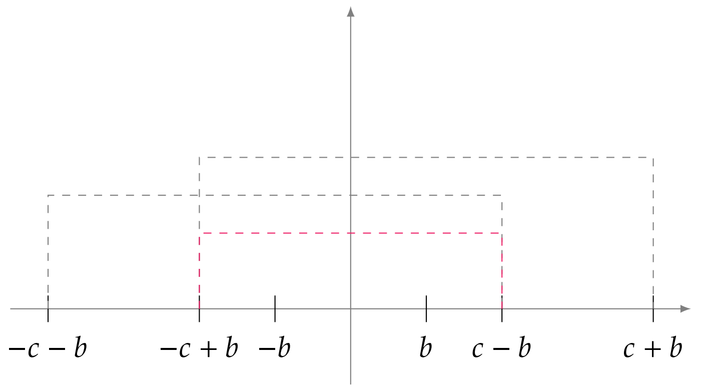
Imagine a very simplified one dimensional example where $|\alpha x|$ is known to be bounded by $b \in \mathbb{Z}_{>0}$ and $\widehat{x}$, intended to mask αx, is sampled from a uniform distribution $[-c, c] \subset \mathbb{Z}$ (for some $c > b$). Then, for a fixed αx the distribution of $z := \widehat{x} + \alpha x$ would be uniform in an interval of length $2c$ (with $2c+1$ elements) inside $[-c-b, c+b]$. The extreme cases being $[-c-b, c-b]$ and $[-c+b, c+b]$ if αx is $-b$ or b respectively.

The prover can abort if z does not lie in $[-c+b, c-b]$, the common elements of all possible intervals (see Figure 1.11, the intersection is not empty because $c > b$). This way, conditioned to the fact that we have not aborted, z is uniformly distributed in the interval $[-c+b, c-b]$ (and it does not depend on αx). Moreover, the probability of aborting is constant, equal to $2b/(2c+1)$. Here we have a trade-off between the probability of aborting and the bounds for z we obtain from the extractor.

A similar strategy can be followed using discrete Gaussian error distributions instead of uniform distributions.

Theorem 1.37 (Vadim Lyubashevsky [78]). *Let V be a subset of $\mathbb{Z}^l$ in which all elements have norms less than T, and let h be a probability distribution over V. We denote by $D_{v,\sigma}$ the discrete Gaussian distribution centered on v (sampled as the usual discrete Gaussian but adding v). Then, for any constant M, there exists a $\sigma = \widetilde{\Theta}(T)$ such that the output distribution of the following algorithms $\mathcal{A}$ and $\mathcal{F}$ are statistically close.*

Figure 1.11 Fiat-Shamir with aborts



$\mathcal{A}$:

$v \leftarrow_R h; \quad z \leftarrow_R D^l_{v,\sigma};$

output (z, v) *with probability:*

$$\min\left(\frac{1}{M}\exp\left(\frac{\|v\|_2^2 - 2\langle z, v\rangle}{2\sigma^2}\right), 1\right)$$

$\mathcal{F}$:

$v \leftarrow_R h; \quad z \leftarrow_R D^l_{\sigma};$

output (z, v) *with probability:*

$$\frac{1}{M}$$

Moreover, the probability that $\mathcal{A}$ outputs something is exponentially close to $1/M$.

With these aborts the probability distribution of the answer is statistically close to a probability distribution centered in the origin and non-dependent on the secret multiplied by the challenge (as the z from the one-dimensional example did not depend on αx). The expected number of iterations until there is no abort is a constant M, that depends on σ, and this σ is going to determine the size of the witness we would be able to extract (usually larger than the size of the secret element known by the prover but hopefully small enough to be useful). Observe that we also still have a constant $(\alpha - \alpha')$, but this is sufficient for some proofs [78].

Stern-Based Proofs

Many lattice-based ZKPoKs are based on the fruitful paper of Stern [116] for a code-based identification scheme, presented a few years before the possibility of using lattice problems to build cryptographic primitives was suggested by Ajtai [5]. In many ways lattice and code problems are very similar, and in some cases we can find analogous problems just substituting a measure like Hamming weight with the euclidean norm. This implies that code-based zero-knowledge protocols face similar problems, and solutions for one setting can usually be adapted to the other. We provide here a brief introduction to the notation, definitions and problems from coding theory necessary to follow Stern-based protocols.

The seminal identification protocol [116] proposed by Stern in 1990 was a ZKPoK of a solution of an instance of the *Syndrome Decoding Problem* (SDP). The syndrome works as a public key and the user can authenticate themself interacting with a verifier and proving knowledge of a solution (a binary vector with small Hamming weight). Let us introduce and define all these code concepts.

Definition 1.38 (Binary linear code).
A *binary linear code*, denoted C, is the kernel of a binary check matrix $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$.

$$C := \left\{ \boldsymbol{w} \in \mathbb{F}_2^n \;\middle|\; \boldsymbol{H}\boldsymbol{w} = 0 \right\}.$$

It can also be defined as the image of a generator matrix $\boldsymbol{G} \in \mathbb{F}_2^{n\times k}$.

$$C := \left\{ \boldsymbol{w} \in \mathbb{F}_2^n \;\middle|\; \boldsymbol{w} = \boldsymbol{G}\boldsymbol{x},\ \boldsymbol{x} \in \mathbb{F}_2^k \right\}.$$

Observation 1.39. Notice the similarities between a binary linear code and a q-ary lattice given in its $\Lambda^\perp$ or Λ form.

A classical problem in coding theory is to recover a codeword $\boldsymbol{w}$ when all we know is a perturbed version $\boldsymbol{v} := \boldsymbol{w} + \boldsymbol{e}$, where $\boldsymbol{e}$ is some error with low Hamming weight. The problem can be stated in terms of the check matrix.

$$\begin{aligned} \boldsymbol{H}\boldsymbol{v} &= \boldsymbol{y} \\ \boldsymbol{H}(\boldsymbol{w} + \boldsymbol{e}) &= \boldsymbol{y} \\ \boldsymbol{H}\boldsymbol{e} &= \boldsymbol{y} \end{aligned}$$

That is, we have to find a low weight vector $\boldsymbol{e}$ with the same *syndrome* $\boldsymbol{y}$ than $\boldsymbol{v}$, to be able to recover $\boldsymbol{w} = \boldsymbol{v} - \boldsymbol{e}$. If the restrictions on the weight of the error are strong enough then the solution is unique, but finding it is a well studied hard problem [21], analogous to the ISIS problem for lattices.

Definition 1.40 (Syndrome Decoding problem – SD). Given a binary matrix $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$, a binary vector $\boldsymbol{y} \in \mathbb{F}_2^{n-k}$ and an integer w, the SDP consists on deciding whether or not there exists a binary vector $\boldsymbol{e}$ such that $\boldsymbol{H}\boldsymbol{e} = \boldsymbol{y}$ and $\|\boldsymbol{e}\|_{\mathsf{H}} \le w$.

If we choose a random $\boldsymbol{e}$ of fixed small Hamming weight w it can be used as a secret key $sk := \boldsymbol{e}$, and its syndrome $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{e}$ can be published as a public key $pk := \boldsymbol{y}$

for an identification protocol. In order to identify themself the user only needs to prove knowledge of a valid $\boldsymbol{e}$.

Then, the zero-knowledge interactive identification scheme by Stern, that we reproduce in Protocol 1.3, allows a prover to convince a verifier that given a parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$ and a syndrome $\boldsymbol{y} \in \mathbb{F}_2^{n-k}$, they know a binary vector $\boldsymbol{e} \in \mathbb{F}_2^n$ of small fixed Hamming weight $\|\boldsymbol{e}\|_\mathsf{H} = w$ such that it has this syndrome $\boldsymbol{y} = \boldsymbol{He}$.

The original Stern's idea to hide the small vector $\boldsymbol{e}$ was to use a masking vector $\boldsymbol{x} \leftarrow_\mathrm{R} \mathbb{F}_2^n$, a masking syndrome $\boldsymbol{y}' \in \mathbb{F}_2^{n-k}$ (an honest prover will compute $\boldsymbol{y}' := \boldsymbol{Hx}$) and a permutation $\pi \leftarrow_\mathrm{R} \mathfrak{S}_n$ (we denote by $\mathfrak{S}_n$ the symmetric group of n elements, i.e. the set of permutations of n elements). Notice that $\boldsymbol{x}+\boldsymbol{e}$ reveals no information about $\boldsymbol{e}$, while $\pi(\boldsymbol{e})$ only reveals its Hamming weight, which is precisely the information we want to disclose. Then the prover shows that the masking elements are properly computed, the linear relation is satisfied and the Hamming weight is correct:

(a) the syndrome of $\boldsymbol{x}$ is $\boldsymbol{y}'$,

(b) the syndrome of $\boldsymbol{x} + \boldsymbol{e}$ is $\boldsymbol{y}' + \boldsymbol{y}$,

(c) the Hamming weight of $\pi(\boldsymbol{e})$ is w.

To do so they build three commitments, $c_1 \leftarrow_\mathrm{R} \mathsf{Com}(\pi, \boldsymbol{y}')$, $c_2 \leftarrow_\mathrm{R} \mathsf{Com}(\pi(\boldsymbol{x}))$, $c_3 \leftarrow_\mathrm{R} \mathsf{Com}(\pi(\boldsymbol{x}+\boldsymbol{e}))$, in such a way that opening each pair of commitments (as it is detailed in Protocol 1.3 depending on the challenge given by the verifier) it is possible to verify each of the properties. Opening c_1 and c_2 one could check (a), opening c_1 and c_3 one could check (b) and finally opening c_2 and c_3 one could check (c).

All the properties combined imply that there is an $\boldsymbol{e}$ with Hamming weight w and syndrome $\boldsymbol{y}$, and it can be extracted from valid openings to all three commitments.

However, a cheating prover could build commitments that satisfy two of the three properties (a), (b) and (c). Let us show how each pair of properties on its own can be satisfied with elements computed without knowing a valid $\boldsymbol{e}$.

For example if a vector $\widehat{\boldsymbol{e}}$ of arbitrary Hamming weight and $\boldsymbol{y}$ syndrome is used with $\boldsymbol{x} \leftarrow_\mathrm{R} \mathbb{F}_2^n$ and $\boldsymbol{y}' = \boldsymbol{Hx}$ the properties (a) and (b) hold. If a vector $\widehat{\boldsymbol{e}}$ with Hamming weight w and arbitrary syndrome is used, and $\boldsymbol{y}' = \boldsymbol{Hx}$ with $\boldsymbol{x} \leftarrow_\mathrm{R} \mathbb{F}_2^n$, properties (a) and (c) still hold. Finally, if we use a vector $\widehat{\boldsymbol{e}}$ with Hamming weight w and arbitrary syndrome and compute $\boldsymbol{y}' = \boldsymbol{H}(\boldsymbol{x} + \widehat{\boldsymbol{e}}) - \boldsymbol{y}$ then properties (b) and (c) hold.

Protocol 1.3 Stern's protocol

```
$\mathcal{P}(\boldsymbol{H}, \boldsymbol{y}; \boldsymbol{e})$                                              $\mathcal{V}(\boldsymbol{H}, \boldsymbol{y})$
1:  $\pi \leftarrow_R \mathfrak{S}_n, \quad \boldsymbol{x} \leftarrow_R \mathbb{F}_2^n$
2:  $(c_1, o_1) \leftarrow_R \mathsf{Com}(\pi, \boldsymbol{H}\boldsymbol{x})$
3:  $(c_2, o_2) \leftarrow_R \mathsf{Com}(\pi(\boldsymbol{x}))$
4:  $(c_3, o_3) \leftarrow_R \mathsf{Com}(\pi(\boldsymbol{x} + \boldsymbol{e}))$
                              $\xrightarrow{c_1, c_2, c_3}$
5:                                                            $b \leftarrow_R \{0, 1, 2\}$
                              $\xleftarrow{b}$
6:  if $b = 0$ then
7:      $\widetilde{\pi} = \pi$
8:      $\widetilde{\boldsymbol{x}} = \pi(\boldsymbol{x})$
9:      $\widetilde{o}_1 = o_1, \quad \widetilde{o}_2 = o_2$
10:     $ans = (\widetilde{\pi}, \widetilde{\boldsymbol{x}}, \widetilde{o}_1, \widetilde{o}_2)$
11: end if
12: if $b = 1$ then
13:     $\widetilde{\pi} = \pi$
14:     $\widetilde{\boldsymbol{z}} = \pi(\boldsymbol{x} + \boldsymbol{e})$
15:     $\widetilde{o}_1 = o_1, \quad \widetilde{o}_3 = o_3$
16:     $ans = (\widetilde{\pi}, \widetilde{\boldsymbol{z}}, \widetilde{o}_1, \widetilde{o}_3)$
17: end if
18: if $b = 2$ then
19:     $\widetilde{\boldsymbol{x}} = \pi(\boldsymbol{x})$
20:     $\widetilde{\boldsymbol{z}} = \pi(\boldsymbol{x} + \boldsymbol{e})$
21:     $\widetilde{o}_2 = o_2, \quad \widetilde{o}_3 = o_3$
22:     $ans = (\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{z}}, \widetilde{o}_2, \widetilde{o}_3)$
23: end if
                              $\xrightarrow{ans}$
24:                                                           if $b = 0$ then
25:                                $\overset{?}{\mathsf{Ver}}(c_1, (\widetilde{\pi}, \boldsymbol{H}\widetilde{\pi}^{-1}(\widetilde{\boldsymbol{x}})), \widetilde{o}_1)$
26:                                               $\overset{?}{\mathsf{Ver}}(c_2, \widetilde{\boldsymbol{x}}, \widetilde{o}_2)$
27:                                                           end if
28:                                                           if $b = 1$ then
29:                            $\overset{?}{\mathsf{Ver}}(c_1, (\widetilde{\pi}, \boldsymbol{H}\widetilde{\pi}^{-1}(\widetilde{\boldsymbol{z}}) - \boldsymbol{y}), \widetilde{o}_1)$
30:                                               $\overset{?}{\mathsf{Ver}}(c_3, \widetilde{\boldsymbol{z}}, \widetilde{o}_3)$
31:                                                           end if
32:                                                           if $b = 2$ then
33:                                               $\overset{?}{\mathsf{Ver}}(c_2, \widetilde{\boldsymbol{x}}, \widetilde{o}_2)$
34:                                               $\overset{?}{\mathsf{Ver}}(c_3, \widetilde{\boldsymbol{z}}, \widetilde{o}_3)$
35:                                               $\|\widetilde{\boldsymbol{z}} - \widetilde{\boldsymbol{x}}\|_{\mathsf{H}} \overset{?}{=} w$
36:                                                           end if
```

We can see how the commitments could be computed in each case following this strategy.

(a) & (b) Obtain $\widehat{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{H}\widehat{\boldsymbol{e}} = \boldsymbol{y}$, $\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_2^n$ and $\pi \leftarrow_{\mathrm{R}} \mathfrak{S}_n$, then
$c_1 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi, \boldsymbol{Hx})$, $\quad c_2 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x}))$, $\quad c_3 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x} + \widehat{\boldsymbol{e}}))$.

(a) & (c) Obtain $\widehat{e} \leftarrow_{\mathrm{R}} \left\{z \in \mathbb{F}_2^n \;\middle|\; \|z\|_{\mathsf{H}} = w\right\}$, $\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_2^n$ and $\pi \leftarrow_{\mathrm{R}} \mathfrak{S}_n$, then
$c_1 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi, \boldsymbol{Hx})$, $\quad c_2 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x}))$, $\quad c_3 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x} + \widehat{\boldsymbol{e}}))$.

(b) & (c) Obtain $\widehat{e} \leftarrow_{\mathrm{R}} \left\{z \in \mathbb{F}_2^n \;\middle|\; \|z\|_{\mathsf{H}} = w\right\}$, $\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_2^n$ and $\pi \leftarrow_{\mathrm{R}} \mathfrak{S}_n$, then
$c_1 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi, \boldsymbol{H}(\boldsymbol{x} + \widehat{\boldsymbol{e}}) - \boldsymbol{y})$, $\quad c_2 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x}))$, $\quad c_3 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi(\boldsymbol{x} + \widehat{\boldsymbol{e}}))$.

We know each of the three properties can be verified opening two commitments, but for every pair of properties we have shown a strategy that would allow a malicious prover to open these two pairs of commitments verifying the relations (it would fail for the remaining pair of commitments related to the third condition). This means there are adversaries with at least $2/3$ cheating probability. We can however prove this is the maximum success probability of a malicious adversary via a generalization of the previously mentioned special soundness.

The *k-Special Soundness* property means that a prover able to answer k challenges is honest, as in this case a witness could be extracted. This particular protocol is 3-special sound because from answers to all the challenges we could directly reconstruct the witness, as properties (a), (b) and (c) specifically characterize a witness and the response to each challenge verifies one of them (and the binding property of the commitment used ensures that the opened elements for each of the challenges are the same). It is not 2-special sound because we already know that a malicious prover can perfectly prepare its initial messages to correctly answer (any) two challenges. It can be seen that a 3-special sound protocol with 3 challenges is sound with a soundness error of $2/3$. To convince the verifier of the fact that the prover is not cheating, the same protocol can be repeated as many times as the verifier considers.

We are going to present more detailed and formal arguments of this kind of soundness notion in Chapter 3 where we are introducing yet another soundness definition that generalizes the concept of k-special soundness for $(2\mu + 1)$-moves interactive protocols.

Many ideas have been later proposed to adapt this scheme to different purposes, either a dual version with the generator matrix of the code, more efficient schemes with a particular subset of codes, q-ary codes and lattices. As we present a further improvement of the existing techniques in Chapter 3 we then devote Section 3.2 in its entirety to detail the variants from which we specifically benefit.

1.4 Probability Preliminaries

We have already mentioned in Section 1.2 the main strategies to prove a scheme is secure, identifying the event that we want to avoid and ensuring the success probability of any PPT adversary is as small as we want. If we are only interested in proving security asymptotically we can disregard constant factors from the definitions, but given that we wouldx like to faithfully compare our proposals with other schemes we need to take these details into account.

For that reason we formalize here all the definitions and strategies that allow us to ensure the security of our schemes, using statistical properties or computational assumptions.

1.4.1 The Advantage of an Adversary

We have previously mentioned the concept of *advantage* of an adversary, measuring the event that we want to avoid. For example, the advantage could just be the probability of the undesired event.

Definition 1.41 (Adv_{KR})**.**
The advantage of an adversary $\mathcal{A}$ in a key recovery attack is defined as follows.

$$Adv_{KR}(\mathcal{A}) := \Pr\left[sk = sk_{\mathcal{A}} \;\middle|\; (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda),\ sk_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(pk, pp, 1^\lambda)\right].$$

In other occasions the advantage is not defined as the probability of some event, because we are interested in the difference of the probabilities of two alternative scenarios. That is what happens with the indistinguishability games we use to prove that an encryption scheme is IND-CPA, a commitment is hiding or a ZKPoK is zero-knowledge. The natural way of defining these properties is ensuring that the adversary has a success probability that is not significantly better than a purely random guess that succeeds with probability $1/2$. We bound this distance to $1/2$ in absolute value because an adversary that almost always chooses the wrong answer can be used to break the security, as we would only need to define another adversary that simply does the contrary.

1.4.2 Advantage in an Indistinguishability Game

We have already mentioned all these indistinguishability properties, but we can properly name the advantages here to see that all of them follow the same approach.

Recall that whenever we allow the adversary to define the two scenarios they would later try to distinguish we divide it into two algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, so that the first sets the information that determines the scenarios and the second outputs the bit with their guess (modeling with an auxiliary variable *aux* all the information the adversary might have precomputed in the first phase).

Definition 1.42 ($Adv_{\mathsf{IND-CPA}}$)**.** Advantage of $(\mathcal{A}_1, \mathcal{A}_2)$ in an *IND-CPA* attack against an encryption scheme, compared with a random guess.

$$Adv_{\mathsf{IND-CPA}}(\mathcal{A}) := \left| \Pr \left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (sk, pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1(pk, pp, 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ c \leftarrow_{\mathrm{R}} \mathsf{Enc}(m_b; pk, pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(c, pk, pp, aux, 1^\lambda) \end{array} \right] - 1/2 \right| .$$

The same kind of advantage can be defined for commitment schemes.

Definition 1.43 (Adv_{Com})**.** Advantage of $(\mathcal{A}_1, \mathcal{A}_2)$ in an attack against the computational hiding property of a commitment scheme, compared with a random guess.

$$Adv_{\mathsf{Com}}(\mathcal{A}) := \left| \Pr \left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (pk; pp) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1(pk, pp, 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ (c, o) \leftarrow_{\mathrm{R}} \mathsf{Com}(m_b; pk, pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(c, pk, pp, aux, 1^\lambda) \end{array} \right] - 1/2 \right| .$$

And also for the zero-knowledge property of a ZKPoK. Consider in this case a challenger Cha that has oracle access to the prover $\mathcal{P}$, the verifier $\mathcal{V}$ and the simulator $\mathcal{S}$, and computes both a real transcript tr_0 executing the protocol with $\mathcal{P}$ and $\mathcal{V}$ and a simulated transcript tr_1 with $\mathcal{S}$. As usual all public information is going to be contained in our model in some public parameters pp outputted by a generator Gen that would define the statement x. Additionally, the generator also outputs the witness w we are going to use for the studied ZKPoK.

Definition 1.44 (Adv_{ZK})**.** Advantage of an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ in an attack against the *Zero-Knowledge* property of a ZKPoK, compared with a random guess.

$$Adv_{\mathsf{ZK}}(\mathcal{A}) := \left| \Pr \left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} (pp, w) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ tr_0 \leftarrow_{\mathrm{R}} \mathsf{Cha}^{\mathcal{P},\mathcal{V}}(pp, w), \\ tr_1 \leftarrow_{\mathrm{R}} \mathsf{Cha}^{\mathcal{S}}(pp), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(tr_b, pp, 1^\lambda) \end{array} \right] - 1/2 \right| .$$

Generic Indistinguishability Game

In order to simplify the analysis and to be able to prove general propositions that we could latter use with the previous definitions we define an additional generic indistinguishability game, where all the relevant information is generated by a generator Gen that outputs the public parameters pp. From this point let as also assume that 1^λ is always part of the public parameters pp, this way we can keep formalities without adding it as supplementary input to every adversary.

The goal of the adversary $\mathcal{A}$ (to simplify the proofs we now consider a single algorithm, but we keep in mind the possibility of allowing the adversary to participate defining the game) is then to distinguish between two challengers, Cha_0 and Cha_1. One of the challengers is selected at random and produces a challenge c that $\mathcal{A}$ analyzes to try to determine if it was produced by Cha_0 or Cha_1.

Definition 1.45 (Adv_{IND})**.** Advantage of ($\mathcal{A}$) in a generic indistinguishability game, compared with a random guess.

$$Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) := \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_b(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] - {}^1\!/_2 \right|.$$

Alternatively, we can also deal with the indistinguishability of two scenarios if we ensure that the output of any adversary is essentially the same in both of them. We can formalize this idea asking the adversary to output a bit $b_{\mathcal{A}} \in \{0,1\}$ and comparing the probability of $b_{\mathcal{A}} = 1$ in both cases.

Definition 1.46 (Adv'_{IND})**.** Alternative definition of the advantage of an adversary $\mathcal{A}$ in a generic indistinguishability game, comparing both scenarios.

$$\underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv'}(\mathcal{A}) := \left| \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_0(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] - \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] \right|.$$

This definition also captures the notion of advantage against indistinguishability, because we understand that an adversary is able to distinguish between the two scenarios if the probability distribution of its output depends on the scenario. We measure this dependency as the difference of probabilities for outputting 1, but notice it is also the difference of probabilities of outputting 0.

Both Definitions 1.45 and 1.46 are useful, as there are propositions more naturally defined with each of them. We can always choose which to use because they are equivalent except for a constant, in this case $Adv'_{\mathsf{IND}} = 2\,Adv_{\mathsf{IND}}$.

Lemma 1.47 ($Adv'_{\mathsf{IND}} = 2\,Adv_{\mathsf{IND}}$)**.** The alternative definition of the indistinguishability advantage $Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}$ is exactly twice the original advantage $Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}$.

$$\underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv'}(\mathcal{A}) = 2 \cdot \underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv}(\mathcal{A})$$

Proof. We can directly prove $Adv_{\mathsf{IND}} = {}^1/_2\,Adv'_{\mathsf{IND}}$ using the law of total probability applied to b, knowing that it would be 0 or 1 with probability ${}^1/_2$.

$$\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_b(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] =$$

$$= \frac{1}{2}\Pr\left[b_{\mathcal{A}} = 0 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_0(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] + \frac{1}{2}\Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right]$$

$$= \frac{1}{2}\left(1 - \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_0(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right]\right) + \frac{1}{2}\Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right]$$

$$= \frac{1}{2} + \frac{1}{2}\left(\Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right] - \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ c \leftarrow_{\mathrm{R}} \mathsf{Cha}_0(pp), \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(c,pp) \end{array} \right]\right)$$

From which we can immediately deduce what we wanted subtracting ${}^1/_2$ on both sides and taking the absolute value.

$$\underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv}(\mathcal{A}) = \frac{1}{2}\underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv'}(\mathcal{A})$$

□

Computational Indistinguishability Game

The most usual strategy is to relate these probabilities (via reductions or sequences of games) with a probability that (possibly under certain assumptions) is negligible in the security parameter.

It might be the case that we precisely assume it is computationally hard to distinguish the distributions (pp, c_0) and (pp, c_1), where $c_b \leftarrow_{\mathrm{R}} \mathsf{Cha}_b$ is the output of the challenger in the b case. For example the decisional $\mathrm{RLWE}_{n\,q\,m\,\chi}$ assumption means precisely that the advantage of any PPT adversary is negligible in the security parameter in a game where $c_0 \leftarrow_{\mathrm{R}} \mathcal{R}^m_{s,\chi}$ and $c_1 \leftarrow_{\mathrm{R}} \mathcal{U}$, where $\mathcal{U}$ is the uniform distribution in the appropriate space $(R^m_q)^2$.

Using the language of advantages of adversaries we can reformulate the decisional-RLWE assumption. Let pp be the public parameters that contain all the information regarding the specifications of the ring, the number of samples, the secret element and noise distribution. We denote the dependence of the distributions on these parameters as $\mathcal{R}_{pp}$ and $\mathcal{U}_{pp}$.

Definition 1.48 (Adv'_{RLWE})**.**

Advantage of an adversary $\mathcal{A}$ against the decisional-RLWE problem.

$$Adv'_{\mathrm{RLWE}}(\mathcal{A}) := \left| \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ r \leftarrow_{\mathrm{R}} \mathcal{R}_{pp}, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(r, pp) \end{array} \right] - \Pr\left[b_{\mathcal{A}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda), \\ r \leftarrow_{\mathrm{R}} \mathcal{U}_{pp}, \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}(r, pp) \end{array} \right] \right| .$$

Sometimes the reduction we need involves two similar games but with a different nature. That is the case of the IND-CPA property of encryption schemes and the hiding property of commitment schemes, that reflect the inability of efficient adversaries for extracting any information about the messages from its impotence distinguishing two encryptions or commitments to two messages (even if the messages have been chosen by themselves), as it has been described in Definitions 1.42 and 1.43.

The most common strategy in lattice-based cryptography is to mask the message with a LWE/RLWE/MLWE sample. Without entering into details about how the message is encoded for each application we can define an example abusing notation and letting $c := m + r$, with $r \leftarrow_{\mathrm{R}} \mathcal{R}_{pp}$, be the output of some general primitive Alg that we want to either prove IND-CPA secure or hiding. Security for this simplified scheme would be based on the pseudorandomness of RLWE samples (in the encryption scheme we have already mentioned in Section 1.3 that this is used two consecutive times, but, for example, the commitment scheme [19] directly follows the idea behind this simplified example).

Proposition 1.49 (Reduction to the advantage of decisional-RLWE)**.** *The advantage against the indistinguishability of the previously defined example, compared with a random guess, is bounded by the advantage of a similarly powerful adversary against the decisional-RLWE, comparing the two scenarios.*

$$Adv_{\mathsf{Alg}}(\mathcal{A}) \leq \max_{\mathcal{B}} \left\{ Adv'_{\mathrm{RLWE}}(\mathcal{B}) \right\} .$$

Proof. We can prove it just manipulating the absolute values and the probabilities.

$$Adv_{\mathsf{Alg}}(\mathcal{A}) =$$

$$= \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ \begin{pmatrix} m_0, m_1, \\ aux \end{pmatrix} \leftarrow_R \mathcal{A}_1(pp), \\ b \leftarrow_R \{0,1\}, \\ r \leftarrow_R \mathcal{R}_{pp},\ c := m_b + r, \\ b_{\mathcal{A}} \leftarrow_R \mathcal{A}_2(c, aux, pp) \end{array} \right] - 1/2 \right|$$

We can subtract and add the probability of guessing the scenario when $r \leftarrow_R \mathcal{U}_{pp}$, and then apply the triangular inequality.

$$\leq \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ \begin{pmatrix} m_0, m_1, \\ aux \end{pmatrix} \leftarrow_R \mathcal{A}_1(pp), \\ b \leftarrow_R \{0,1\}, \\ r \leftarrow_R \mathcal{R}_{pp},\ c := m_b + r, \\ b_{\mathcal{A}} \leftarrow_R \mathcal{A}_2(c, aux, pp) \end{array} \right] - \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ \begin{pmatrix} m_0, m_1, \\ aux \end{pmatrix} \leftarrow_R \mathcal{A}_1(pp), \\ b \leftarrow_R \{0,1\}, \\ r \leftarrow_R \mathcal{U}_{pp},\ c := m_b + r, \\ b_{\mathcal{A}} \leftarrow_R \mathcal{A}_2(c, aux, pp) \end{array} \right] \right|$$

$$+ \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ \begin{pmatrix} m_0, m_1, \\ aux \end{pmatrix} \leftarrow_R \mathcal{A}_1(pp), \\ b \leftarrow_R \{0,1\}, \\ r \leftarrow_R \mathcal{U}_{pp},\ c := m_b + r, \\ b_{\mathcal{A}} \leftarrow_R \mathcal{A}_2(c, aux, pp) \end{array} \right] - 1/2 \right|$$

Then we introduce a new adversary $\mathcal{B}$ against the decisional-RLWE game. This adversary, on input r, gets m_0, m_1, aux from $\mathcal{A}_1$, adds $m_b + r$, for a uniformly sampled $b \leftarrow_R \{0,1\}$, computes $b_{\mathcal{A}} \leftarrow_R \mathcal{A}_2(m_b + r, aux)$ and returns $b_{\mathcal{B}} = 1$ if $b = b_{\mathcal{A}}$ or $b_{\mathcal{B}} = 0$ otherwise. The first term is precisely the advantage of this adversary against the decisional-RLWE problem comparing both scenarios.

We can also ensure the second item is 0 because aux and c are independent of b given that r is uniformly random, so the probability of $b_{\mathcal{A}} = b$ is exactly $1/2$.

$$= \left| \Pr\left[b_{\mathcal{B}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ r \leftarrow_R \mathcal{R}_{pp}, \\ b_{\mathcal{B}} \leftarrow_R \mathcal{B}(r, pp) \end{array} \right] - \Pr\left[b_{\mathcal{B}} = 1 \;\middle|\; \begin{array}{c} pp \leftarrow_R \mathsf{Gen}(1^\lambda), \\ r \leftarrow_R \mathcal{U}_{pp}, \\ b_{\mathcal{B}} \leftarrow_R \mathcal{B}(r, pp) \end{array} \right] \right| + 0$$

$$= Adv'_{\mathrm{RLWE}}(\mathcal{B})$$

□

It should be noted that $\mathcal{A}$ and $\mathcal{B}$ are adversaries playing different games, and their running times are not exactly the same. That would be irrelevant for an asymptotic proof, as the reduction is polynomial, but if we wanted to bound the advantage of any adversary that is guaranteed to end in a specific time t then we would get that for any $\mathcal{A}$ with running time bounded by this t there is an adversary $\mathcal{B}$ with running time t' for which

$$\max_{\|\mathcal{A}\|\leq t} \left\{Adv_{\mathsf{Alg}}(\mathcal{A})\right\} \leq 2 \max_{\|\mathcal{B}\|\leq t'} \left\{Adv_{\mathrm{RLWE}}(\mathcal{B})\right\},$$

where t' is slightly greater than t because it includes a sum and a comparison (and we are omitting the details about how the message is encoded, which could again add a few more operations). Observe too that a factor 2 arises when using the same kind of definition for the advantage *Adv*, as we were previously bounding *Adv* against the Adv' for the RLWE problem.

We can not go far beyond this, because $Adv'_{\mathrm{RLWE}}(\mathcal{B})$ is something we do not know how to properly quantify. The hardness hypothesis just gives us an asymptotic bound, not specific probabilities. In particular every polynomial adversary has a running time bounded by a polynomial applied to the size of its input, but these polynomials can be different. For each adversary and every constant c there exists a λ_0 such that for every $\lambda > \lambda_0$ the advantage is smaller than λ^{-c}, but such λ_0 depends on the adversary. We simply assume that, if some analysis (as the lattice estimator from Albrecht *et al.* [8]) tells us that solving the problem takes 2^λ operations then that would imply that any real adversary bounded to polynomially many t' operations would have a tiny advantage. We do not estimate any particular advantage for adversaries with running time $t' < 2^\lambda$, as there is always a trade of between success probability and running time, but it is hard to quantify. An adversary with twice the running time might not exactly obtain a doubled success probability, so it makes no sense to finely characterize the relation between t' and the original t. For the same reasons as above we might just ignore the factor 2.

Statistical Indistinguishability Game

The other main reason that prevents an adversary from distinguishing (pp, c_0) and (pp, c_1) is far more simple and does not involve any computational problem. It is hard to differentiate two distributions if they are too similar. In this case, as there is no assumption involved, it does not matter if the adversary has an unbounded computational power.

The only option a more powerful adversary would have to amplify its success probability would be having access to multiple samples. Sometimes we assume the number of samples is polynomial and some other times a finite fixed number of samples is determined by design of the scheme because the distributions depend on a secret freshly obtained for each use.

The main notion we use to quantify this similarity among distributions is the statistical distance. There are other alternatives, as the Rényi divergence, that can be used for the same purpose and have been useful to determine more efficient parameters for some lattice-based constructions, but this kind of considerations are out of the scope of this dissertation [72, 16].

Definition 1.50 (Statistical Distance)**.** The *Statistical Distance* (SD) between two discrete probability distributions $\mathcal{X}$ and $\mathcal{Y}$ defined over a countable support $\mathcal{N}$, denoted as $\Delta(\mathcal{X},\mathcal{Y})$, is

$$\Delta(\mathcal{X},\mathcal{Y}) := \frac{1}{2}\sum_{n\in\mathcal{N}} \left|\mathcal{X}(n) - \mathcal{Y}(n)\right|.$$

We have denoted by $\mathcal{X}(n)$ the probability of sampling n when following the distribution $\mathcal{X}$. Equivalently for any event $E \subseteq \mathcal{N}$ we denote by $\mathcal{X}(E)$ the probability of sampling any element from E.

This distance is a useful tool because it allows us to prove that the probability of an event is similar if two distributions have a small distance.

Proposition 1.51 (Probability preserving property of the SD)**.**
Given an arbitrary event $E \subseteq \mathcal{N}$ we have that

$$\mathcal{Y}(E) \geq \mathcal{X}(E) - \Delta(\mathcal{X},\mathcal{Y}).$$

Proof. It can be directly deduced from the definition of the SD.

$$\begin{aligned}
\Delta(\mathcal{X},\mathcal{Y}) &= \frac{1}{2}\sum_{n\in\mathcal{N}} \left|\mathcal{X}(n) - \mathcal{Y}(n)\right| \\
&= \frac{1}{2}\sum_{n\in E} \left|\mathcal{X}(n) - \mathcal{Y}(n)\right| + \frac{1}{2}\sum_{n\in\mathcal{N}\setminus E} \left|\mathcal{X}(n) - \mathcal{Y}(n)\right| \\
&\geq \frac{1}{2}\sum_{n\in E} \left(\mathcal{X}(n) - \mathcal{Y}(n)\right) + \frac{1}{2}\sum_{n\in\mathcal{N}\setminus E} \left(\mathcal{Y}(n) - \mathcal{X}(n)\right) \\
&= \frac{1}{2}\sum_{n\in E} \left(\mathcal{X}(n) - \mathcal{Y}(n)\right) + \frac{1}{2}\left(1 - \sum_{n\in E}\mathcal{Y}(n) - 1 + \sum_{n\in E}\mathcal{X}(n)\right) \\
&= \mathcal{X}(E) - \mathcal{Y}(E)
\end{aligned}$$

□

This way we know that if $\mathcal{Y}(E)$ is bounded by ε we can also bound $\mathcal{X}(E) \leq \varepsilon + \Delta(\mathcal{X}, \mathcal{Y})$.

Corollary 1.52 (Probability difference). *The probability difference of the same event with two different probability distributions is bounded by its statistical distance.*

$$\left|\mathcal{X}(E) - \mathcal{Y}(E)\right| \leq \Delta(\mathcal{X}, \mathcal{Y})$$

Proof. Immediately follows from the previous Proposition 1.51, because the argument is symmetric, and we can proof both $\Delta(\mathcal{X}, \mathcal{Y}) \geq \mathcal{Y}(E) - \mathcal{X}(E)$ and $\Delta(\mathcal{X}, \mathcal{Y}) \geq \mathcal{X}(E) - \mathcal{Y}(E)$. □

Another important and well-known property is that the statistical distance between two distributions cannot ever increase if we apply the same (probabilistic) function to both distributions. This implies that the distributions of the output of any algorithm, when switching a sample from one distribution by a sample from a different distribution, are at a distance not larger than the SD of the two distributions.

We can then compare the advantage of an adversary that has to distinguish between two scenarios, Cha_0 and Cha_1, with the advantage of the same adversary distinguishing the two scenarios when we alter the first, Cha_0' and Cha_1. Let us denote the first advantage as $Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A})$ and consequently call the second advantage $Adv'_{\mathsf{IND}(\mathsf{Cha}_0',\mathsf{Cha}_1)}(\mathcal{A})$.

Substituting $c_0 \leftarrow_{\mathrm{R}} \mathsf{Cha}_0$ by $c_0 \leftarrow_{\mathrm{R}} \mathsf{Cha}_0'$, where C_0 and C_0' are the probability distributions induced by the challengers Cha_0 and Cha_0', we then have, heavily simplifying notation keeping only the relevant differences, that

$$\left|\, \Pr\left[b_{\mathcal{A}} = 1 \mid \mathsf{Cha}_0\right] - \Pr\left[b_{\mathcal{A}} = 1 \,\middle|\, \mathsf{Cha}_0'\right] \right| \leq \Delta(C_0, C_0').$$

Proposition 1.53 (Advantages and SD I). *The difference of advantages, comparing both scenarios, is bounded by the statistical distance if we change one of the challenges.*

$$Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) \leq Adv'_{\mathsf{IND}(\mathsf{Cha}_0',\mathsf{Cha}_1)}(\mathcal{A}) + \Delta(C_0, C_0').$$

Proof.

$$
\begin{aligned}
&Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) = \\
&= \Big| \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_1\big] \Big| \\
&= \Big| \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}'_0\big] + \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}'_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_1\big] \Big| \\
&\le \Big| \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}'_0\big] \Big| + \Big| \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}'_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_1\big] \Big| \\
&\le \Delta(C_0, C'_0) + \Big| \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}'_0\big] - \Pr\big[b_{\mathcal{A}} = 1 \big| \mathsf{Cha}_1\big] \Big| \\
&= \Delta(C_0, C'_0) + Adv'_{\mathsf{IND}(\mathsf{Cha}'_0,\mathsf{Cha}_1)}(\mathcal{A})
\end{aligned}
$$

□

This argument cannot only be used when one of the scenarios changes, but also when some distribution is used in both of them, and we want to compare the advantage with the one the adversary would have if this distribution was altered in both cases.

Proposition 1.54 (Advantages and SD II)**.** *The difference of advantages, compared with a random guess, is bounded by the statistical distance if we substitute one distribution D by an alternative distribution D' in both challenges.*

$$Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) \le Adv_{\mathsf{IND}(\mathsf{Cha}'_0,\mathsf{Cha}'_1)}(\mathcal{A}) + \Delta(D, D').$$

Proof.

$$
\begin{aligned}
&Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) = \\
&= \Big| \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}_b\right] - 1/2 \Big| \\
&= \Big| \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}_b\right] - \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}'_b\right] + \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}'_b\right] - 1/2 \Big| \\
&\le \Big| \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}_b\right] - \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}'_b\right] \Big| + \Big| \Pr\left[b_{\mathcal{A}} = b \mid \mathsf{Cha}'_b\right] - 1/2 \Big| \\
&\le \Delta(D, D') + Adv_{\mathsf{IND}(\mathsf{Cha}'_0,\mathsf{Cha}'_1)}(\mathcal{A})
\end{aligned}
$$

□

Notice so far we are being scarce on details about the distributions C_0, C'_0, D or D', but all these distributions have to be defined with extreme care, paying attention to the details of the game we are defining to be able to correctly compute the statistical distances.

For example, perhaps surprisingly, we see that the difference of advantages is always bounded by the statistical distance when we change one distribution no matter if we are computing the advantage compared with a random guess Adv_{IND} or the advantage comparing the two scenarios Adv'_{IND}, even if the first is half the second. What really happens is that distributions are defined on different probability spaces, and there is no discrepancy if we take this fact into account.

From the second bound we can recover the first if we consider a case where the second challenge does not depend on D and therefore $\mathsf{Cha}'_1 = \mathsf{Cha}_1$, because in that case $\Delta(D, D') = {}^1/_2\Delta(C_0, C'_0)$ (the ${}^1/_2$ term appears because the scenario 0 only happens with probability ${}^1/_2$, and we have to consider probabilities with respect to the whole game). This matches the fact that Adv is half the Adv'. We use one interpretation or the other for convenience, but always keep in mind the corresponding probability space.

In general, we would be able to prove the security properties via a chain of these arguments, possibly combining some computational ones with other with a statistical nature. The number of statistical ones, for which we can precisely measure the difference, are going to determine how close each pair of distributions have to be to finally obtain the desired margin.

Particularly interesting is that we can directly apply Proposition 1.53 if the statistical distance between C_0 and C_1 is smaller than ε. The advantage of any adversary would then be smaller than ε, considering that we can bound $Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}$ by $\Delta(C_0, C_1) + Adv'_{\mathsf{IND}(\mathsf{Cha}_1,\mathsf{Cha}_1)}$, and the second term is just zero.

1.4.3 Dismissing Events with Negligible Probability

A common source of difficulties is that the different properties a scheme has to satisfy might require different conditions that are challenging to piece together.

A common example in lattice-based schemes is the fact that we understand the difficulty of the RLWE problem when the errors are sampled from discrete Gaussian distributions, but many problems and scheme properties (such as correctness of an encryption scheme or soundness in a ZKPoK) require the errors to be small. An integer $e \in \mathbb{Z}$ sampled from a discrete Gaussian D_σ is likely to be small (how small depends on the σ parameter), but there is always a non-zero chance of getting a result arbitrarily large.

There are two things we can do about this issue. First, we have to either accept that the scheme would fail with small probability, or prove that it would remain secure if we use instead a bounded discrete Gaussian $D_{\sigma,B}$. Second we have to ensure that the probability of the tails is sufficiently small. We are going to focus here on proving that bounded discrete Gaussians would work almost as well using the propositions from the previous subsection, as long as the probability of the tails is sufficiently small. Then, in the following subsection, we are going to deal with that probability.

Alternatively, in order to be able to guarantee the zero-knowledge property of the Fiat-Shamir with aborts strategy we need for the rejection sampling Theorem 1.37 that the secret multiplied by the challenge belong to a bounded set. However, that can only be guaranteed for discrete Gaussians except with some small probability, so we would need the opposite argument, being able to use a distribution that is only bounded except with some small probability when the property would require a bounded distribution.

We are interested in studying what occurs with the adversary when the event that happens with a small probability is some error being bounded. In general, we can describe any condition we are interested in as a generic event that we can denote as E. Let D_E be the distribution conditioned to such event E.

$$D_E(x) := \begin{cases} \dfrac{1}{1 - D(\overline{E})} D(x) & \text{if } x \in E, \\ 0 & \text{if } x \notin E. \end{cases}$$

Proposition 1.55 (SD of the conditioned distribution)**.** *The statistical distance between an original probability distribution and the conditioned distribution is precisely the probability of not the event we are conditioning on.*

$$\Delta(D, D_E) = D(\overline{E})$$

Proof.

$$\begin{aligned} \Delta(D, D_E) &= \frac{1}{2} \sum_{x \in \mathcal{N}} \left| D(x) - D_E(x) \right| \\ &= \frac{1}{2} \sum_{x \in E} \left| D(x) - D_E(x) \right| + \frac{1}{2} \sum_{x \notin E} \left| D(x) - D_E(x) \right| \\ &= \frac{1}{2} \sum_{x \in E} \left| \frac{D(\overline{E})}{1 - D(\overline{E})} D(x) \right| + \frac{1}{2} \sum_{x \notin E} \left| D(x) \right| \\ &= {}^1/_2 D(\overline{E}) + {}^1/_2 D(\overline{E}) \\ &= D(\overline{E}) \end{aligned}$$

□

From this proposition we can deduce that the difference between the advantages when we condition to some event is related to the probability of that event not happening. Depending on whether we want that event to be considered in one or the two scenarios we can more naturally write it using each of the alternative definitions of the advantages.

Proposition 1.56. *The difference of advantages when we condition the distributions by an event E is bounded by the probability of this event not happening.*

$$\left|Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) - Adv_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1|E)}(\mathcal{A})\right| \leq \Pr\left[\overline{E}\right].$$
$$\left|Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) - Adv'_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1)}(\mathcal{A})\right| \leq \Pr\left[\overline{E}\right].$$

Proof. This proposition can be deduced from the relation between the advantage and the statistical distances (Propositions 1.53 and 1.54) and the fact that the SD is the probability of the event not happening (Proposition 1.55). □

As an immediate application we can use this to relate the advantages of conditioned scenarios to the advantages of the original ones.

Corollary 1.57 (Advantages and negligible events I)**.**

$$Adv_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1|E)}(\mathcal{A}) \leq Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) + \Pr\left[\overline{E}\right].$$

Corollary 1.58 (Advantages and negligible events II)**.**

$$Adv'_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1)}(\mathcal{A}) \leq Adv'_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) + \Pr\left[\overline{E}\right].$$

Observe that, even if we always use the same notation for the event E, the probability space where it is defined is important to measure its probability, and it is not the same if we consider one game or another, as we mentioned before.

The inequalities would also work in the other direction, but directly studying the probabilities we can obtain a more tight result.

Proposition 1.59 (Advantages and negligible events III)**.** *If we want to bound the advantage of the original game by the advantage of the conditioned game then we can obtain a tighter bound given by*

$$Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) \leq Adv_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1)}(\mathcal{A}) + \frac{1}{2}\Pr\left[\overline{E}\right],$$

$$Adv_{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}(\mathcal{A}) \leq Adv_{\mathsf{IND}(\mathsf{Cha}_0|E,\mathsf{Cha}_1|E)}(\mathcal{A}) + \frac{1}{2}\Pr\left[\overline{E}\right].$$

Proof. Independently of whether the event affects the distribution of one or both of the challenges we can prove the proposition, taking into account that the $\Pr\left[\overline{E}\right]$

depends on the probability space.

$$\begin{aligned}
&\left|\Pr\left[b_{\mathcal{A}} = b\right] - 1/2\right| = \\
&\quad = \left|\Pr\left[b_{\mathcal{A}} = b \,\middle|\, E\right] \cdot \Pr\left[E\right] + \Pr\left[b_{\mathcal{A}} = b \,\middle|\, \overline{E}\right] \cdot \Pr\left[\overline{E}\right] - 1/2\Big(\Pr\left[E\right] + \Pr\left[\overline{E}\right]\Big)\right| \\
&\quad = \left|\Big(\Pr\left[b_{\mathcal{A}} = b \,\middle|\, E\right] - 1/2\Big) \cdot \Pr\left[E\right] + \Big(\Pr\left[b_{\mathcal{A}} = b \,\middle|\, \overline{E}\right] - 1/2\Big) \cdot \Pr\left[\overline{E}\right]\right| \\
&\quad \leq \left|\Pr\left[b_{\mathcal{A}} = b \,\middle|\, E\right] - 1/2\right| \cdot \Pr\left[E\right] + \left|\Pr\left[b_{\mathcal{A}} = b \,\middle|\, \overline{E}\right] - 1/2\right| \cdot \Pr\left[\overline{E}\right] \\
&\quad \leq \left|\Pr\left[b_{\mathcal{A}} = b \,\middle|\, E\right] - 1/2\right| + 1/2 \Pr\left[\overline{E}\right]
\end{aligned}$$

□

Observe that there are simple instances that achieve the equality when considering the other direction (bounding the conditioned advantage by the original one plus the probability of the event not being true). Consider two challengers Cha_0 and Cha_1 that independently sample a uniformly random bit $c_0, c_1 \leftarrow_{\mathrm{R}} \{0,1\}$. For the sake of simplicity let $\mathcal{A}$ be the adversary that takes c_b as input and directly outputs it as its guess. It is clear that the advantage of this particular adversary is identically zero, because its outputs follow the same distribution in both scenarios.

$$\underset{\mathsf{IND}(\mathsf{Cha}_0,\mathsf{Cha}_1)}{Adv'}(\mathcal{A}) = \left|\Pr\left[b_{\mathcal{A}} = 1 \,\middle|\, \begin{array}{c} c \leftarrow_{\mathrm{R}} \mathsf{Cha}_0, \\ b_{\mathcal{A}} := c \leftarrow \mathcal{A}(c) \end{array}\right] - \Pr\left[b_{\mathcal{A}} = 1 \,\middle|\, \begin{array}{c} c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1, \\ b_{\mathcal{A}} := c \leftarrow \mathcal{A}(c) \end{array}\right]\right| = 0.$$

However, if we consider a different challenger Cha'_0 that this time always outputs 0 the advantage of the same adversary would then be $1/2$, because now it only outputs 1 when the challenger is Cha_1 and $|0 - 1/2| = 1/2$.

$$\underset{\mathsf{IND}(\mathsf{Cha}'_0,\mathsf{Cha}_1)}{Adv'}(\mathcal{A}) = \left|\Pr\left[b_{\mathcal{A}} = 1 \,\middle|\, \begin{array}{c} c := 0 \leftarrow \mathsf{Cha}'_0, \\ b_{\mathcal{A}} := c \leftarrow \mathcal{A}(c) \end{array}\right] - \Pr\left[b_{\mathcal{A}} = 1 \,\middle|\, \begin{array}{c} c \leftarrow_{\mathrm{R}} \mathsf{Cha}_1, \\ b_{\mathcal{A}} := c \leftarrow \mathcal{A}(c) \end{array}\right]\right| = 1/2.$$

Notice we can see Cha'_0 as a challenger that uses the uniform distribution C_0 from Cha_0 but conditioned to the event E given by $c_0 = 0$. The difference of the advantages is precisely $\Pr\left[\overline{E}\right] = 1/2$.

It is also interesting to analyze what happens when we define this same game considering the advantage definition compared to a random guess, which is again identically 0 because $b_{\mathcal{A}}$ is independent of b.

$$\underset{\mathsf{IND(Cha_0,Cha_1)}}{Adv}(\mathcal{A}) = \left|\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} c_0 \leftarrow_{\mathrm{R}} \mathsf{Cha}_0, \\ c_1 \leftarrow_{\mathrm{R}} \mathsf{Cha}_1, \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ b_{\mathcal{A}} := c_b \leftarrow \mathcal{A}(c_b) \end{array}\right] - 1/2\right| = \left|1/2 - 1/2\right| = 0.$$

Now the advantage distinguishing Cha'_0 and Cha_1 would be $1/4$, because the adversary would always guess Cha'_0 correctly and would also guess Cha_1 half the time, for a total success probability of $3/4$ which is $1/4$ above a random guess.

$$\underset{\mathsf{IND(Cha'_0,Cha_1)}}{Adv}(\mathcal{A}) = \left|\Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} c_0 := 0 \leftarrow \mathsf{Cha}'_0, \\ c_1 \leftarrow_{\mathrm{R}} \mathsf{Cha}_1, \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ b_{\mathcal{A}} := c_b \leftarrow \mathcal{A}(c_b) \end{array}\right] - 1/2\right| = \left|3/4 - 1/2\right| = 1/4.$$

We can consider again that this scenario is equivalent to the previous one but conditioning C_0 to be 0, so this time the advantage would be $1/2 \Pr\left[\overline{E}\right]$, which is consistent with $Adv(\mathcal{A}) = 1/2\, Adv'(\mathcal{A})$ as seen in Lemma 1.47. However, provided that the output of Cha'_0 does not have any effect if $b = 1$ we can also see this game as an instance of the previous one conditioned by the event E' given by $c_0 = 0$ or $b = 1$. The probability of E' is $3/4$, so the advantage now would be equal to $\Pr\left[\overline{E'}\right]$, satisfying the equality.

These subtleties mean that we have to be very careful defining the events and the probability spaces, so we can always use the tightest bound to obtain the best reductions possible.

1.4.4 Multidimensional Gaussians

As we have been mentioning the hardness of lattice problems is well understood for discrete Gaussians D_σ, but the schemes need bounded distributions, as truncated Gaussians $D_{\sigma,B}$. An important question is then how to bound the probability of the tails, considering that we are usually sampling errors $\boldsymbol{e} \in R_q^m$ from truncated multidimensional Gaussians $\boldsymbol{e} \leftarrow_{\mathrm{R}} (D^n_{\sigma,B})^m$. We then consider the distribution $\boldsymbol{e} \leftarrow_{\mathrm{R}} D^d_{\sigma,B}$ with $d := n \cdot m$.

Let $\Pr_{e \leftarrow_{\mathrm{R}} D_\sigma}\left[\,|e| > B\right] \leq 2^{-a}$, we can then discuss how likely is that a tuple of independent samples is also bounded.

We have that

$$\Pr_{e \leftarrow_R D_\sigma} \left[|e| > B \right] \leq 2^{-a},$$
$$\Pr_{e \leftarrow_R D_\sigma} \left[|e| \leq B \right] \geq 1 - 2^{-a}.$$

Then,

$$\Pr_{e_i \leftarrow_R D_\sigma} \left[|e_i| \leq B \;\middle|\; \forall i \in \{1, \ldots, d\} \right] \geq (1 - 2^{-a})^d,$$
$$\Pr_{\boldsymbol{e} \leftarrow_R D_\sigma^d} \left[\|\boldsymbol{e}\|_\infty > B \right] \leq 1 - (1 - 2^{-a})^d.$$

We would like to find the b such that $1 - (1 - 2^{-a})^d = 2^{-b}$.

$$\begin{aligned} -b &= \log(1 - (1 - 2^{-a})^d) \\ b &= -\log(1 - (1 - 2^{-a})^d) \\ b &= -\log\left(1 - \sum_{i=0}^{d} (-1)^i \binom{d}{i} 2^{-ai}\right) \\ b &= -\log\left(- \sum_{i=1}^{d} (-1)^i \binom{d}{i} 2^{-ai}\right) \end{aligned}$$

We expand the binomial power because it allows us to remove one of the steps where rounding errors could be produced, subtracting something close to 1 from 1.

Observe inside the logarithm we are adding negative powers of 2 that decrease really fast. If we keep only the first term we get a very reasonable approximation.

$$b \approx -\log(d \cdot 2^{-a}) = a - \log(d)$$

Depending on what we want to obtain it could be sufficient to get an inequality. Observe that the approximation we have obtained is indeed a lower bound for b, and therefore $2^{-(a-\log(d))} \geq 2^{-b}$, as it corresponds to bounding the probability of some of the components being greater than B by the addition of the probabilities for each of them, computing $\sum_{i=1}^{d} 2^{-a} = 2^{-(a-\log_2(d))} \geq 2^{-b}$.

We can also obtain this bound profiting from the fact that the statistical distance satisfies the triangular inequality. We could then build a chain of inequalities using consecutive distributions that start from D_σ^d and end up with $D_{\sigma,B}^d$ by truncating each time the tails from one dimension. $\Delta(D_\sigma^d, D_{\sigma,B}^d) \leq d\Delta(D_\sigma, D_{\sigma,B}) \leq d2^{-a} = 2^{-(a-\log(d))}$.

Proposition 1.60 (Multidimensional Gaussian tails)**.**
If $\Pr_{e \leftarrow D_\sigma} \left[|e| > B \right] \leq 2^{-a}$ *then* $\Delta(D_\sigma^d, D_{\sigma,B}^d) \leq 2^{-(a-\log(d))}$.

Observation 1.61. We have already proved this proposition from a probabilistic point of view and applying properties of the statistical distance, but it is worth mentioning that this result is equivalent to Bernoulli's inequality (we will later introduce it in Proposition 4.10).

We finally reproduce here the influential lemma from Lyubashevsky's [77].

Lemma 1.62 (Lemma 4.4 from [77]).
For any $k > 0$ we have that $\Pr\left[\,|z| > k\sigma \;\middle|\; z \leftarrow_{\mathrm{R}} D_\sigma\right] \leq 2\exp(-k^2/2)$.

Auxiliary Functions

To simplify coming proofs we make use of the recent propositions to define several auxiliary functions practical for defining parameters and checking security properties.

We think this is a useful summary. In each of these propositions the relevant points are the direction of the inequalities and whether the results that we obtain are the smaller or the larger for which we can ensure the considered property.

Proposition 1.63 (boundedPr). *The function boundedPr, defined as follows, outputs the greater a for which we can guarantee that the probability of a sample from D_σ not belonging to $[-B, B]$ is lower or equal than 2^{-a}.*

$$\mathsf{boundedPr}(\sigma, B) := \max\left\{0, \frac{B^2}{2\sigma^2}\log(e) - 1\right\}.$$

Proof. Directly comes from Lemma 1.62 (Lemma 4.4 in [77]). □

Proposition 1.64 (vecBoundedPrFromBoundedPr). *The function vecBoundedPrFromBoundedPr, defined as follows, outputs the greater b for which we can ensure that the sub-infinity norm of a vector of dimension d not being bounded by a certain bound has a probability smaller or equal to 2^{-b} if a single sample has a probability smaller or equal to 2^{-a} of not being within that bound.*

$$\mathsf{vecBoundedPrFromBoundedPr}(a, d) := \max\left\{a - \log(d), 0\right\}.$$

Proof. Direct, iteratively applying the triangular inequality to bound the statistical distance (using Proposition 1.60), as we have discussed before. We have only been able to obtain inequalities, so the actual probability would be smaller, but this is what we can guarantee. □

Proposition 1.65 (vecBoundedPrToBoundedPr). *The function vecBoundedPrToBoundedPr, defined as follows, outputs the smaller a for which we can ensure that all independently sampled elements of a vector of dimension d satisfy some condition except with probability* *smaller or equal than 2^{-b} if a single sample satisfies it except with probability smaller or equal than 2^{-a}.*

$$\mathsf{vecBoundedPrToBoundedPr}(b, d) := b + \log(d).$$

Proof. Can be deduced too from Proposition 1.60. □

Proposition 1.66 (sigmaFromB). *The function* sigmaFromB *outputs the greater parameter σ for which we can ensure that a sample from D_σ only surpasses the bound B with a probability smaller or equal than 2^{-a}.*

$$\mathsf{sigmaFromB}(a, B) := B\sqrt{\frac{\log(e)}{2(a+1)}}.$$

Proof. Can be deduced from Lemma 1.62 (Lemma 4.4 in [77]) again. □

Observation 1.67. With these auxiliary functions we have been considering a symmetric scenario discussing whether samples belong or not to the interval $[-B, B]$. However, to ensure the small elements can be characterized solely from the length of its bit decomposition we require the interval to be $[-2^\kappa, 2^\kappa)$ for some $\kappa \in \mathbb{Z}_{\geq 0}$.

For that reason we just use $B = 2^\kappa - 1$, as $[-2^\kappa + 1, 2^\kappa - 1] \subset [-2^\kappa, 2^\kappa)$. This means the bounds that we use are not the tightest ones we could obtain, but the tests we have conducted with real instantiations of the schemes that are going to be presented in the following chapters show no perceptible difference on the final parameters from using this much more simple symmetrical inequalities.

1.5 Contributions

The first contribution, presented in Chapter 2, is a complete analysis of the necessary and sufficient conditions that we need to be able to efficiently multiply polynomials in the particular quotient rings of interest for lattice-based cryptography, i.e. $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ with q an odd prime number and n a power of two. Fast Fourier multiplication is a well-known technique for efficient multiplication of polynomials, but cannot always be used.

The *Fast Fourier Transform* (FFT) approach can be generalized to more general situations, with the so-called partial FFT multiplication algorithm, and we particularly study when partial FFT can be applied to general rings $R_m = \mathbb{Z}[x]/\langle x^n - a\rangle$ with m and a arbitrary positive integers. This techniques are extensively used in lattice-based cryptographic implementations, but many of the conditions are vaguely defined in the literature, or the algorithm is described only as a set of rules for a particular case without any further explanation, so we believe a rigorous theoretical analysis as the one presented in Chapter 2 would be of interest for the lattice-based cryptographic community. The results presented in this chapter are under review for publication at the time of deposit of this dissertation, and a preprint version is available at [88].

The main contribution of this dissertation is split between Chapters 3 and 4. We present new and more efficient Stern-based ZKPoKs, and apply them defining a new lattice-based commitment scheme providing protocols to prove knowledge of valid openings of given commitments and to prove that the openings of three commitments satisfy linear or multiplicative relations.

The commitment scheme and the interactive proofs are presented in Chapter 3, proving that the commitment and the proofs are secure provided a set of conditions is satisfied. A previous version of the commitment and the interactive proofs were presented at the 17th IMA International Conference on Cryptography and Coding, celebrated in Oxford in 2019 [87].

Finally, in Chapter 4 we see how to transform the interactive proof into NIZKPoKs via the Fiat-Shamir transform, and study its security. We implement both the commitment and the NIZKPoKs, describe how to obtain optimal sets of secure parameters and benchmark the performance of the prototype implementation. The implementation has been a joint work with Sergi Rovira, and the details about it, together with the NIZKPoKs and the security analysis of the instantiations are also currently under review for publication. A preprint version is available at [89].

Chapter 2

Partial FFT Multiplication

One of the main advantages of lattice-based cryptography is that, computationally speaking, every computation is fairly simple (perhaps excluding Gaussian sampling). Regular lattices only require linear operations as matrix-vector products, with polynomially large elements, and ideal-lattices only add polynomial products, allowing even more efficient implementations. This makes some lattice constructions good candidates for lightweight cryptography, designed to be executed in resource-constrained hardware, for example with Internet of Things devices [71].

For this reason, having efficient multiplication algorithms for the quotient ring of polynomials $\mathbb{Z}_m[x]/\langle x^n - a\rangle$, with n a power of 2 and m a non necessarily prime integer, is of great importance. The most efficient algorithm known involves the FFT when m is a prime (and then usually denoted as q) and a is -1, such that $x^n + 1$ fully splits in linear factors modulo q. Through this chapter we systematize the existing knowledge and meticulously study the necessary and/or sufficient conditions required for the applicability of these multiplication algorithms. This work allows us to unify the different approaches to the problem of efficiently computing the product of two polynomials in these quotient rings and formally generalize existing algorithms to partially splitting rings where $x^n + 1$ does not fully split in linear factors.

2.1 Introduction

Constructing efficient multiplication algorithms for polynomials with coefficients in a ring R, through this chapter we denote by R the arbitrary ring of the coefficients (and not the ring of polynomials itself), has been an extensive research area. Given two polynomials $g(x), h(x) \in R[x]$ of degree bounded by n, $g(x) = \sum_{i=0}^{n-1} g_i x^i$ and $h(x) = \sum_{i=0}^{n-1} h_i x^i$, computing its product in a naïve way (known as the schoolbook multiplication algorithm),

$$(g \cdot h)(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \left(g_i \cdot h_j\right) x^{i+j},$$

requires a quadratic number, n^2, of multiplications of elements from the ring R.

$$\begin{array}{rrrrrrr}
 & & & g_3x^3 + & g_2x^2 + & g_1x + & g_0 \\
 & & \times & h_3x^3 + & h_2x^2 + & h_1x + & h_0 \\
\hline
 & & & g_3 \cdot h_0x^3 + & g_2 \cdot h_0x^2 + & g_1 \cdot h_0x + & g_0 \cdot h_0 \\
 & & g_3 \cdot h_1x^4 + & g_2 \cdot h_1x^3 + & g_1 \cdot h_1x^2 + & g_0 \cdot h_1x & \\
 & g_3 \cdot h_2x^5 + & g_2 \cdot h_2x^4 + & g_1 \cdot h_2x^3 + & g_0 \cdot h_2x^2 & & \\
g_3 \cdot h_3x^6 + & g_2 \cdot h_3x^5 + & g_1 \cdot h_3x^4 + & g_0 \cdot h_3x^3 & & & \\
\hline
\sum\limits_{i+j=6} g_ih_jx^6 + & \sum\limits_{i+j=5} g_ih_jx^5 + & \sum\limits_{i+j=4} g_ih_jx^4 + & \sum\limits_{i+j=3} g_ih_jx^3 + & \sum\limits_{i+j=2} g_ih_jx^2 + & \sum\limits_{i+j=1} g_ih_jx + & g_0h_0
\end{array}$$

If we now want to work in a ring $R[x]/\langle f(x)\rangle$ the schoolbook multiplication algorithm would perform the same amount of operations taking into account that at the end we might perform a reduction.

For the particular cases $f(x) = x^n \pm 1$ it is clear that the algorithm performs the same number of operations, as we just need to multiply by ∓ 1 all ith coefficients from n to $2n - 2$ and add them to their corresponding $i - n$ column.

The goal of this chapter is to formalize and unify some concepts used to build more efficient multiplication algorithms focusing on the ring $R[x] = \mathbb{Z}_m[x]$ and then on its quotient $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$, with n a power of 2 and m non necessarily prime (although sometimes we would also consider $x^n - 1$ or in general $x^n - a$).

We choose to study optimizations for this particular ring as it is widely used by cryptographic constructions that base their security on ideal lattices [83], that can be identified with ideals in $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$ (as we have seen in Section 1.3, restricting ourselves to the particular case of a prime modulus, but lattice problems can also be defined with non-prime modulus).

Most of the literature usually deals with this matter by providing a set of recipes that can be applied for some specific particular rings (considering whether m is prime or not or how does $x^n - a$ split modulo m), without specifying if the imposed conditions in these recipes are necessary or only sufficient. This lack of detail might make more difficult the applicability of such recipes. Through this chapter we instead analyze what are the fundamental properties that allow us to obtain a computational speedup from a comprehensive and mathematical point of view, so that the reader can apprehend these techniques and distinguish intrinsic properties from superfluous conventions.

Therefore, the analysis presented in this chapter will help the reader to avoid confusions when using multiplication algorithms for polynomials in quotient rings as $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$.

Many of the ideas developed in this chapter are folklore when working on other rings such as $\mathbb{C}[x]$ or $\mathbb{Z}_q[x]$ with q a prime satisfying certain conditions, but an exhaustive analysis might be very helpful to analyze when these ideas can be, completely or partially, generalized to our ring of interest and why the required conditions for the underlying ring are indeed necessary or just sufficient.

2.1.1 Polynomial Multiplication Related Work

Karatsuba Multiplication Algorithm

The first subquadratic multiplication algorithm was designed by Karatsuba in [67], with a cost of $\mathcal{O}(n^{\log 3})$ derived from a clever divide and conquer strategy. We include it here as our approach uses Karatsuba's algorithm as a subroutine.

Let $g(x)$ and $h(x)$ be two polynomials of degree strictly smaller than n (a power of two). We can split these polynomials into upper and lower degree polynomials as $g(x) = g_U(x)x^{n/2} + g_L(x)$ and $h(x) = h_U(x)x^{n/2} + h_L(x)$ where g_L, g_U, h_L, h_U have all degree smaller than $n/2$.

A naïve computation would be

$$\begin{aligned} g(x)\cdot h(x) &= (g_U(x)x^{n/2} + g_L(x))(h_U(x)x^{n/2} + h_L(x)) \\ &= (g_U(x)\cdot h_U(x))\, x^n \\ &\quad + (g_U(x)\cdot h_L(x) + g_L(x)\cdot h_U(x))\, x^{n/2} \\ &\quad + g_L(x)\cdot h_L(x). \end{aligned}$$

That way we divide the full multiplication into four multiplications of polynomials of size $n/2$. One can see however that this does not improve the efficiency of the computation. Let $T(n)$ be the number of operations required for computing the product using this method. The recurrence obtained, $T(n) = 4T(n/2) + \mathcal{O}(n)$, implies (via the Master Theorem for divide-and-conquer recurrences [20]) that $T(n) = \mathcal{O}(n^2)$.

Karatsuba's gifted idea was to notice that the crossed terms can be obtained from the other terms and a single multiplication of $n/2$-polynomials. That is, we can write the term $(g_U(x) \cdot h_L(x) + g_L(x) \cdot h_U(x))$ as

$$(g_L(x) + g_U(x))(h_L(x) + h_U(x)) - (g_U(x) \cdot h_U(x)) - (g_L(x) \cdot h_L(x)) .$$

Notice how the recurrence now computes only 3 products of half size and a linear amount of operations ($T(n) = 3T(n/2) + \mathcal{O}(n)$), providing the desired sublinear running time of $T(n) = \mathcal{O}(n^{\log 3})$ (solving again the recurrence with the Master Theorem for divide-and-conquer recurrences [20]).

Algorithm 2.1 KARATSUBA

```
   Input: Two polynomials $g(x)$ and $h(x)$ of degree bounded by $n$
   Output: Product of $g(x) \cdot h(x)$
1: if n = 1 then return $g \cdot h$
   Split $g(x)$ and $h(x)$ into $g_L(x), g_U(x), h_L(x), h_U(x)$.
2: $a(x) := \textsc{Karatsuba}(g_U(x), h_U(x))$
3: $b(x) := \textsc{Karatsuba}(g_L(x), h_L(x))$
4: $c(x) := \textsc{Karatsuba}(g_L(x) + g_U(x), h_L(x) + h_U(x))$
5: return $a(x)x^n + (c(x) - a(x) - b(x))x^{n/2} + b(x)$
```

Karatsuba Multiplication Algorithm mod $f(x)$

However, notice this is not the most natural way of writing this recursion when working modulo $x^n \pm 1$ as all recursive calls work the same way but the last one, where a reduction has to be performed.

Alternatively we can split $g(x) = g_1(x^2)x + g_0(x^2)$, with g_0 containing the even coefficients and g_1 containing the odd ones. This allows us to think of $g(x) \in R[x]$ of degree smaller than n as $g(x, y) = g_0(y) + g_1(y)x \in R[x, y]$ of x-degree 1 and y-degree smaller than $n/2$. It is called *Dual Karatsuba* and the idea remains the same:

$$\begin{aligned} g(x) \cdot h(x) &= (g_1(x^2)x + g_0(x^2))(h_1(x^2)x + h_0(x^2)) \\ &= \left(g_1(x^2) \cdot h_1(x^2)\right) x^2 \\ &\quad + \left(g_1(x^2) \cdot h_0(x^2) + g_0(x^2) \cdot h_1(x^2)\right) x \\ &\quad + g_0(x^2) \cdot h_0(x^2) \end{aligned}$$

And analogously as we did before we can write the second term with only one additional multiplication. $\left(g_1(x^2) \cdot h_0(x^2) + g_0(x^2) \cdot h_1(x^2)\right)$ is

$$\left(\left(g_1(x^2) + g_0(x^2)\right) \cdot \left(h_1(x^2) + h_0(x^2)\right) - g_1(x^2) \cdot h_1(x^2) - g_0(x^2) \cdot h_0(x^2)\right) .$$

Now the reduction modulo $x^n \pm 1$ works at each level of the recursive algorithm, as $\widehat{g}(y) \cdot \widehat{h}(y) \pmod{y^{n/2} \pm 1}$ is equivalent to $\widehat{g}(x^2) \cdot \widehat{h}(x^2) \pmod{x^n \pm 1}$ once we change variables again. We obtain no computational advantage, but it allows us to understand it from a different perspective.

Algorithm 2.2 KARATSUBA $\pmod{x^n \pm 1}$

```
   Input: Two polynomials $g(x)$ and $h(x)$ of degree bounded by $n$
   Output: Product of $g(x) \cdot h(x)$
1: if n = 1 then return $g \cdot h$
   Split $g(x)$ and $h(x)$ into $g_0(x), g_1(x), h_0(x), h_1(x)$.
2: $a(y) :=$ KARATSUBA($g_1(y), h_1(y)$)
3: $b(y) :=$ KARATSUBA($g_0(y), h_0(y)$)
4: $c(y) :=$ KARATSUBA($g_0(y) + g_1(y), h_0(y) + h_1(y)$)
5: return $a(x^2)x^2 + (c(x^2) - a(x^2) - b(x^2))x + b(x^2) \pmod{x^n \pm 1}$
```

Notice $g_0(y) + g_1(y) \equiv g(x, y) \mod x - 1$ and $g_0(y) \equiv g(x, y) \mod x$. These ideas are considered in [22] in order to see all these tools as part of the same framework.

Faster Multiplication Algorithms

Even faster algorithms can be obtained from more clever recurrences. For example mapping the polynomials into a different domain in a recursive way where they can be efficiently multiplied in linear time. If the recursion works computing two transforms of half the size, $T(n) = 2T(n/2) + O(n)$, then the final computational complexity is $O(n \log n)$.

Through this work we are going to focus and systematically explore the multiplication algorithms derived from the FFT paradigm, that is going to be extensively described in the following sections. This approach is usually referred as *Number Theoretic Transformation* (NTT) when working with finite fields.

The main idea of the FFT recurrence is attributed to Gauss and was fully developed by Cooley and Tukey in its seminal work [42], considering the ring of complex numbers.

Many variants have been developed since then, generalizing [42] to non-power of two bounded degree polynomials or providing additional tricks and interpretations from which we benefit, such as [53]. An extensive and magnificently well documented survey can be found in [22].

Most of the work has focused on the particular case of multiplications in

$\mathbb{Z}_q[x]/\langle x^n + 1\rangle$ when q is prime and $x^n + 1$ fully splits in linear factors. That has been studied for a while, both from a software [79] and hardware [99] point of view.

We are particularly interested in the ideas presented in [86], as they specifically discuss multiplications of polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$ when $x^n + 1$ does not fully split in linear factors (a situation that happens in some lattice-based cryptographic schemes such as some commitment schemes [17], and that is the case of the proposals we are going to present in the following chapters) and the standard FFT can only be partially applied. However, [86] only considers the case with a prime q and briefly describes the procedure.

This technique of partially applying an FFT is sometimes called incomplete NTT [40] and usually interpreted like a *Chinese Remainder Transform*, as in [85] doing Fast Chinese remaindering [121].

However, most of the literature only provides some sufficient conditions that allow some particular implementation or specific abstraction of a fast multiplication algorithm that are not directly generalizable.

In spite of that we present a more general framework for multiplications in $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$ that would allow the reader to comprehend why some folklore assumptions are indeed necessary and why some others are not, from a mathematically rigorous, yet accessible for readers not familiarized with algebraic constructions, point of view.

2.1.2 Conventions

Since our goal is to work in $R[x]/\langle f(x)\rangle$, with f a monic polynomial, we choose as a representative for $g(x) \in R[x]/\langle f(x)\rangle$ its remainder when divided by $f(x)$, denoted by $g(x) \ \text{rem}\, f(x)$.

We borrow most of our notation from [22], and present some new definitions through Sections 2.3 and 2.4, that we believe are of independent interest.

2.2 Pointwise Product

The main idea behind any FFT multiplication technique is to compute the product of two polynomials via the pointwise product of their evaluations on certain points of the ring R.

It is straightforward by the definition of the product of polynomials that, given a point $x_0 \in R$, the evaluation of the product is equal to the product of the evaluations $(g \cdot h)(x_0) = g(x_0) \cdot h(x_0)$.

This way, we intend to compute enough evaluations (we denote this transform as T), then we could perform a pointwise product (denoted by $\odot$) of the evaluations and finally, if possible, interpolate back (T^{-1}) the polynomial.

$$\begin{array}{ccccc} (R[x])^2 & \xrightarrow{\odot\circ T} & R^k & \xrightarrow{T^{-1}} & R[x] \\ (g(x), h(x)) & \mapsto & \begin{pmatrix} g(x_0) \\ g(x_1) \\ \vdots \\ g(x_{k-1}) \end{pmatrix} \odot \begin{pmatrix} h(x_0) \\ h(x_1) \\ \vdots \\ h(x_{k-1}) \end{pmatrix} & \mapsto & (g \cdot h)(x) \end{array}$$

In the following sections we will explore what possibilities do we have for R, k and $x_0, x_1, \dots, x_{k-1}$ so that T^{-1} is well-defined and both T and T^{-1} are efficiently computable. We are going to characterize the necessary and sufficient conditions the evaluation points have to satisfy to be able to perform these operations.

2.2.1 General Invertibility of T

To deal with the invertibility of transform T when applied to polynomials of degree bounded by n we notice it is a linear mapping and characterize it by its associated matrix

$$V = \begin{pmatrix} 1 & x_0 & {x_0}^2 & \cdots & {x_0}^{n-1} \\ 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k-1} & {x_{k-1}}^2 & \cdots & {x_{k-1}}^{n-1} \end{pmatrix}.$$

This special matrix is known as a Vandermonde matrix. If we choose $k = n$ we have a square Vandermonde matrix, and we can discuss its invertibility.

Since we are working with a general commutative ring with unity R, a matrix is invertible if and only if its determinant is invertible [118]. The determinant of a Vandermonde matrix is easy to compute and has the form

$$\det(V) = \prod_{0 \le i < j < n} (x_j - x_i).$$

Now if R is a field it is just sufficient to choose n different evaluation points $x_0, \dots, x_{n-1}$.

If R is only a commutative ring then it is a necessary and sufficient condition to choose points such that their differences are invertible in R.

Condition 1 (Points with invertible differences)**.** We say a set of points satisfies Condition 1 if the difference of every pair is invertible in R.

Remark 2.1. Choosing n evaluation points with invertible differences in R is a necessary and sufficient condition for the transform T to be invertible in $R[x]$.

Notice how we are talking about transforming and anti-transforming polynomials of a certain degree. Given two polynomials $g(x)$ and $h(x)$ of degrees n and n', since we want to recover the polynomial $g(x) \cdot h(x)$, we would have to think them as polynomials of degree smaller or equal than $n + n'$ and use $(n + n' + 1) \times (n + n' + 1)$ Vandermonde matrices.

2.2.2 Pointwise Product of Evaluations Modulo $f(x)$

Given $a \in R^*$ we can always consider polynomials in $R[x]/\langle x^n - a\rangle$ as polynomials in $R[x]$ with degree strictly bounded by n (using the canonical representative rem), compute, as we said before, their product as a polynomial in $R[x]$ with degree strictly bounded by $2n$ via a $2n$-transform and then applying $\text{rem } x^n - a$ again to obtain the representative with degree smaller than n.

The main issue we face when trying to use the pointwise product of evaluations technique to compute the product of two polynomials modulo $f(x)$ is that evaluation is not well-defined in general as it depends on the representative we choose from the class of equivalence modulo $f(x)$.

In general, for an arbitrary x_0, it is not the same to compute the product $(g(x) \ \text{rem} \, f(x))(x_0) \cdot (h(x) \ \text{rem} \, f(x))(x_0)$ than $(g(x) \cdot h(x) \ \text{rem} \, f(x))(x_0)$.

For this reason we have to choose specific points where evaluation is compatible with the congruence classes. If α is such that for any two equivalent polynomials $g(x) \equiv \widehat{g}(x) \pmod{f(x)}$ we obtain $g(\alpha) = \widehat{g}(\alpha)$ then we necessarily have $f(\alpha) = 0$ and α has to be a root of $f(x)$.

*One can see that in the special cases $a = \pm 1$ we have that the product in $R[x]/\langle x^n - 1\rangle$ and $R[x]/\langle x^n + 1\rangle$ is usually described in the literature as a cyclic/anti-cyclic convolution (denoted by $*$).

$$\begin{aligned}
(g \cdot h)(x) \quad (\text{rem } x^n - 1) &= g(x) \cdot h(x) \quad (\text{rem } x^n - 1) \\
&= \left(\sum_{i=0}^{n-1} g_i x^i\right)\left(\sum_{j=0}^{n-1} h_j x^j\right) \quad (\text{rem } x^n - 1) \\
&= \sum_{k=0}^{n-1} \left(\sum_{\substack{i,j \\ i+j \equiv k \\ \text{mod } n}} g_i \cdot h_j \right) x^k \\
&= (g * h)(x)
\end{aligned}$$

This intuition might be of independent interest as a convolution product in the regular domain is a pointwise product in the transformed domain.

Let α be a root of $f(x)$. Then $g(x) \equiv \widehat{g}(x) \pmod{f(x)}$ implies $g(\alpha) = \widehat{g}(\alpha)$ and therefore

$$(g(x) \ \operatorname{rem} f(x))(\alpha) \cdot (h(x) \ \operatorname{rem} f(x))(\alpha) = (g(x) \cdot h(x) \ \operatorname{rem} f(x))(\alpha).$$

In conclusion, for the particular case with $f(x) = x^n - a$, where $a \in R$, α has to be an nth root of a. By choosing $\alpha_0, \ldots, \alpha_{n-1}$ different nth roots of a with invertible differences we would be able to directly recover $g(x) \cdot h(x) \ \operatorname{rem} x^n - a$ from the pointwise product of their evaluations in $\alpha_0, \ldots, \alpha_{n-1}$.

This reduces the number of computations needed, since only n evaluation points are required (we directly recover the canonical representative $g(x) \cdot h(x) \ \operatorname{rem} x^n - a$, which has a degree bounded by n), and no further reduction is needed.

Condition 2 (Roots of $f(x)$ as points)**.** We say a set of points in R for a polynomial in $R[x]/\langle f(x)\rangle$ satisfies Condition 2 if they are roots of $f(x)$.

Remark 2.2. Choosing nth roots of a as evaluation points is a necessary and sufficient condition for the evaluations of polynomials in $R[x]/\langle x^n - a\rangle$ to be well-defined. We are going to consider this case.

So far Conditions 1 and 2 are necessary and sufficient conditions to use the pointwise product of evaluations as a technique to compute the product of two polynomials in $R[x]/\langle x^n - a\rangle$. The next step is to study when can this be computed efficiently.

2.3 Efficient Transforms

In this section we are going to define efficient transform and anti-transform algorithms from a theoretical and asymptotic point of view. Additional implementation tricks or approaches (for example, whether the recurrences are solved in an iterative or recursive way) could have an important impact to save up space or computations, but are out of the scope of this dissertation.

2.3.1 Efficient Evaluation of T

Once we have seen the requirements for the pointwise product of evaluations to work under each possible concerned circumstances we have to discuss how to efficiently apply them.

Computing each of the n evaluations individually would require $O(n)$ operations, for a total of $O(n^2)$. The Fast Fourier approach outperforms that computing the n evaluations at the same time by means of a divide-and-conquer recursive strategy.

If we call x_i to one of the evaluation points, and we decompose the polynomial $g(x)$ into two polynomials $g_0(x)$ and $g_1(x)$ of half the size with even and odd coefficients respectively, as in Dual Karatsuba, we can write

$$g(x_i) = g_0(x_i^2) + x_i \cdot g_1(x_i^2).$$

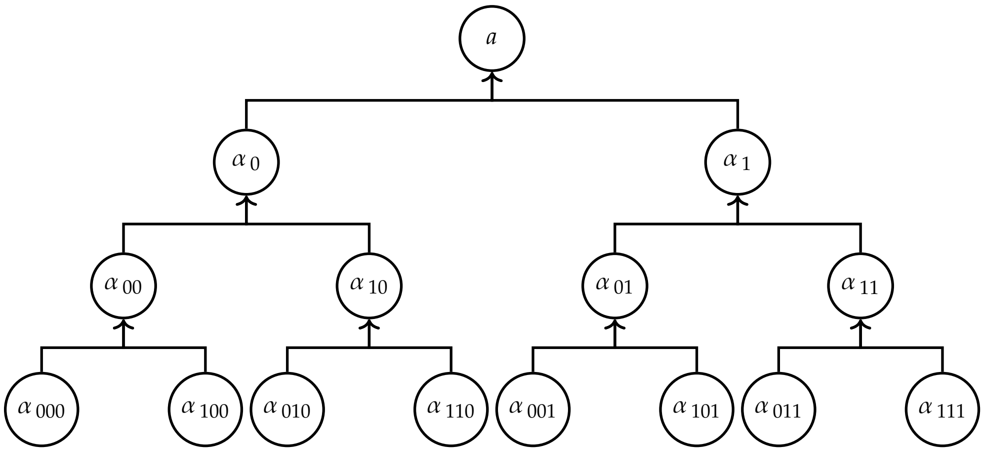
With this recursion we reduce a single polynomial evaluation of degree bounded by n to two polynomial evaluations of degree bounded by $^n/_2$, a product in R and an addition in R. Directly doing this would not save us any cost, as it would still take $O(n)$ per evaluation.

The main idea is to choose the evaluation points $\{x_0, x_1, \ldots, x_{n-1}\}$ so that the set containing their squares $\{y \mid y = {x_i}^2\}$ contains only $^n/_2$ elements (therefore we could reuse the evaluations of g_0 and g_1 on the squares). We would like that to be true recursively, so we introduce the following definition, already satisfying Condition 2.

Definition 2.3 (Twofold set of nth roots)**.** An indexed set $\alpha_0, \ldots, \alpha_{n-1} \in R$ (properly reindexed if required) of nth roots of an element $a \in R$, with n a power of 2, is said to be a *twofold set of nth roots* if $i \equiv j \pmod{2^{\log(n)-k}}$ implies ${\alpha_i}^{2^k} = {\alpha_j}^{2^k}$ for k from 0 to $\log(n)$.

We can visually represent it as a full binary tree like in Figure 2.1. Observe the evaluation points, leafs in the tree, appear in *bit-reversed order*.

Figure 2.1 Twofold set of 8th roots



Remark 2.4. Given $\alpha_0, \ldots, \alpha_{n-1}$ a twofold set of nth roots of $a \in R$ then the set $\{{\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2\}$ is a twofold set of $^n/_2$th roots of a.

Choosing as evaluation points a twofold set of roots $\alpha_0, \ldots, \alpha_{n-1}$ we have that for every $0 \leq i < n/2$ the equality ${\alpha_i}^2 = {\alpha_{i+n/2}}^2$ holds, and we can write

$$\begin{aligned} g(\alpha_i) &= g_0({\alpha_i}^2) + \alpha_i \cdot g_1({\alpha_i}^2), \\ g(\alpha_{i+n/2}) &= g_0({\alpha_i}^2) + \alpha_{i+n/2} \cdot g_1({\alpha_i}^2). \end{aligned}$$

We can use this property to present our first general description of a FFT Algorithm 2.3.

Algorithm 2.3 FFT

```
    Input: A polynomial $g(x)$ of degree bounded by $n$ and a twofold set $\alpha_0, \dots, \alpha_{n-1}$
     of $n$th roots of $a$
    Output: Evaluations of $g(x)$ at $\alpha_0, \dots, \alpha_{n-1}$
 1: if n = 1 then return $g$
    Split $g(x)$ into $g_0(x)$ and $g_1(x)$.
 2: $\boldsymbol{y}_0 := \mathrm{FFT}(g_0(x), {\alpha_0}^2, \dots, {\alpha_{n/2-1}}^2)$
 3: $\boldsymbol{y}_1 := \mathrm{FFT}(g_1(x), {\alpha_0}^2, \dots, {\alpha_{n/2-1}}^2)$
 4: return $(\boldsymbol{y}_0 + \boldsymbol{y}_1 \odot (\alpha_0, \dots, \alpha_{n/2-1})) \parallel (\boldsymbol{y}_0 + \boldsymbol{y}_1 \odot (\alpha_{n/2}, \dots, \alpha_{n-1}))$
```

Analyzing its computational cost now we find that computing the n-FFT takes as much time as computing two $n/2$-FFT plus a linear amount of products and additions in R ($T(n) = 2T(n/2) + O(n)$). Using again [20] we end up with a total cost $O(n \log n)$. Observe that now this is the total cost of the whole transform and not per evaluation as it was the case before.

Condition 3 (Twofold set of roots)**.**

We say a set of points satisfies Condition 3 if it is a twofold set.

Remark 2.5. Choosing the evaluation points as a twofold set of nth roots of $a \in R$ is a sufficient condition for the existence of an efficient FFT.

Observe that, by definition, Condition 3 implies Condition 2. However, we prefer to treat it separately as it is sufficient for an efficient implementation but unnecessary for a general pointwise product method.

As a way to simplify subsequent propositions dealing with the indexes of twofold sets we are going to use the following notation. For any $i \in \{0, 1, \dots, n-1\}$ let $\bar{\imath} = i + n/2 \operatorname{rem} n$. Analogously with j and $\bar{\jmath}$.

2.3.2 Efficient Evaluation of T^{-1}

In order to obtain an efficient multiplication algorithm we need not only an efficient transform algorithm but also an efficient anti-transform algorithm.

Notice beforehand that, the same way we saw in Remark 2.4 that Condition 3 is preserved when squaring the evaluation points we should check if the same holds with Condition 1. To do so we first require the following Lemma 2.6.

Lemma 2.6.
Let $\alpha_0, \dots, \alpha_{n-1}$ be a twofold set of nth roots of a with invertible differences. Then $\alpha_{\bar{i}} = -\alpha_i$.

Proof. Since the set satisfies Condition 3 we know ${\alpha_i}^2 = {\alpha_{\bar{i}}}^2$. That is

$$0 = {\alpha_i}^2 - {\alpha_{\bar{i}}}^2 = (\alpha_i - \alpha_{\bar{i}})(\alpha_i + \alpha_{\bar{i}}).$$

Using that the elements have invertible differences we obtain $\alpha_i + \alpha_{\bar{i}} = 0$. □

Proposition 2.7 (Squares of a set satisfying Conditions 1 to 3 also satisfy Conditions 1 to 3)**.** *Let $\alpha_0, \dots, \alpha_{n-1}$ be a twofold set of nth roots of a with invertible differences. Then the set ${\alpha_0}^2, \dots, {\alpha_{n/2-1}}^2$ is a twofold set of $n/2$th roots of a with invertible differences.*

Proof. From the definition of a twofold set it directly follows that the set of squares ${\alpha_0}^2, \dots, {\alpha_{n/2-1}}^2$ is a twofold set of $n/2$th roots of a.

We only need to check if the differences among the squares are still invertible.

$${\alpha_i}^2 - {\alpha_j}^2 = (\alpha_i - \alpha_j)(\alpha_i + \alpha_j) = (\alpha_i - \alpha_j)(\alpha_i - \alpha_{\bar{j}}).$$

Using Lemma 2.6 we have seen the differences among the squares are products of differences among original elements, invertible by hypothesis, implying the squares also satisfy the conditions. □

After these preliminaries one can define the anti-transform from a constructive point of view by reversing the transform algorithm or explicitating the inverse of the transform matrix. However, to get a deeper insight we are going to describe it using the language of Lagrange interpolation.

Given an indexed set of points $\{\alpha_i\}_{i=0}^{n-1}$ with invertible differences and the evaluations of a polynomial in such points $\{g(\alpha_i)\}_{i=0}^{n-1}$ we can recover the original polynomial $g(x)$ using Lagrange polynomials $l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x)$ as

$$g(x) = \sum_{i=0}^{n-1} g(\alpha_i) l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x), \quad l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n-1} \frac{x - \alpha_j}{\alpha_i - \alpha_j}.$$

Notice the Lagrange polynomials are well-defined because we ensure the evaluation points have invertible differences.

The key point is to observe that, due to the particular requirements of our set of evaluation points, that is Conditions 1 to 3, our Lagrange polynomials factorize in a special way.

Using Lemma 2.6 we can see how the Lagrange polynomial splits.

$$\begin{aligned}
l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) &= \prod_{\substack{j=0\\ j\neq i}}^{n-1} \frac{x-\alpha_j}{\alpha_i-\alpha_j} \\
&= \frac{x-\alpha_{\bar{i}}}{\alpha_i-\alpha_{\bar{i}}} \prod_{\substack{j=0\\ j\not\equiv i\\ (\mathrm{mod}\ n/2)}}^{n/2-1} \left(\frac{x-\alpha_j}{\alpha_i-\alpha_j}\right)\left(\frac{x-\alpha_{\bar{j}}}{\alpha_i-\alpha_{\bar{j}}}\right) \\
&= \frac{x-\alpha_{\bar{i}}}{\alpha_i-\alpha_{\bar{i}}} \prod_{\substack{j=0\\ j\not\equiv i\\ (\mathrm{mod}\ n/2)}}^{n/2-1} \frac{x^2-{\alpha_j}^2}{{\alpha_i}^2-{\alpha_j}^2} \\
&= l_i^{\{\alpha_i,\alpha_{\bar{i}}\}}(x)\, l_{i \text{ rem } n/2}^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x^2).
\end{aligned}$$

It is crucial to note that with a twofold set $l_{\bar{i} \text{ rem } n/2}^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x) = l_{i \text{ rem } n/2}^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x)$. Then we can write

$$\begin{aligned}
g(x) &= \sum_{i=0}^{n-1} g(\alpha_i)\, l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) \\
&= \sum_{i=0}^{n-1} g(\alpha_i)\, l_i^{\{\alpha_i,\alpha_{\bar{i}}\}}(x)\, l_{i \text{ rem } n/2}^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x) \\
&= \sum_{i=0}^{n/2-1} \left(g(\alpha_i)\, l_i^{\{\alpha_i,\alpha_{\bar{i}}\}}(x) + g(\alpha_{\bar{i}})\, l_{\bar{i}}^{\{\alpha_i,\alpha_{\bar{i}}\}}(x)\right) l_i^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x^2).
\end{aligned}$$

Once we have this decomposition the advantage of this language is that it allows us to interpret it. We were considering $g(x)$ as $g_0(x^2) + xg_1(x^2)$. Polynomials $l_i^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x)$ would help us interpolate $g_0(x)$ and $g_1(x)$ if we had their images $\{g_0({\alpha_i}^2)\}$ and $\{g_1({\alpha_i}^2)\}$.

However, the images we have are $\{g(\alpha_i)\}$. But then $l_i^{\{\alpha_i,\alpha_{\bar{i}}\}}(x)$ and $l_{\bar{i}}^{\{\alpha_i,\alpha_{\bar{i}}\}}(x)$ are precisely the polynomials that interpolate $g_0({\alpha_i}^2) + xg_1({\alpha_i}^2)$ (a polynomial of degree 1 that has the desired evaluations as coefficients) from $g(\alpha_i)$ and $g(\alpha_{\bar{i}})$.

As we are going to use it later lets explicitate

$$g(\alpha_i)l_i^{\{\alpha_i,\alpha_{\bar{i}}\}}(x) + g(\alpha_{\bar{i}})l_{\bar{i}}^{\{\alpha_i,\alpha_{\bar{i}}\}}(x) = \alpha_i\left(\frac{g(\alpha_i)+g(\alpha_{\bar{i}})}{\alpha_i-\alpha_{\bar{i}}}\right) + x\left(\frac{g(\alpha_i)-g(\alpha_{\bar{i}})}{\alpha_i-\alpha_{\bar{i}}}\right).$$

This can be used to build an efficient interpolation algorithm. From the n evaluations of a polynomial g of degree bounded by n at points $\{\alpha_i\}_{i=0}^{n-1}$ we can recover the evaluations of polynomials g_0 and g_1 at points $\{{\alpha_i}^2\}_{i=0}^{n/2-1}$ (this is done with interpolations of polynomials of degree bounded by 2, so each requires a constant time, and we need a total of $\mathcal{O}(n)$ operations).

Then we use them to interpolate g_0 and g_1, each of them polynomials of degree bounded by $n/2$ defined in $R[x]/\langle x^{n/2}-a\rangle$. That is $T(n) = 2T(n/2) + \mathcal{O}(n)$, and we end up again achieving $T(n) = \mathcal{O}(n\log n)$.

Algorithm 2.4 IFFT

```
Input: A vector of evaluations $\boldsymbol{y}$ of size $n$ and a twofold set $\alpha_0,\dots,\alpha_{n-1}$ of $n$th roots of $a$ with invertible differences
Output: Coefficients of a polynomial $g(x)$ interpolating $\boldsymbol{y}$ at $\alpha_0,\dots,\alpha_{n-1}$
1: if n = 1 then return $y$
2: for $i \in 0,\dots,n/2-1$ do
3:     $\boldsymbol{y}_0[i] := \alpha_i\left(\frac{\boldsymbol{y}[i]+\boldsymbol{y}[i+n/2]}{\alpha_i-\alpha_{i+n/2}}\right)$
4:     $\boldsymbol{y}_1[i] := \left(\frac{\boldsymbol{y}[i]-\boldsymbol{y}[i+n/2]}{\alpha_i-\alpha_{i+n/2}}\right)$
5: end for
6: $g_0(x) :=$ IFFT($\boldsymbol{y}_0, {\alpha_0}^2,\dots,{\alpha_{n/2-1}}^2$)
7: $g_1(x) :=$ IFFT($\boldsymbol{y}_1, {\alpha_0}^2,\dots,{\alpha_{n/2-1}}^2$)
8: return $g_0(x^2) + xg_1(x^2)$
```

2.3.3 Efficient Multiplication Algorithm in $R[x]/\langle x^n - a\rangle$

Combining both Algorithms 2.3 and 2.4 we describe in Algorithm 2.5 an efficient multiplication algorithm in $R[x]/\langle x^n - a\rangle$.

Thereby our work is to study the existence of sets of evaluation points satisfying Conditions 1 to 3 and how to find them in our desired R.

Algorithm 2.5 Efficient FFT Multiplication

```
    Input: Two polynomials $g(x)$, $h(x)$ of degree bounded by $n$
    Auxiliary: A twofold set $\alpha_0, \ldots, \alpha_{n-1}$ of $n$th roots of $a$ with invertible differences
    Output: The product $(g \cdot h)(x)$ of $g(x)$ and $h(x)$ in $R[x]/\langle x^n - a\rangle$
1: $\boldsymbol{g} := \mathrm{FFT}(g(x), \alpha_0, \ldots, \alpha_{n-1})$
2: $\boldsymbol{h} := \mathrm{FFT}(h(x), \alpha_0, \ldots, \alpha_{n-1})$
3: $\boldsymbol{f} := \boldsymbol{g} \odot \boldsymbol{h}$
4: $f(x) := \mathrm{IFFT}(\boldsymbol{f}, \alpha_0, \ldots, \alpha_{n-1})$
5: return $f(x)$
```

2.4 Characterization of suitable sets of evaluation points in the ring $\mathbb{Z}_m[x]/\langle x^n - a\rangle$

In this section we focus on $\mathbb{Z}_m$ and study the relations among the given conditions to see that these are precisely the required notions and provide necessary and sufficient conditions for the existence of proper evaluation sets.

We can start certifying that Conditions 1 to 3 are indeed independent in general.

Proposition 2.8 (Condition 3 does not imply Condition 1 in $\mathbb{Z}_m$ if m is not a power of a prime). *Let $\alpha_0, \ldots, \alpha_{n-1}$ be a twofold set of nth roots of a in $\mathbb{Z}_m$, where m is not a power of a prime. There is a twofold set of nth roots of a in $\mathbb{Z}_m$ without invertible differences.*

Proof. Decomposing $m = pq$ with p and q coprime proper factors we can always construct another twofold set defining $\alpha'_i \equiv \alpha_i \pmod{p}$ but $\alpha'_i \equiv \alpha_0 \pmod{q}$. It would be a twofold set, but none of their differences would be invertible. $\square$

When the modulus is a power of a prime p^e invertibility comes from being different modulo p, which is not implied in general by being different modulo p^e. We have first to further characterize nth roots in $\mathbb{Z}_{p^e}$, when p is prime, to address this particular case.

Theorem 2.9 (Hensel's lemma as in Theorem 2.23 from [93]). *Suppose that $f(x)$ is a polynomial with integral coefficients. If $f(x_0) \equiv 0 \pmod{p^e}$ and $f'(x_0) \not\equiv 0 \pmod{p}$, then there is a unique $t \pmod{p}$ such that $f(x_0 + tp^e) \equiv 0 \pmod{p^{e+1}}$.*

In our case we are particularly interested in $f(x) = x^n - a$, with n a power of 2, so $f'(x) = nx^{n-1}$. Since our solutions are nth roots of a (when considered modulo p^e and therefore also modulo p) $n{x_0}^{n-1} \not\equiv 0 \pmod{p}$ as long as $p \neq 2$ and $a \neq 0$.

Corollary 2.10. *There is a one to one correspondence of nth roots in $\mathbb{Z}_{p^e}$ and in $\mathbb{Z}_p$, where p is an odd prime.*

Proof.

On the one hand we can see each root in $\mathbb{Z}_{p^e}$ as a root in $\mathbb{Z}_p$ by applying rem p.

On the other hand one just needs to apply Theorem 2.9 iteratively $e-1$ times from $\mathbb{Z}_p$ to $\mathbb{Z}_{p^e}$. □

In particular this directly implies the order of nth roots of 1, as elements of the group, is preserved, given that powers of a root are uniquely lifted to powers of the lifted root.

Remark 2.11. We omit here the case $p = 2$ as in the following sections we are going to see that other conditions forbid this particular case.

Proposition 2.12 (Condition 2 implies Condition 1 in $\mathbb{Z}_m$ if m is a power of an odd prime)**.** *Let $m = p^e$, with p an odd prime, and let $\alpha_0, \dots, \alpha_{n-1}$ be n different nth roots of a in $\mathbb{Z}_{p^e}$. Then $\alpha_i - \alpha_j$ is invertible in $\mathbb{Z}_{p^e}$ for all $i \neq j$.*

Proof. Every α_i is a root of $f(x) = x^n - a$ when considered modulo p. By the previous corollary we have $\alpha_i \not\equiv \alpha_j \pmod{p^e}$ implies $\alpha_i \not\equiv \alpha_j \pmod{p}$. Therefore, $\gcd(\alpha_i - \alpha_j, p) = 1$, as we wanted. □

Proposition 2.13 (Condition 3 implies Condition 1 in $\mathbb{Z}_m$ if m is a power of an odd prime)**.** *Let $m = p^e$, with p an odd prime, and let $\alpha_0, \dots, \alpha_{n-1}$ be a twofold set of nth roots of a in $\mathbb{Z}_{p^e}$. Then $\alpha_i - \alpha_j$ is invertible in $\mathbb{Z}_{p^e}$ for all $i \neq j$.*

Proof. Condition 3 implies Condition 2 and Condition 2 implies Condition 1. □

Working with an arbitrary modulus m, that has $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ as its prime decomposition, we can completely determine any $d \in \mathbb{Z}_m$, via the *Chinese Remainder Theorem* (CRT), from $d_{(i)}$ such that

$$\begin{aligned} d_{(1)} &\equiv d \pmod{{p_1}^{e_1}}, \\ d_{(2)} &\equiv d \pmod{{p_2}^{e_2}}, \\ &\vdots \\ d_{(k)} &\equiv d \pmod{{p_k}^{e_k}}. \end{aligned}$$

Using this representation we can prove the following theorem.

Theorem 2.14 (Conditions 1 and 2 hold if and only if Condition 2 holds modulo every ${p_j}^{e_j}$)**.** *Let $\alpha_0, \dots, \alpha_{n-1} \in \mathbb{Z}_m$ be a set of different nth roots of $a \in \mathbb{Z}_m$, and let $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ be the prime decomposition of an odd module m.*

The differences among these elements are invertible modulo m if and only if all the elements are still different when considered modulo any of the ${p_j}^{e_j}$.

Proof. Follows the same ideas as the previous propositions, since an element is invertible if and only if it is invertible modulo all the coprime factors of a factorization of its modulus, we have seen (Corollary 2.10) that such roots are different modulo ${p_j}^{e_j}$ if and only if they are different modulo p_j, and therefore have invertible differences. □

The same way we can see how other implications are not true in general either.

Proposition 2.15 (Conditions 1 and 2 do not imply Condition 3 in $\mathbb{Z}_m$ if m is not a power of a prime)**.** *Not every n-set of nth roots of $a \in \mathbb{Z}_m$ with invertible differences is a twofold set of nth roots. It is not the case in general, not even for roots of unity.*

Proof. Consider the set $\{12, 14, 18, 21\} \subset \mathbb{Z}_{65}$, of 4th roots of unity, and let us also compute its squares.

$$\begin{aligned} 12^4 &\equiv 1 \pmod{65}, & 12^2 &\equiv 14 \pmod{65},\\ 14^4 &\equiv 1 \pmod{65}, & 14^2 &\equiv 1 \pmod{65},\\ 18^4 &\equiv 1 \pmod{65}, & 18^2 &\equiv 64 \pmod{65},\\ 21^4 &\equiv 1 \pmod{65}, & 21^2 &\equiv 51 \pmod{65}. \end{aligned}$$

Even if all of them are indeed 4th roots of unity, and the set of their differences $\{2, 3, 4, 6, 7, 9\}$ only contains invertible elements modulo 65, we have seen it is not a twofold set because all the squares are different modulo 65. □

Observe that the evaluation points from the example are indeed, after some reorderings, a twofold set of nth roots of unity in $\mathbb{Z}_5$ and in $\mathbb{Z}_{13}$, but the *reorderings* are different.

$$\begin{aligned} 12^2 &\equiv 18^2 \pmod{5}, & 12^2 &\equiv 14^2 \pmod{13},\\ 14^2 &\equiv 21^2 \pmod{5}, & 18^2 &\equiv 21^2 \pmod{13}. \end{aligned}$$

Once again the proposition does hold if we work modulo a power of an odd prime. To prove it we require a couple of lemmas that show the important role of roots of unity and allow us to focus on them, with the goal of better understanding these orderings.

Lemma 2.16 (Conditions 1 and 2 imply the evaluation points are invertible)**.** *Let $\alpha_0, \ldots, \alpha_{n-1}$ be different nth roots of a in $\mathbb{Z}_m$ such that $\alpha_i - \alpha_j$ is invertible in $\mathbb{Z}_m$ for all $i \neq j$. Then α_i is invertible in $\mathbb{Z}_m$ for all i.*

Proof. Choose two indices i and j and let d be a square-free common divisor of α_i and m. Since ${\alpha_i}^n \equiv a \pmod{m}$ and ${\alpha_j}^n \equiv a \pmod{m}$ we have that $m | {\alpha_i}^n - {\alpha_j}^n$, implying that $d|\alpha_j$ (here we have to use that d is square free). We would finally get $d|\alpha_i - \alpha_j$, but we know $\gcd(\alpha_i - \alpha_j, m) = 1$ and therefore $d = 1$, proving $\gcd(\alpha_i, m) = 1$ for all i. □

Remark 2.17. If roots of a are invertible it directly follows that a itself has to be invertible. We will impose it when required, since this argument implies it is a necessary condition for the existence of the inverse transform T^{-1}.

Lemma 2.18 (Roots of a and roots of 1)**.** Let $a \in \mathbb{Z}_m$. The following two statements are equivalent:

(i) The set $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{Z}_m$ satisfies Conditions 1 and 2.

(ii) The set $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{Z}_m$ can be constructed from an invertible nth root of a, let us denote it α, and n different nth roots of unity $\omega_0, \ldots, \omega_{n-1}$ with invertible differences in $\mathbb{Z}_m$ such that $\alpha_i = \alpha\omega_i$.

Proof. Let us prove both implications:

- (i) $\Longrightarrow$ (ii)

  Let $\alpha_0, \ldots, \alpha_{n-1}$ be the roots satisfying the conditions.

  We can define $\alpha := \alpha_0$ (invertible by Lemma 2.16) and $\omega_i := \alpha_i \cdot {\alpha_0}^{-1}$. We can check ω_i are roots of unity, and their differences are invertible

$$(\omega_i - \omega_j)^{-1} = \left(\frac{\alpha_i}{\alpha_0} - \frac{\alpha_j}{\alpha_0}\right)^{-1} = \alpha_0 \cdot (\alpha_i - \alpha_j)^{-1}.$$

- (ii) $\Longrightarrow$ (i)

  Let $\alpha, \omega_0, \ldots, \omega_{n-1}$ be a set of roots satisfying the conditions.

  Define now $\alpha_i := \alpha \cdot \omega_i$. Again, by construction, all α_i are nth roots of a and their differences are invertible since

$$(\alpha_i - \alpha_j)^{-1} = (\alpha \cdot \omega_i - \alpha \cdot \omega_j)^{-1} = \alpha^{-1}(\omega_i - \omega_j)^{-1}.$$

□

This motivates the definition of a sufficient condition that, as we are going to see, will be necessary when m is a power of an odd prime.

Definition 2.19 ((α, ω)-set)**.** Let α be any nth root of an invertible $a \in R$ and ω an nth root of unity in R of order n whose powers have invertible differences. Then the set defined as $\alpha_i = \alpha\omega^i$ is said to be an *(α, ω)-set*.

Condition 4 ((α, ω)-set)**.** We say a set of evaluation points satisfies Condition 4 if (after some reordering) it is an (α, ω)-set for some $\alpha, \omega \in R$.

Remark 2.20. Let $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ be the prime decomposition of the module m. An nth root of unity $\omega \in \mathbb{Z}_m$ can be determined from $\omega_{(i)} \equiv \omega \pmod{p_i^{e_i}}$ and the order of ω is just the least common multiple of the orders of $\omega_{(i)}$ in $\mathbb{Z}_{p_i^{e_i}}$. Since we choose n to be a power of 2 and all the orders of each $\omega_{(i)}$ divide n the least common multiple is just going to be the maximum of the orders. Then, for ω to be an nth root of unity of order n it is only necessary that one of these $\omega_{(j)}$ has order n. However, as we also need to impose invertibility of the differences of its powers then every $\omega_{(i)}$ has to have order n in $\mathbb{Z}_{p_i^{e_i}}$.

Proposition 2.21 (Condition 4 implies Conditions 1 to 3)**.** *An (α, ω)-set in $\mathbb{Z}_m$ is a twofold set of nth roots of a with invertible differences.*

Proof. Let $\alpha_i := \alpha\omega^i$ and let $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ be the prime decomposition of m.

We can start checking Condition 1. From the proof of Lemma 2.18 we know an (α, ω)-set of nth roots satisfies Condition 1 if α is invertible and $\omega^i - \omega^j$ are invertible too.

Since a is invertible in $\mathbb{Z}_m$ then α as an nth root of a has to be invertible too. Otherwise, if $\alpha \equiv 0 \pmod{p_j}$ for some j then $a \equiv 0 \pmod{p_j}$ for the same j and it would not be invertible either, contradicting the statement.

The second condition is ensured from the definition. Therefore, every difference is invertible modulo m.

Condition 2 follows from the construction of an (α, ω)-set.

Finally, for Condition 3, let $i \equiv j \pmod{2^{\log(n)-k}}$ and, without loss of generality, assume $i \geq j$ and therefore $i = j + c \cdot 2^{\log(n)-k}$ for some non-negative integer c. Then

$$\alpha_i^{2^k} = \left(\alpha_0\omega^i\right)^{2^k} = \alpha_0^{2^k}\omega^{(j+c\cdot 2^{\log(n)-k})\cdot 2^k} = \alpha_0^{2^k}\omega^{j\cdot 2^k + c\cdot n} = \left(\alpha_0\omega^j\right)^{2^k} = \alpha_j^{2^k}.$$

□

This condition is quite convenient, for example it allows us to explicitly describe the transform from its Vandermonde matrix, as we mentioned before, in a compact way given that $(V)_{ij} = (\alpha\omega^i)^j$ and $(V^{-1})_{ij} = {}^1/_n(\alpha^{-1}\omega^{-j})^i$. Notice that the inverse matrix looks like ${}^1/_n$ times the transpose of the evaluation matrix of an $(\alpha^{-1}, \omega^{-1})$-set. If we were working in $\mathbb{Z}_m[x]/\langle x^n - 1\rangle$ with $\alpha = 1$, the transpose would be irrelevant as the matrix would be symmetric and the same efficient recursive evaluation techniques from the direct transform would work directly for its inverse.

However, even if Condition 4 implies Conditions 1 to 3, the converse is not true in general, as we are going to see. Besides that, we can see it holds for some particular cases, when m is a power of a prime.

Proposition 2.22 (Conditions 1 and 2 do imply Condition 4 in $\mathbb{Z}_m$ if m is a power of an odd prime)**.** *Let $\alpha_0, \ldots, \alpha_{n-1}$ be a set of nth roots of a with invertible differences in $\mathbb{Z}_{p^e}$, with p an odd prime. This is (except reordering) an (α, ω)-set of nth roots of a in $\mathbb{Z}_{p^e}$.*

Proof. From Lemma 2.18 we deduce the existence of an invertible α and $\omega_0, \ldots, \omega_{n-1}$ with invertible differences such that $\alpha_i = \alpha \cdot \omega_i$.

Given that $\mathbb{Z}_p$ is a field its multiplicative group is a cyclic group of order $p - 1$. Then the set of nth roots of unity in $\mathbb{Z}_p$ is also a group, and as a subgroup of a cyclic group it is also cyclic.

It is also known that $x^n - 1$ has at most n solutions modulo p (Theorem 2.6 from [93]), therefore the n roots $\omega_i \ \operatorname{rem} p$ (all different since $\alpha\omega_i \ \operatorname{rem} p$ are lifted to different points in $\mathbb{Z}_{p^e}$) form the whole cyclic group of roots of unity and in consequence are generated by one of them.

As there is a one to one correspondence among nth roots in $\mathbb{Z}_p$ and in $\mathbb{Z}_{p^e}$ the original $\omega_i \in \mathbb{Z}_{p^e}$ are generated too by one ω of order n. That is, there exists a permutation π such that $\omega_{\pi(i)} = \omega^i$.

After this reordering defined by π the set $\alpha_{\pi(0)}, \alpha_{\pi(1)}, \ldots, \alpha_{\pi(n-1)}$ is an (α, ω)-set. □

Proposition 2.23 (Conditions 1 and 2 do imply Condition 3 in $\mathbb{Z}_m$ if m is a power of an odd prime)**.** *Let $\alpha_0, \ldots, \alpha_{n-1}$ be a set of nth roots of a with invertible differences in $\mathbb{Z}_{p^e}$, with p an odd prime. This is (except reordering) a twofold-set of nth roots of a in $\mathbb{Z}_{p^e}$.*

Proof. Direct as we know by Proposition 2.22 that Conditions 1 and 2 imply Condition 4 in $\mathbb{Z}_m$ if m is a power of an odd prime and Condition 4 always implies Condition 3, as seen in Proposition 2.21. □

This new condition seems the right choice, and the FFT is usually introduced from constructions equivalent to this definition, but we should study first if restricting to this particular family of sets of evaluation points reduces the options for computing an FFT multiplication when m is not a power of an odd prime.

Once again Conditions 1 and 2 being true modulo every ${p_j}^{e_j}$ should imply that Condition 4 holds modulo every ${p_j}^{e_j}$, but the permutations might be different.

There are cases where these permutations allow Condition 3 to be true while Condition 4 is not.

Theorem 2.24 (Conditions 1 to 3 do not imply Condition 4 in general, but imply the existence of a set satisfying Condition 4)**.** *Let $\alpha_0, \ldots, \alpha_{n-1}$ be a twofold set of nth roots* *of a with invertible differences in $\mathbb{Z}_m$, where m is odd and not a power of a prime and n is a power of two greater than 2^2. Then a has an inverse in $\mathbb{Z}_m$, there is both a twofold set $\alpha'_0, \dots, \alpha'_{n-1}$ of nth roots of a in $\mathbb{Z}_m$ with invertible differences not satisfying Condition 4 and a set $\alpha''_i = \alpha\omega^i$ where α is an nth root of a and ω is a root of unity of order n with all its powers having invertible differences, that is, an (α, ω)-set.*

Proof. Following Lemma 2.16 we get a is invertible.

Let $m = {p_1}^{e_1}{p_2}^{e_2} \dots {p_k}^{e_k}$ be the prime decomposition of m.

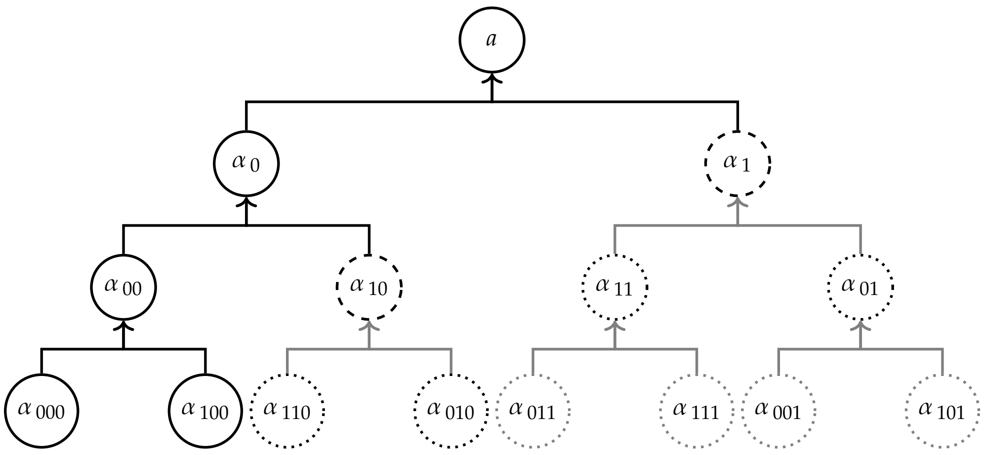
There is a unique (except reordering) twofold set of nth roots of a with invertible differences in $\mathbb{Z}_{{p_j}^{e_j}}$. This is the case since the values of each of the roots of a modulo ${p_j}^{e_j}$ are completely determined, being the unique elements lifted to $\mathbb{Z}_{{p_j}^{e_j}}$ from the unique n roots of a in $\mathbb{Z}_{p_j}$. By Proposition 2.22 it satisfies Condition 4. The only thing we could choose is the respective order they have.

This order is irrelevant in $\mathbb{Z}_{{p_j}^{e_j}}$ but becomes important when considering $\mathbb{Z}_m$ as once we fix an order modulo ${p_1}^{e_1}$ the different respective orders for the remaining ${p_j}^{e_j}$ would produce different elements in $\mathbb{Z}_m$.

From the twofold set definition we got a tree structure in Figure 2.1 and a specific notation for the points (a bit decomposition of the index). The twofold structure is only preserved by the tree structure, so the only possible reorderings are those that come from swapping the left and right children of a node. That is, choosing $b_{i-1} \dots b_0$ (an internal node), and mapping $\alpha_{b_{\log(n)-1} \dots b_{i+1} b_i b_{i-1} \dots b_0}$ to $\alpha_{b_{\log(n)-1} \dots b_{i+1} \overline{b_i} b_{i-1} \dots b_0}$, for all $b_{\log(n)-1}, \dots, b_{i+1}$, still preserves this structure.

For example, choosing nodes α_1 and α_{10} we obtain a different ordering like in Figure 2.2.

Figure 2.2 Twofold set of 8th roots swapping left and right descendants of α_1 and α_{10}



This means that, given a twofold set of nth roots, there are exactly 2^{n-1} possible reorderings (since the tree has exactly $n-1$ inner nodes, and we can swap or not each of them).

When odd m is not a power of a prime once we fix the order of the roots modulo ${p_1}^{e_1}$ each possible reordering of the roots modulo ${p_j}^{e_j}$ for the remaining j produces a new twofold set with invertible differences, for a total of $2^{(k-1)(n-1)}$ possibilities.

On the other hand, we can count the number of (α, ω)-sets. To do so it is important to notice that (α, ω) and a different pair (β, ξ) can generate the same set (just in a different order). If that is the case let $\pi \in \mathfrak{S}_n$ be the permutation such that $\beta_i = \alpha_{\pi(i)}$. Let $c = \pi(0)$, then $\beta = \beta_0 = \alpha_{\pi(0)} = \alpha\omega^c$. Let $d = \pi(1) - \pi(0)$, then $\xi = \beta_1/\beta_0 = \alpha_{\pi(1)}/\alpha_{\pi(0)} = (\alpha\omega^{\pi(1)})/(\alpha\omega^{\pi(0)}) = \omega^{\pi(1)-\pi(0)} = \omega^d$.

This allows us to completely characterize permutation π as

$$\beta_i = \beta\xi^i = \alpha\omega^c\left(\omega^d\right)^i = \alpha\omega^{c+di} = \alpha_{c+di}.$$

For π to be a permutation d has to be invertible modulo n, so it has to be odd.

That is everything required as any pair of $c \in \mathbb{Z}_n$ and odd $d \in \mathbb{Z}_n$ would produce a new set of generators $(\alpha\omega^c, \omega^d)$ for the same set. That is, each (α, ω)-set with invertible differences can be constructed from $n^2/2$ different pairs of roots of a and 1 (we have n options for c and $n/2$ options for d).

As a result, from the existence of a twofold set with invertible differences and previous propositions we know there are n possible $\alpha_{(j)}$ roots of a in $\mathbb{Z}_{{p_j}^{e_j}}$ and $n/2$ possible $\omega_{(j)}$ roots of unity in $\mathbb{Z}_{{p_j}^{e_j}}$ of order n that, when combined via the CRT, would define an (α'', ω'')-set with invertible differences. That is a total of $n^{2k}/2^k$ pairs of generators, and since every set is defined by $n^2/2$ pairs we would have $n^{2(k-1)}/2^{(k-1)}$ unique sets.

However, if $n > 2^2$ then $2^{(k-1)(n-1)} > n^{2(k-1)}/2^{(k-1)}$ and therefore some of the $2^{(k-1)(n-1)}$ sets $\{\alpha'_i\}_i$ satisfying Conditions 1 to 3 would not satisfy Condition 4.

□

This important theorem ensures that, even if Condition 4 is not necessary to design an efficient FFT multiplication algorithm, as it is sometimes indirectly taken for granted in the literature, we can safely assume it when working in $\mathbb{Z}_m[x]$ as it adds no additional restrictions on m to the necessary Conditions 1 to 3. Only having done this analysis we can safely use (α, ω)-sets when convenient without loosing any generality.

2.4.1 Existence and Construction of Suitable Roots in $\mathbb{Z}_m$

Once we have established the necessary and sufficient conditions for the transform to be useful to efficiently compute the product of two polynomials modulo $x^n - a$

we are left with the task of studying whether such points exist in our desired ring and how to find them.

As we have seen in Theorem 2.24 the existence of suitable points satisfying Conditions 1 to 3 implies the existence of an (α, ω)-set satisfying Condition 4. For convenience, we are going to characterize when such set of points exists and how to find one.

Remark 2.25. When $R = \mathbb{C}$ we just have to choose $\omega = e^{\frac{i2\pi}{n}}$ and $\alpha = \sqrt[n]{a}$ (for example $\alpha = 1$ if $a = 1$ or $\alpha = e^{\frac{i\pi}{n}}$ if $a = -1$). This choice is the standard Fourier Transform, and it is usually introduced directly in the literature.

Now we can consider the case $R = \mathbb{Z}_m$.

On the one hand, finding a primitive nth root of unity ω in $\mathbb{Z}_m$, that is, an nth root of unity of order n, such that all its powers have invertible differences, implies finding a primitive nth root of unity $\omega_{(i)}$ in every $\mathbb{Z}_{{p_i}^{e_i}}$ and then reconstruct ω using the CRT.

That way we have reduced the problem of finding a primitive nth root of unity in $\mathbb{Z}_m$ for an arbitrary m to finding a primitive nth root of unity in $\mathbb{Z}_{p^e}$.

The proof of Theorem 2.9 in [93] explicitly tells us how to lift a solution x_e modulo p^e to a solution x_{e+1} modulo p^{e+1}. It can be computed recursively using $x_{e+1} \equiv x_e - f(x_e)\overline{f'(x_1)} \pmod{p^{e+1}}$ where $\overline{f'(x_1)}$ denotes the inverse of $f'(x_1)$ when considering it in $\mathbb{Z}_p$. Recall $f(x)$ was $x^n - a$ and therefore $f'(x) = nx^{n-1}$.

The only step of this computation that is not immediate is to compute $\overline{f'(x_1)}$. Since we are sure $f'(x_1)$ is not 0 modulo p we can use the Extended Euclidean algorithm to compute integers r and s so that $f'(x_1)r + ps = \gcd(f'(x_1), p) = 1$, and r would be the desired $\overline{f'(x_1)}$.

We finally want to analyze under which conditions on p and n do primitive nth roots of unity exist in $\mathbb{Z}_p$ and how to find them. We obtain necessary conditions from Fermat's Little Theorem 2.26.

Theorem 2.26 (Fermat's Little Theorem as in Theorem 2.7 from [93])**.**
Let p be a prime. If $p \nmid x_0$ then

$$x_0{}^{p-1} \equiv 1 \pmod{p}.$$

Corollary 2.27. *If $\mathbb{Z}_p$ contains an nth root of unity of order n then $n \mid p - 1$.*

Proof. Let x_0 be an nth root of unity of order n in $\mathbb{Z}_p$. By Theorem 2.26 we have $x_0{}^{p-1} \equiv 1 \pmod{p}$ and therefore its order divides $p-1$, that is, $n \mid p-1$ or $p = kn+1$ for an integer k. □

Remark 2.28. Taking advantage of this condition, in the particular case $f(x) = x^n \pm 1$, the calculation of $\overline{f'(x_1)}$, with x_1 an nth root of ∓ 1 modulo p, can now be computed explicitly as $\overline{f'(x_1)} = \pm x_1(p-1)/n$, as can be easily checked computing $\overline{f'(x_1)} \cdot f'(x_1)$ and getting $(\pm x_1(p-1)/n)(n{x_1}^{n-1}) \equiv 1 \pmod{p}$.

Remark 2.29. This necessary condition rules out all the additional considerations we were having about $n \not\equiv 0 \pmod{p_i}$, for example ensuring m has to be odd. Observe that in Algorithm 2.4 we have to compute a quotient with $\alpha_i - \alpha_{\bar{i}}$, that is, $2\alpha_i$ in the denominator. Requesting Conditions 1 to 3 always implies that twice the unity of the ring has to be invertible (via Lemmas 2.6 and 2.16).

Corollary 2.30. *$\mathbb{Z}_p$ contains nth roots of unity of order n if and only if $n \mid p-1$.*

Proof. We have already seen one implication, let us consider now $\mathbb{Z}_p$ with $n \mid p-1$. We know the multiplicative group $\mathbb{Z}_p^*$ is cyclic. Let g be a generator, it has order $p-1$ and since $n \mid p-1$ we can choose $\omega := g^{(p-1)/n}$. By construction, it would be an nth root of unity of order n. □

Remark 2.31. Notice that, by Dirichlet's theorem on arithmetic progressions, there are infinitely many primes of this form.

Now we can start computing $\alpha \pmod{p_j}$ and $\omega \pmod{p_j}$. Observe the previous proof of the latest corollary is not directly constructive, as it needs a known generator. Anyway, choosing u_j a quadratic nonresidue in $\mathbb{Z}_{p_j}$ we can let $\omega_{(j)} \equiv u_j^{(p_j-1)/n} \in \mathbb{Z}_{p_j}$ and then lift it to $\mathbb{Z}_{{p_j}^{e_j}}$ using again the constructive proof of Theorem 2.9.

Notice first that there is no known deterministic polynomial-time algorithm able to find a quadratic nonresidue (see [15]). Nevertheless, checking if a uniformly random element u_j from $\mathbb{Z}_{p_j}^*$ is a quadratic nonresidue can be done computing the Legendre symbol $\left(\frac{u_j}{p_j}\right) \equiv {u_j}^{(p_j-1)/2}$, as it is -1 if and only if u_j is a quadratic nonresidue modulo p (see Theorem 3.1 from [93]). This procedure has a success probability of almost one half.

Provided that we know $n \mid p_j - 1$ is a condition for the existence of appropriate roots, we can ensure $\omega_{(j)} := {u_j}^{(p_j-1)/n}$ is well-defined. Its nth power is ${\omega_{(j)}}^n = {u_j}^{p_j-1} = 1$ (applying this time Theorem 2.26).

The only thing we have left is to check all its powers have invertible differences. If it was not the case then $\omega_{(j)}$ would have order k with $k < n$. This cannot be possible because since we already know ${\omega_{(j)}}^n = 1$ then it would imply $k|n$, and if k was a power of 2 strictly smaller than n we would get a contradiction as, by construction, $-1 = u_j^{(p_j-1)/2} = {\omega_{(j)}}^{n/2} = \left({\omega_{(j)}}^k\right)^{n/(2k)} = 1^{n/(2k)} = 1$.

This ensures our final ω, obtained using the CRT with all the $\omega_{(j)}$ lifted to $\mathbb{Z}_{{p_j}^{e_j}}$, meets the required conditions.

For computing α we might have different approaches. If $a = 1$ the trivial solution $\alpha = 1$ works perfectly fine for our purposes. Similarly, if $a = -1$ we can follow an analogous procedure, noticing that as nth roots of -1 are $2n$th roots of 1 the condition is now that $2n \mid p_j - 1$, and let $\alpha_{(j)} \equiv {u_j}^{(p_j-1)/2n} \pmod{p_j}$, once again lifting them and computing the final α from its CRT representation.

If $a \neq \pm 1$ then our alternative would be to make use of the general Tonelli–Shanks Algorithm [107] for computing square roots.

Algorithm 2.6 Tonelli–Shanks from ([107])

```
    **Input:** An odd prime $p$, a quadratic residue $a \in \mathbb{Z}_p$ and a nonresidue $u \in \mathbb{Z}_p$
    **Output:** An element $\alpha \in \mathbb{Z}_p$ square root of $a$
1:  Let $v$ and $s$ be such that $p - 1 = v2^s$ and $v$ is odd.
2:  Let
        $k \leftarrow s$
        $c \leftarrow u^v$
        $t \leftarrow a^v$
        $r \leftarrow a^{\frac{v+1}{2}}$
3:  **while** $t \neq 0$ **and** $t \neq 1$ **do**
4:      Find least $i$, $0 < i < k$, such that $t^{2^i} = 1$
5:      Let $d \leftarrow c^{2^{k-i-1}}$ and set
            $k \leftarrow i$
            $c \leftarrow d^2$
            $t \leftarrow td^2$
            $r \leftarrow rd$
6:  **end while**
7:  **if** $t = 0$ **then return** $\alpha = 0$
8:  **if** $t = 1$ **then return** $\alpha = r$
```

This algorithm allows us to efficiently compute a square root of a quadratic residue in $\mathbb{Z}_p$. Note this algorithm uses again as an auxiliary element a quadratic nonresidue. Iteratively applying it we can use ${\alpha_{(j)}}^{n/2^i} \pmod{p_j}$, the 2^ith root of a, to compute its square root ${\alpha_{(j)}}^{n/2^{i+1}}$, until we finally reach $\alpha_{(j)}$, from which we can recover α.

Coming back again to the case $x^n + 1$, we describe every step to compute (α, ω) in Algorithm 2.7.

Algorithm 2.7 Computation of α and ω

```
    Input: A power of two $n$ and a modulus $m$ (with known factorization)
    Output: Suitable $(\alpha, \omega)$ roots of $-1$ and $1$
    Let $m = {p_1}^{e_1}{p_2}^{e_2} \dots {p_k}^{e_k}$ be the prime decomposition of $m$.
 1: Ensure $2n \mid p_j - 1$ for every prime and abort otherwise.
 2: for $j \in 1, \dots, k$ do
    ▷ obtain a quadratic nonresidue in $\mathbb{Z}_{p_j}$ ◁
 3:     $tests$ := False                          ▷ whether a candidate is a nonresidue
 4:     while not $tests$ do
 5:         $u_j \leftarrow_R \mathbb{Z}_{p_j}$                       ▷ choose a nonresidue candidate
 6:         if $u_j^{(p_j-1)/2} \equiv -1 \pmod{p_j}$ then
 7:             $tests$ := True                              ▷ it is a nonresidue
 8:     end while
    ▷ compute $\alpha$ and $\omega$ $\pmod{{p_j}^{e_j}}$ ◁
 9:     $\alpha_{(j)} := u_j^{(p-1)/2n}$                        ▷ compute $\alpha \pmod{p_j}$
10:     $\omega_{(j)} := u_j^{(p-1)/n}$                         ▷ compute $\omega \pmod{p_j}$
11:     $c, aux \leftarrow$ ExtEuclides$(n\alpha_{(j)}^{2n-1}, p_j)$           ▷ compute $\overline{f'(\alpha_{(j)})}$
12:     $d, aux \leftarrow$ ExtEuclides$(n\omega_{(j)}^{n-1}, p_j)$           ▷ compute $\overline{f'(\omega_{(j)})}$
13:     for $e \in 2, \dots, e_j$ do                        ▷ apply Hensel Lemma
14:         $\alpha_{(j)} \leftarrow \alpha_{(j)} - (\alpha_{(j)}^n + 1)c \;\text{ rem }\; {p_j}^e$     ▷ lift from ${p_j}^{e-1}$ to ${p_j}^e$
15:         $\omega_{(j)} \leftarrow \omega_{(j)} - (\omega_{(j)}^n - 1)d \;\text{ rem }\; {p_j}^e$     ▷ lift from ${p_j}^{e-1}$ to ${p_j}^e$
16:     end for
17: end for
18: Reconstruct $\alpha$ from $(\alpha_{(1)}, \dots, \alpha_{(k)})$                 ▷ via the CRT
19: Reconstruct $\omega$ from $(\omega_{(1)}, \dots, \omega_{(k)})$                 ▷ via the CRT
20: return $(\alpha, \omega)$
```

2.5 FFT Generalizations

As we have seen in Section 2.4 we only have suitable evaluation points in the ring $\mathbb{Z}_m[x]/\langle x^n+1\rangle$ if $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ is such that every $p_i \equiv 1 \pmod{2n}$.

These congruences are deeply related to the factorization of x^n+1 (irreducible in $\mathbb{Z}[x]$) when considered modulo m. This has been described in [86], where Theorem 2.32 is presented.

Theorem 2.32 (Corollary 1.2 in [86])**.** *Let $n \geq d > 1$ be powers of 2 and $p \equiv 2d+1 \pmod{4d}$ be a prime. Then the polynomial x^n+1 factors as*

$$x^n + 1 \equiv \prod_{j=0}^{d-1}\left(x^{n/d} - \alpha_j\right) \pmod{p}$$

for distinct $\alpha_j \in \mathbb{Z}_p^*$, where $x^{n/d} - \alpha_j$ are irreducible in $\mathbb{Z}_p[x]$.

Theorem 2.32 can also be generalized to a not necessarily prime modulus m taking advantage of the results discussed in the previous sections.

Theorem 2.33 (Generalization of Theorem 2.32 to a not necessarily prime modulus m)**.** *Let $n \geq d > 1$ be powers of 2, $m = {p_1}^{e_1}{p_2}^{e_2}\dots{p_k}^{e_k}$ the prime decomposition of m such that $p_i \equiv 2d+1 \pmod{4d}$. Then the polynomial x^n+1 factors as*

$$x^n + 1 \equiv \prod_{j=0}^{d-1}\left(x^{n/d} - \alpha_j\right) \pmod{m}$$

for distinct $\alpha_j \in \mathbb{Z}_m^*$.

Proof. Notice first $p_i \equiv 2d+1 \pmod{4d}$ implies $p_i \equiv 1 \pmod{2d}$, therefore by the results discussed in Section 2.4.1 we know there exists a twofold set of dth roots of -1 with invertible differences $\alpha_0, \dots, \alpha_{d-1}$. From Lemma 2.16 we know each $\alpha_i \in \mathbb{Z}_m^*$. We also know from Lemma 2.6 that

$$\prod_{j=0}^{d-1}\left(x^{n/d} - \alpha_j\right) \equiv \prod_{j=0}^{d/2-1}\left(x^{n/d} - \alpha_j\right)\left(x^{n/d} - \alpha_{\bar{j}}\right) \equiv \prod_{j=0}^{d/2-1}\left(x^{2n/d} - {\alpha_j}^2\right).$$

Given that squaring preserves the initial properties (as seen in Proposition 2.7), iteratively applying the same idea we finally get $\prod_{j=0}^{d-1}\left(x^{n/d} - \alpha_j\right) \equiv x^n + 1 \pmod{m}$ as desired. □

This implies that, under the necessary conditions for an FFT multiplication algorithm, $x^n + 1$ fully splits in linear factors when considered modulo m. However, sometimes this is not the case, and we are enforced to use modulus that specifically require $x^n + 1$ to split in a smaller number of factors.

That is the case of some cryptographic constructions that use Theorem 2.32 from [86] (or similar versions) to guarantee the invertibility of particular subsets of elements in $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$, as it is going to happen with the commitment scheme that we are going to propose in the following chapter.

The condition $p_i \equiv 2d + 1 \pmod{4d}$ implies that there are dth roots of -1 in $\mathbb{Z}_m$, but no $2d$th roots of -1. Therefore, if $d < n$ no suitable evaluation points exist.

However, having only dth roots of -1 does not prevent us from finding a reasonably efficient multiplication algorithm. We cannot complete the recursion strategy, but we can still partially use it.

To do so we just need to apply a technique called ${}^{n}/_{d}$-degree *striding* (as described in [22]), mapping our polynomials to a more convenient ring, defining an auxiliary new variable $y = x^{n/d}$. We can always consider $R[x]/\langle x^n + 1\rangle$ as a subring of $R[x, y]/\left\langle x^{n/d} - y, x^n + 1\right\rangle$, and observe we can also describe this second ring as $R[x][y]/\left\langle x^{n/d} - y, y^d + 1\right\rangle$.

With this simple change of variables we can now represent our original polynomial as a polynomial in y with y-degree bounded by d that has as coefficients polynomials in $R[x]$ of x-degree bounded by n/d. As a polynomial in y it satisfies all required conditions as we are only considering modulus $y^d + 1$ and the new ring $R' = R[x]$ does contain d evaluation points $\alpha_0, \dots, \alpha_{d-1}$ satisfying Conditions 1 to 3.

With these dth roots of -1 we can efficiently use $g(x, y)$ and $h(x, y)$ to compute evaluations $\{g(x, \alpha_i)\}$ and $\{h(x, \alpha_i)\}$, do a pointwise product in $R[x]$ (using an auxiliary multiplication algorithm, such as (Dual) Karatsuba) and invert the transform to recover the product $(g \cdot h)(x, y)$ that directly gives us the desired solution substituting again y with $x^{n/d}$.

Observe that, the same way we have to choose an α that is a dth root of -1 to make the evaluation compatible with the quotient $\left\langle y^d + 1\right\rangle$ the other quotient $\left\langle x^{n/d} - y\right\rangle$ implies that when computing the evaluation of the variable y at α we obtain as a result a polynomial in $R[x]/\left\langle x^{n/d} - \alpha\right\rangle$, which is then only defined modulo $x^{n/d} - \alpha$. This is an artifact of the technique due to the additional variable introduced that has no impact in the final result, but helps us keep these x-polynomials bounded when computing their products.

The running time for both the transform and the anti-transform is now $O(n \log d)$, added to the d products of polynomials of degree ${}^{n}/_{d}$ that require $O\left(({}^{n}/_{d})^{\log 3}\right)$ operations each, for a total of $O\left(n \log d + d({}^{n}/_{d})^{\log 3}\right)$.

From an abstract point of view, we could directly apply Algorithm 2.5, as it was described for a general ring R, and therefore we could use it just taking into account to which ring each element belongs. However, for the sake of readability, we explicitate this generalization in Algorithm 2.8.

Algorithm 2.8 Generalized Efficient FFT Multiplication

```
    Input: Two polynomials $g(x)$, $h(x)$ of degree bounded by $n$
    Auxiliary: A twofold set $\alpha_0, \ldots, \alpha_{d-1}$ of $d$th roots of $-1$ with invertible differences
    Output: The product $(g \cdot h)(x)$ of $g(x)$ and $h(x)$ in $\mathbb{Z}_m[x]/\langle x^n + 1\rangle$
 1: $\widehat{g}(y) := g(x) \ \text{rem} \ x^{n/d} - y$                ▷ polynomial in $(\mathbb{Z}_m[x])[y]$
 2: $\widehat{h}(y) := h(x) \ \text{rem} \ x^{n/d} - y$                ▷ polynomial in $(\mathbb{Z}_m[x])[y]$
 3: $\boldsymbol{g} := \text{FFT}(\widehat{g}(y), \alpha_0, \ldots, \alpha_{d-1})$     ▷ vector of polynomials in $\mathbb{Z}_m[x]$
 4: $\boldsymbol{h} := \text{FFT}(\widehat{h}(y), \alpha_0, \ldots, \alpha_{d-1})$     ▷ vector of polynomials in $\mathbb{Z}_m[x]$
    Define $\boldsymbol{f}$ a vector of polynomials in $\mathbb{Z}_m[x]$ of size $d$.
 5: for $i \in 0, \ldots, d-1$ do
 6:     $\boldsymbol{f}[i] := \text{Karatsuba}(\boldsymbol{g}[i], \boldsymbol{h}[i]) \ \text{rem} \ x^{n/d} - \alpha_i$
 7: end for
 8: $\widehat{f}(y) := \text{IFFT}(\boldsymbol{f}, \alpha_0, \ldots, \alpha_{d-1})$
 9: $f(x) := \widehat{f}(y) \ \text{rem} \ y - x^{n/d}$
10: return $f(x)$
```

2.5.1 Fast Chinese Remaindering

As we mentioned in the introduction, this particular issue of partially splitting rings, where we cannot directly apply the original full FFT to the initial polynomials, has been studied in [86] for rings with prime modulus from a different point of view, considering FFT-like algorithms for efficiently applying the CRT [85, 121].

Recall the evaluation $g(\alpha)$ of any polynomial in α is equivalent to computing its remainder after dividing by $x - \alpha$ (this is known as the Polynomial Remainder Theorem). The main idea of these CRT approaches is to consider evaluations at α_i as representatives for $g(x) \pmod{x - \alpha_i}$, sufficient for determining $g(x)$ via the CRT since $x^n + 1 \equiv \prod_{j=0}^{n-1} (x - \alpha_j) \pmod{m}$ (as we know from Theorems 2.32 and 2.33). The same kind of recursions apply, given that both $g(x) \ \text{rem} \ x - \alpha_i$ and $g(x) \ \text{rem} \ x - \alpha_{\bar{i}}$ can be computed from $g(x) \pmod{x^2 - {\alpha_i}^2}$ (in an equivalent manner to what we saw in Section 2.3).

We can see the vector whose components are these evaluations as the CRT representation of the polynomial. Pointwise multiplication of these vectors of evaluations for two polynomials $g(x)$ and $h(x)$ is just a multiplication in the CRT domain, and the interpolation consists of recovering the polynomial coefficients from the CRT representations.

It is essentially a different interpretation of the same idea. It allows the same analysis as the CRT representation over a set of factors can be computed even if $x^n - a$ does not fully split.

The important point is that this argument works even if $d \neq n$, and we could use it to represent any polynomial with d remainders $(\text{rem } x^{n/d} - \alpha_i)$, compute the pointwise products among polynomials of degree bounded by $^n/_d$ and then recover back the product polynomial modulo $x^n - a$.

Given a twofold set $\alpha_0, \dots, \alpha_{d-1}$ of dth roots of -1 with invertible differences, what we do in Algorithm 2.8 is precisely computing the reminders $(\text{rem } x^{n/d} - \alpha_i)$ that determine the original polynomials via the CRT.

However, we believe our previous presentation is still more direct and informative, since the generalization to a not necessarily prime modulus in a partially splitting ring, not explored in [86], came completely for free, while the interpretation using the CRT requires a much more technically involved analysis when the ring is not a Principal Ideal Domain, or not even a Unique Factorization Domain. In order to verify the hypothesis of the theorem one should check whether some ideals are comaximal to be able to ensure that every mapping is indeed an isomorphism.

Theorem 2.34 (CRT for $\mathbb{Z}_m[x]/\langle x^n - a\rangle$)**.** *Let $x^n - a = \prod_{i=0}^{d-1}(x^{n/d} - \alpha_i)$ where $\{\alpha_i\}_i$ are a twofold set of dth roots of a with differences invertible in $\mathbb{Z}_m$, then*

$$\mathbb{Z}_m[x]/\langle x^n - a\rangle \cong \mathbb{Z}_m[x]/\left\langle x^{n/d} - \alpha_0\right\rangle \times \cdots \times \mathbb{Z}_m[x]/\left\langle x^{n/d} - \alpha_{d-1}\right\rangle.$$

Proof. For convenience, we are going to label the d roots as $\alpha_{b_{\log(d)-1}\dots b_0}$ as we did in Figure 2.1.

From an argument analogous to Theorem 2.33 we know that we can write $x^{n/2^k} - \alpha_{b_{k-1}\dots b_0} = (x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0})(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})$.

We have to prove that $\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^k} - \alpha_{b_{k-1}\dots b_0}\right\rangle$ is isomorphic to

$$\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}\right\rangle \times \mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}\right\rangle.$$

We can define a map from $\mathbb{Z}_m[x]$ to

$$\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}\right\rangle \times \mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}\right\rangle$$

by computing rem, and the kernel would be

$$\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}\right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}\right\rangle.$$

That is, the map is also well-defined from

$$\mathbb{Z}_m[x] \Big/ \left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0} \right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0} \right\rangle .$$

So far this discussion has been completely general. However, for these particular polynomials we have

$$\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0} \right\rangle + \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0} \right\rangle \cong \mathbb{Z}_m[x].$$

This is the case since

$$(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}) - (x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}) = \alpha_{1b_{k-1}\dots b_0} - \alpha_{0b_{k-1}\dots b_0},$$

which is invertible in $\mathbb{Z}_m$ (as every difference of roots is invertible), implying

$$1 \in \left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0} \right\rangle + \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0} \right\rangle .$$

We can explicitly write this saying there are two polynomials $g(x)$ and $h(x)$ such that

$$(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0})g(x) + (x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})h(x) = 1.$$

On the one hand this implies that the map is surjective. From the previous identity we know any pair of polynomials $(a(x), b(x))$ has a preimage $b(x)g(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}) + a(x)h(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})$.

On the other hand this implies

$$\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0} \right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0} \right\rangle = \left\langle x^{n/2^{k}} - \alpha_{b_{k-1}\dots b_0} \right\rangle .$$

The right-hand side is directly a subset of the left-hand side. To see the other inclusion we can check that for any

$$a(x) \in \left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0} \right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0} \right\rangle ,$$

that is, there are polynomials $b(x)$ and $c(x)$ such that

$$a(x) = b(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}) = c(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}),$$

it is also true that

$$\begin{aligned}
a(x) &= a(x)\cdot 1\\
&= a(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0})g(x) + a(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})h(x)\\
&= c(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0})g(x)\\
&\quad +b(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0})(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0})h(x)\\
&= (c(x)g(x) + b(x)h(x))(x^{n/2^k} - \alpha_{b_{k-1}\dots b_0}),
\end{aligned}$$

and therefore $a(x) \in \left\langle x^{n/2^k} - \alpha_{b_{k-1}\dots b_0}\right\rangle$.

Summing up, the desired mapping is an isomorphism.

The main issue here is that the CRT is usually defined for principal ideal domains or at least unique factorization domains. If it is not the case, such as with our construction, we have to specifically check these additional properties, such as $\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\dots b_0}\right\rangle$ and $\left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\dots b_0}\right\rangle$ being comaximal. $\square$

We then have a polynomial $g(x) \in \mathbb{Z}_m[x]/\langle x^n - a\rangle$ and a twofold set of dth roots of a with invertible differences $\{\alpha_{b_{\log(d)-1}b_{\log(d)-2}\dots b_0}\}$.

Our goal is to efficiently compute all $g_{b_{\log(d)-1}\dots b_0}(x) = g(x) \pmod{x^{n/d} - \alpha_{b_{\log(d)-1}\dots b_0}}$ (rem $x^{n/d}-\alpha_{b_{\log(d)-1}\dots b_0}$).

To do so we start computing $g_0(x) = g(x)$ (rem $x^{n/2} - \alpha_0$) and $g_1(x) = g(x)$ (rem $x^{n/2} - \alpha_1$).

Then we notice $g_{0b_0}(x) = g(x)$ (rem $x^{n/4} - \alpha_{0b_0}$) $= g_{b_0}(x)$ (rem $x^{n/4} - \alpha_{0b_0}$) and $g_{1b_0}(x) = g(x)$ (rem $x^{n/4} - \alpha_{1b_0}$) $= g_{b_0}(x)$ (rem $x^{n/4} - \alpha_{1b_0}$).

In general, we can recursively compute $g_{b_ib_{i-1}\dots b_0}(x) = g(x)$ (rem $x^{n/2^{i+1}} - \alpha_{b_ib_{i-1}\dots b_0}$) $= g_{b_{i-1}\dots b_0}(x)$ (rem $x^{n/2^{i+1}} - \alpha_{b_ib_{i-1}\dots b_0}$). Therefore, at the ith level, computing each remainder takes $O(n/2^i)$ operations and has to be done $O(2^i)$ times, for a total cost of $O(n)$.

Observe computing $g_{b_{i-1}\dots b_0}(x)$ (rem $x^{n/2^{i+1}} - \alpha_{0b_{i-1}\dots b_0}$) is done taking the lower coefficients of $g_{b_{i-1}\dots b_0}(x)$ and adding the higher coefficients multiplied by $\alpha_{0b_{i-1}\dots b_0}$.

The same way, since $\alpha_{1b_{i-1}\dots b_0} = -\alpha_{0b_{i-1}\dots b_0}$ we have that $g_{b_{i-1}\dots b_0}(x)$ (rem $x^{n/2^{i+1}} - \alpha_{1b_{i-1}\dots b_0}$) is computed taking the lower coefficients of $g_{b_{i-1}\dots b_0}(x)$ and subtracting the higher coefficients multiplied by $\alpha_{0b_{i-1}\dots b_0}$.

Notice the multiplications are the same (and could be reused), and the only difference is that we add or subtract depending on the case.

The number of levels is $\log(d)$, and we end up with $O(n\log(d))$ operations to compute the CRT representation of $g(x)$ taking modulus over the d different polynomials of degree n/d.

Algorithm 2.9 FFT (CRT)

```
    Input: A polynomial $g(x)$ of degree bounded by $n$ and a twofold set $\alpha_0, \dots, \alpha_{d-1}$
     of $d$th roots of $a$ with invertible differences
    Output: Remainders of $g(x)$ when divided by $x^{n/d} - \alpha_0, \dots, x^{n/d} - \alpha_{d-1}$
 1: if $r = 1$ then return $g$
 2: $\{g_{b_{\log(d)-2}\dots b_0}\} := \mathrm{FFT}(g, {\alpha_0}^2, \dots, {\alpha_{d/2-1}}^2)$
 3: for $b_{\log(d)-2} \dots b_0 \in \{0,1\}^{\log(d)-1}$ do
          Split $g_{b_{\log(d)-2}\dots b_0}$ into $g^L_{b_{\log(d)-2}\dots b_0}$ and $g^H_{b_{\log(d)-2}\dots b_0}$
 4:       $g_{0b_{\log(d)-2}\dots b_0} := g^L_{b_{\log(d)-2}\dots b_0} + g^H_{b_{\log(d)-2}\dots b_0} \cdot \alpha_{0b_{\log(d)-2}\dots b_0}$
 5:       $g_{1b_{\log(d)-2}\dots b_0} := g^L_{b_{\log(d)-2}\dots b_0} - g^H_{b_{\log(d)-2}\dots b_0} \cdot \alpha_{0b_{\log(d)-2}\dots b_0}$
 6: end for
 7: return $\{g_{b_{\log(d)-1}b_{\log(d)-2}\dots b_0}\}$
```

At this point we could use Karatsuba's algorithm to multiply them, that is, a cost of $O\Big(d(n/d)^{\log(3)}\Big)$.

Then we have to invert these operations. To do so we could follow the same ideas inverting the operations at each level. In order to recover the lower part of $g_{b_{i-1}\dots b_0}(x)$ we add $g_{0b_{i-1}\dots b_0}(x) + g_{1b_{i-1}\dots b_0}(x)$ and divide by two (that is, multiply each coefficient by 2^{-1}). To recover the upper part we now subtract them computing $g_{0b_{i-1}\dots b_0}(x) - g_{1b_{i-1}\dots b_0}(x)$ and multiply each coefficient by $2^{-1}(\alpha_{0b_{i-1}\dots b_0})^{-1}$. The cost of these operations is again $O(n\log(d))$.

As we have to divide by 2 at each level one could just skip this step and divide by d at the end.

Algorithm 2.10 IFFT (CRT)

```
    Input: A CRT representation $g_0 \dots g_{d-1}$ of a polynomial $g(x)$ of degree bounded
     by $n$, and a twofold set $\alpha_0, \dots, \alpha_{d-1}$ of $d$th roots of $a$ with invertible differences
    Output: The polynomial $g(x)$
 1: if $d = 2$ then return $2^{-1}\left((g_0 + g_1) + x^{n/2}(g_0 + g_1) \cdot {\alpha_0}^{-1}\right)$
 2: for $b_{\log(d)-2} \dots b_0 \in \{0,1\}^{\log(d)-1}$ do
          $g_{b_{\log(d)-2}\dots b_0} := 2^{-1}(g_{0b_{\log(d)-2}\dots b_0} + g_{1b_{\log(d)-2}\dots b_0}) +$
 3:
              $x^{n/d}2^{-1}(g_{0b_{\log(d)-2}\dots b_0} - g_{1b_{\log(d)-2}\dots b_0}) \cdot {\alpha_{0b_{\log(d)-2}\dots b_0}}^{-1}$
 4: end for
 5: return IFFT($\{g_{b_{\log(d)-2}\dots b_0}\}, {\alpha_0}^2, \dots, {\alpha_{d/2-1}}^2$)
```

This alternative interpretation has then same properties and can also be used to design the same efficient multiplication algorithms. We reiterate here that we believe the previous presentation was more insightful, provided that some conditions, as the requirement for m to be odd, only appear as artifacts of the CRT construction.

2.6 Conclusions

The main result can be summarized in the following way, in order to design an efficient multiplication algorithm in $\mathbb{Z}_m[x]/\langle x^n - a\rangle$ via an FFT, the necessary and sufficient condition is to have a set of n different nth roots of a (Condition 2, so that multiplication is compatible with congruence classes) with invertible differences (Condition 1, so that the inverse transform is defined) such that its recursive squares are equal two by two (Condition 3, so that the computation can be efficiently done recursively).

This characterization is similar but not equivalent to the usual characterization with roots of unity (Condition 4), which is sufficient but not necessary. Despite that, we have proven in Theorem 2.24 that restricting to sets satisfying Condition 4 does not decrease the applicability of these efficient multiplication algorithms as the defined (α, ω)-sets exist if and only if the necessary and sufficient sets satisfying Conditions 1 to 3 exist.

As intermediate result we have also proven that these properties are indeed independent in the general case, and we do believe that this analysis might help to clarify whether some considerations and conditions usually stated in the folklore are fundamental considerations about the algebraic structure or just conventions for a particular instantiation on a particular setting (as it is the case with roots of unity).

This framework is also a general introduction to FFT multiplication from a rigorous mathematical point of view while still keeping it readable for an audience not familiarized with more advanced algebraic considerations.

For example our analysis directly generalizes, as we have seen in Section 2.5, to a ring where $x^n - a$ does not fully split and $\mathbb{Z}_m$ is not a field, while alternative interpretations are much more delicate to work with.

Chapter 3

Commitment scheme and companion ZKPoKs

This chapter and the following constitute the core contribution of this dissertation. Now that we have motivated the need of new post-quantum cryptographic primitives, explained the power of ZKPoKs and carefully analyzed the properties of the quotient rings of polynomials that allow efficient operations, we already have all the tools to develop improved ZKPoKs for lattice constructions.

Rather than presenting a bare proof of knowledge of the secret that characterizes a RLWE distribution we define a lattice-based commitment scheme and design interactive proofs of knowledge for several statements, including linear and multiplicative relations among committed elements. This work is going to be continued in the next chapter with the implementation of non-interactive versions of these proofs.

3.1 Introduction

The goal of this chapter is to design new ways of proving linear and multiplicative relations between elements hidden in lattice-based structures, without revealing any additional information about the elements themselves. This kind of proofs play an important role in many applications, from authentication protocols to electronic voting, and can be used as building blocks for more complex cryptographic primitives.

We choose to apply these new techniques to a commitment scheme and the relations between committed elements because the simple structure of the scheme allows us to explain what are the advantages and disadvantages of our methods and at the same time we provide a very flexible primitive with actual utility to solve a great variety of problems. In fact, the general strategy of proving statements about committed elements in zero-knowledge, called commit-and-prove, is an active research field on its own, and we further increase its lattice-based branch.

We focus on improving the applications of the seminal code-based Stern protocol [116, 117] (mentioned in Section 1.3.8) to the lattice setting. It provides the most efficient ZKPoK for code-based cryptography [114], but we use its ideas to deal with the lattice-based RLWE problem as underlying hardness assumption. We increase the efficiency of existing proofs and show how to obtain exact ZKPoKs convincing a verifier that the prover knows a valid opening for a commitment. Then we use this opening proof as the foundation for the relation proofs, showing that committed messages satisfy linear or multiplicative relations.

3.1.1 Lattice-Based Zero-Knowledge Proofs Related Work

In this chapter we specially benefit from the adaptation of Stern's protocol to lattices from Ling *et al.* [76], the modification of Cayrel *et al.* [38] for reducing the soundness error increasing the number of rounds and the proposals of Jain *et al.* [64] and Xie *et al.* [124] for proving linear and multiplicative relations, that we further improve.

It is also important to mention the contribution of Benhamouda *et al.* [19], who generalized the commitment idea of [124] without using Stern's approach. We still use Stern-based techniques for the proofs but borrow the notation and structure from [19], as it is the more natural adaptation of [64] to the lattice scenario. They instead use Fiat-Shamir with aborts (the alternative technique we have also briefly discussed in Section 1.3.8). This decision implies relaxing the definition of commitment for their construction, because they are forced to admit as valid a set of openings larger than the set of openings an honest prover could get when committing to an element (the prover obtain an opening satisfying a relation $\mathfrak{R}$ but is only able to prove a milder relation $\mathfrak{R}'$). In particular, even if the prover samples the noise terms from a bounded error distribution with bound B the verifier is only convinced of the fact that it is bounded by a significantly larger $B' \gg B$.

This has an impact on the restrictions the parameters have to satisfy. On the one hand, we have already seen in Section 1.3 the importance of the fact that the noise terms are small, which is extensively used in security proofs of lattice-based cryptography (in our case and in [19] it is used to prove that the commitment is binding). On the other hand, the error parameter has to be sufficiently large so that the underlying RLWE problem is hard enough. If from the same noise distribution bounded by B we are only able to prove knowledge of an error term bounded by $B' \gg B$ then we might be forced to use this less tight bound to prove the remaining properties, obtaining worse results due to that gap. On the contrary, our proposal increases the literature of exact zero-knowledge proofs, providing ZKPoKs of committed elements without any soundness slack. This allows us to tightly choose all parameters as we do not have to take into account any error growth.

This trade-off between efficiency and the soundness gap is a constant in the related literature. Bootle *et al.* [28] presented some succinct zero-knowledge arguments with an extreme soundness slack. In this case, a prover knowing a certain bound on the error term can only convince a verifier that they are able to extract an error which is exponentially larger.

Nevertheless, several improvements have been made on rejection sampling techniques for proving relations among committed elements. First, Baum *et al.* [17] greatly improved the efficiency of this approach using the MLWE problem as hardness assumption and changing the structure of the commitment, so the minimum size of the proofs is greatly reduced, though the binding property is ensured only computationally instead of statistically. While [17] already had an efficient proof for linear relations, the recent work of Attema *et al.* [14] provides new proofs for multiplicative relations for this commitment scheme. It has also been modified to provide even more efficient opening proofs in [119].

Some proposals as [29] or [51] try to circumvent the issues that arise from the relaxed definition of the commitment verifying additional conditions to obtain exact lattice-based proofs. A very prolific research path is the use of a relaxed commitment scheme with companion rejection sampling ZKPoKs as a subroutine of a more complex protocol that then obtains efficient exact proofs. As relaxed commitment scheme these applications use [17] (with small modifications). That is the case of [126, 50, 82, 80]. Exact lattice-based ZKPoKs are therefore an active field of research.

On the other hand, Stern-based techniques have been applied to prove knowledge of several relations (including multiplications) when committing (using lattices) to exponentially large integers using a polynomially large modulus q, as it was done by Libert *et al.* in [74], and to prove matrix-vector relations [75] again by Libert *et al.* The first has been outperformed by Kuchta *et al.* in [70] and also in [81] by Lyubashevsky *et al.*, using the rejection sampling scenario. A general technique useful to reduce the soundness error of Stern-based protocols was presented in [24], greatly reducing the proof sizes at the expense of significantly increasing the computational cost.

3.1.2 Our Contribution

The main contribution of this chapter is an improvement over the first two Stern-based ZKPoKs for linear and multiplicative relations from [64, 124]. Our ideas on proving multiplicative relations can be easily adapted to any scenario where messages are encoded as RLWE samples.

Regarding the commitment itself we follow the notation from the proposal of

Benhamouda *et al.* [19], explicitly defining two separated ideal lattices, one for encoding the message and another one for obtaining the RLWE sample that allows us to hide the message.

We get rid of the relaxations and limitations that were necessary in Benhamouda *et al.* commitment scheme without needing the quadratic logarithm of q overhead from Xie *et al.* for the proofs. For the opening and the linear relation cases we apply standard techniques to improve the original Stern protocol, increasing the number of moves to get a smaller soundness error. We also add some original modifications to carefully reduce some constants in the communication cost. For the multiplicative relation we construct a new efficient proof. We achieve this by asking the verifier for two challenges in order to relate the committed messages and get soundness. Honest-verifier zero-knowledge is obtained as we explicitly provide a perfect simulator for each protocol.

Many applications demand to evaluate arbitrary arithmetic circuits on secret elements. FHE could be a solution (which can be achieved with lattices by means of the Gentry *et al.* scheme [60], and the many subsequent FHE schemes that have been later proposed). Working with small circuits an alternative could be to apply our proofs for linear and multiplicative relations to prove knowledge of valid evaluations of the gates. The first lattice-based Attributed Based Signature scheme for unbounded circuits [47] uses this strategy with the ZKPoKs from [124]. Directly replacing their construction with our proposal greatly improves the efficiency of the signature scheme.

Our proposal is a 5-move protocol with a soundness error slightly above $1/2$. It allows us to prove exact knowledge of the secret inside a RLWE sample, that is, the secret is a polynomial in $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. The proposed commitment scheme is perfectly binding with overwhelming probability over the choice of the public key and computationally hiding under the RLWE assumption, widely believed to be post-quantum (as we have discussed in Section 1.3).

As we mentioned in Section 1.5, this chapter is extensively based on a previous version of this proposal that has been published in [87]. It has been however significantly improved since then. The more complete version included here has a tighter security analysis providing specific values, while [87] (or [19]) only proved some properties asymptotically. The more detailed analysis conducted here significantly helps the implementation and performance analysis we are going to present in the following chapter.

The soundness proof has also been completely revamped so that it satisfies the standard definitions. We have adapted the conditions on the ring R_q so that the efficient multiplication algorithm discussed in Chapter 2 can be used. Besides the computational advantage this additionally provides us with the potential trade-offs that arise from choosing d, the number of polynomials $x^n + 1$ splits when considered modulo q (in [87] this was fixed to 2 as in [19]). All security proofs have been updated to take it into account.

Furthermore, many secondary propositions necessary to ensure the security of the commitment that were omitted in [87] due to space restrictions of the publication are formally proved in this chapter.

3.1.3 Structure of the Chapter

The organization of this chapter is as follows. In Section 3.1 we have introduced the topic and the context of this work. Then in Section 3.2 we review the historic iterations and improvements of Stern-based protocols to have a deeper knowledge of the kind of techniques we can apply and fully contextualize the improvements we will introduce.

After these preliminaries we present the commitment in Section 3.3, along with the interactive proofs in Section 3.4. A proof of knowledge of a valid opening is designed and proven secure in Section 3.4.1, and analogously we then give proofs of a linear relation and a multiplicative relation in Sections 3.4.2 and 3.4.3, respectively. We finally end the chapter with some conclusions in Section 3.5.

Moreover, in Appendix 3.A we specify the differences between the proposal from this chapter and the version published in [87].

3.2 Stern-Based Schemes

Although we have already mentioned many Stern-based protocols in Section 3.1.1, we think it is interesting to start the chapter with a historical journey revisiting the main advances in this family of protocols. This way we can specially remark the ideas from which we explicitly benefit or that we further improve.

The original identification scheme by Stern, the one we have introduced in Section 1.3.8, was a 3-move protocol with a soundness error of $2/3$, but he also presented alternative variants with 5-moves and identity based versions [116]. One of the 5-move variants reduced the computational complexity and the other reduced the soundness error to almost $1/2$. However, the size of the proof increased, and it turned out to be less efficient for practical cases. Numerous additional variants have been published since then, addressing the lack of efficiency that comes from the $2/3$ soundness error and the size of the keys, and providing new features and applications (different signature schemes, commitment schemes, possibility of using integers module q as secrets instead of only using bits, applications to lattice-based cryptography, etc.). We describe some of them in the following paragraphs.

The first new scheme that claimed to improve the original protocol was due to Véron in 1997 [120], where he presented a dual version of the original Stern protocol. Given a generator matrix $\boldsymbol{G} \in \mathbb{F}_2^{n\times k}$ and given a perturbed codeword $\boldsymbol{t} \in \mathbb{F}_2^n$, he wanted to prove knowledge of a secret $\boldsymbol{x} \in \mathbb{F}_2^k$ and a low Hamming weight error term $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{t} = \boldsymbol{G}\boldsymbol{x} + \boldsymbol{e}$. He claimed that it was more efficient than the original Stern protocol, however it was not zero-knowledge. His proof was flawed because the simulation implicitly assumed that two elements were independently and uniformly random distributed. It was true that the elements were uniformly random distributed, but they were not independent. Therefore, the joint distribution of the protocol was different from the distribution of the simulator, and information about the secret could be extracted (as was later pointed out by Jain *et al.* in [64], who correctly used a generator matrix to build a sound zero-knowledge proof).

We mention here this failed proposal because it is not a single anecdote, as some subsequent articles with new identification schemes, signatures and plaintext knowledge proofs have been based on [120] and inherited the same lack of zero-knowledge [33, 3, 91] (the issue was noticed by the authors of [91] and fixed in [63]). Even worse is the case of [48], where besides implementing the unsecure scheme from [120] without noticing it, they also insecurely implement other two secure schemes [116, 38] (we will latter discus the last one [38]). In the interactive version of these protocols the verifier asks the prover three possible challenges, but to determine the challenge in a non-interactive way they hash the previous conversation and then map pairs of bits `00`, `01` and `10` to the different challenges (following the Fiat-Shamir paradigm). However, if the pair of bits from the hash is 11 they then assign one of the three challenges following a predetermined order in a cyclic way. This makes the challenges not independent, lowering the security and preventing a direct ROM sound proof. Other proposals also fail to obtain secure LWE-based identification schemes, as it is the case of [111], because they state the errors are sampled from a discrete Gaussian distribution and at the same time assume the errors have fixed Hamming weight (completely confusing the lattice and code assumptions), getting unsound proofs and unrealizable simulators for the zero-knowledge property.

Once again we remark the need of complete formal proofs of security to avoid this kind of mistakes.

In 2007, Gaborit and Girault [55] provided the first step towards a practical scheme, noticing that the use of quasi-cyclic linear codes (codes whose parity check matrix is built as a concatenation of an identity matrix and a circulant matrix) could considerably reduce the size of the keys preserving security, analogous to what we can do using ideal lattices instead of arbitrary q-ary lattices, as the SDP is still believed to be hard for this particular family of codes.

Their approach was first implemented in 2008 by Cayrel *et al.* in [35], remarking the fact that only simple operations as matrix-vector products are needed and therefore it is perfectly suitable for low-resource devices. Furthermore, they also show how to turn the original identification scheme into a signature scheme, using the Fiat-Shamir transform. They do not enter into detail albeit from the numbers they provide it seems they assume no security loss from the transform.

The first application to lattice cryptography was due to Kawachi *et al.* in [68], where they used an instance of the ISIS problem instead of the SDP. When we think about ISIS instances, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \in \mathbb{Z}_q^n$, we usually choose the small term $\boldsymbol{x} \in \mathbb{Z}_q^m$ from a Gaussian distribution. In this case $\boldsymbol{x}$ is selected to have half of its components equal to 0 and the other half equal to 1. Then they only need to check its Hamming weight, in a direct translation of what was done using codes.

The dimensions of the matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ are chosen so that this special ISIS problem has more than one solution and another small $\boldsymbol{x}'$ satisfying the equation exists. Security is then proved with the following reasoning, the protocol is zero-knowledge, therefore it is also witness indistinguishable (if the statement has more than one witness the verifier cannot distinguish which one is being used by the prover) and extracting two solutions (it is possible to efficiently extract two different solutions because of the witness indistinguishability property) we would have $\boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}') = \boldsymbol{0}$, a solution to the SIS problem.

This scheme is also interesting because it exemplifies the versatility of Stern-based protocols extending the original proposal to allow anonymous identification. Using N public keys $\boldsymbol{y}_i$, and increasing N times the proof size, the prover is able to prove knowledge of one of the secret keys without revealing to which one of the public keys it corresponds. The main idea involves user i using a standard basis vector $\boldsymbol{v}_i$, with zeroes everywhere except at index i where there is a 1, so that they can use it to vanish all the other public keys except their own with a matrix-vector product where the matrix has the public keys as columns and the vector is the ith standard basis vector $\boldsymbol{A}\boldsymbol{x} = [\boldsymbol{y}_1|\dots|\boldsymbol{y}_N]\boldsymbol{v}_i = \boldsymbol{y}_i$. Of course these standard basis vectors can also be described as binary vectors with Hamming weight equal to 1, and therefore the same proof techniques can be applied to hide which public key are they using in the anonymous identification scheme permuting the vector $\boldsymbol{v}_i$ to hide the position of the 1 while still ensuring it has Hamming weight 1.

A major improvement was done by Cayrel, Véron and El Yousfi when they efficiently used 5 moves to reduce the soundness error to almost $1/2$ in [38]. In this case the new strategy does decrease the total proof size for practical parameters.

Using q-ary codes instead of binary codes and an additional challenge $\alpha \leftarrow_{\mathrm{R}} \mathbb{F}_q$ they can combine two of the commitments and only need to check two properties to verify that $\boldsymbol{t} = \boldsymbol{He} \in \mathbb{F}_q^{n-k}$ with $\boldsymbol{e} \in \mathbb{F}_q^n$ having a fixed low Hamming weight w.

(a) The syndrome of $\boldsymbol{x} + \alpha\boldsymbol{e}$ is $\boldsymbol{y} + \alpha\boldsymbol{t}$.

(b) The Hamming weight of $\Pi(\boldsymbol{e})$ is w, for some Hamming weight preserving application Π.

If these two properties are true for two different α a valid witness could be extracted (here we omit the details here). This implies that the soundness error is approximately $1/2$. In this case the scheme efficiently cuts back the communication cost as it reduces the number of repetitions necessary to achieve the desired soundness without significantly increasing the size of a single repetition.

A permutation is not enough to completely hide a q-ary word with low Hamming weight, since it would still reveal each of the symbols, and they need to design a natural generalization. Let $\gamma \in \mathbb{F}_{q^*}^n$ be a vector of non-zero integers modulo q, and $\pi \in \mathfrak{S}_n$ a permutation. They define the application $\Pi_{\gamma,\pi} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$ that permutes the coordinates with π and then multiplies the result component-wise by γ. This application completely randomizes vectors of known Hamming weights, is compatible with addition and multiplication by integers and preserves Hamming weights.

The prover starts choosing a masking element $\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_q^n$, and computes a masking syndrome $\boldsymbol{y} := \boldsymbol{Hx} \in \mathbb{F}_q^{n-k}$. They also sample uniformly random $\pi \leftarrow_{\mathrm{R}} \mathfrak{S}_n$ and $\gamma \leftarrow_{\mathrm{R}} \mathbb{F}_{q^*}^n$. Next, they build only two commitments: $c_1 \leftarrow_{\mathrm{R}} \mathsf{Com}(\pi, \gamma, \boldsymbol{y})$, $c_2 \leftarrow_{\mathrm{R}} \mathsf{Com}(\Pi_{\pi,\gamma}(\boldsymbol{x}), \Pi_{\pi,\gamma}(\boldsymbol{e}))$.

The verifier chooses its integer challenge $\alpha \leftarrow_{\mathrm{R}} \mathbb{F}_q^n$ and the prover computes and sends $\boldsymbol{g} := \Pi_{\pi,\gamma}(\boldsymbol{x} + \alpha\boldsymbol{e})$.

Finally, the verifier chooses to verify one of these two properties.

(a) The syndrome of $\Pi_{\pi,\gamma}^{-1}(\boldsymbol{g})$ is $\boldsymbol{y} + \alpha\boldsymbol{t}$.

(b) $\boldsymbol{g}$ is constructed from $\Pi_{\pi,\gamma}(\boldsymbol{x})$ and $\alpha\Pi_{\pi,\gamma}(\boldsymbol{e})$ such that $\Pi_{\pi,\gamma}(\boldsymbol{e})$ has Hamming weight w.

Both properties combined imply the original properties, as the same $\boldsymbol{g}$ is used and was sent before the verifier chose which one should be verified. We are going to considerably benefit from this idea in our own construction.

It is interesting to notice that reducing the soundness error does not always yield to better efficiency, as was point out by Hu *et al.* in [62], who noticed that the approach from [38] is less efficient when $q = 3$ or $q = 4$, and showed that for these two particular cases the 3-move and $2/3$ soundness error protocol is superior.

This last proposal was soon ported to the lattice setting by Cayrel *et al.* in [36], combining the ideas from both [38] and [68]. That is, even if working with lattice problems as the ISIS they still relay on fixed Hamming weight binary vectors.

The next significant improvement was due to Ling *et al.* in [76], showing how the original Stern protocol could be run several times in parallel to prove that a solution has small infinity norm (and not only small Hamming weight). This, for the first time, allowed proving knowledge of a general solution of a lattice problem, in this case the ISIS problem. We recall here that previous lattice proposals have been restricted to binary secrets or low-hamming weight secrets, and therefore needed to rely on the witness indistinguishability and the existence of other solutions to get a solution to the SIS problem. The main idea of [76] is to decompose the secret in binary (with sign), $\boldsymbol{e} = \sum_j 2^j \boldsymbol{e}_j \in \mathbb{Z}_q^m$, and run the protocol in parallel for each $\boldsymbol{e}_j \in \{-1, 0, 1\}^m$. An ad-hoc basis could be used, but we prefer to keep notation simple and decompose the elements in binary assuming that the bound is a power of 2.

In the original protocol the secret can be randomized with a permutation since it has fixed weight. Here, each $\boldsymbol{e}_j$ has only $-1, 0, 1$, but the amount of each digit reveals information about the secret. Therefore, the vectors have to be extended tripling its size, adding blocks of -1's, 0's and 1's at the end, so that they have the same number of each digit, and only then are permuted. With one of the challenges the permutation is removed and only the first third is considered, but the secret is masked with a masking $\boldsymbol{x}$. In other challenges the masking element is removed, but the secret is extended and permuted and therefore reveals no information. As an application they present an Identity-based identification scheme.

We also use this idea for our proposal, further optimizing it to avoid using -1 as part of the decomposition, reducing the expansion factor from 3 to 2.

Many proposals have been made applying these techniques to instantiate different primitives, as the Identity Based identification schemes of El Yousfi *et al.* in [49], Yang *et al.* in [125] and Song *et al.* in [113], or the Batch Identification scheme by Silva *et al.* [110].

The signature scheme that arises from transforming the identification protocol into a non-interactive proof of the secret key associated with a public key can also be naturally adapted to build threshold ring signatures, where t out of N signers have to participate without revealing the specific t subset of participants, again hiding the identity of the participating subset by means of a permutation of a vector of size N and Hamming weight t (using a strategy similar to the anonymous identification of [68]). This approach is used by Aguilar-Melchor *et al.* in [2], Cayrel *et al.* in [37, 34], Bettaieb and Schrek in [23] and Branco and Mateus in [31].

Even more exotic signature schemes have been built using Stern-based protocols, as the undeniable signature scheme by Aguilar-Melchor *et al.* in [1], the blind signature [25] by Blazy *et al.*, the group signature [6] by Alamélou *et al.* or the designated verifier signature [61] by Hongbin and Yan.

We end this section revisiting with more detail the commitment schemes with Stern-based ZKPoKs for linear and multiplicative relations among committed elements. The first proposal in this direction was [64], due to Jain *et al.*

The authors of [64] built a commitment scheme from the *Learning Parity with Noise* (LPN) problem. This problem states that given $\boldsymbol{A} \in \mathbb{F}_2^{k \times l}$ a binary matrix, it is hard to distinguish $\boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{e}$ from $\boldsymbol{u}$, where $\boldsymbol{x} \in \mathbb{F}_2^l$ and $\boldsymbol{u} \in \mathbb{F}_2^k$ are uniformly random binary vectors and $\boldsymbol{e} \in \mathbb{F}_2^k$ is a low-weight binary vector. It can be seen as a binary version of the decisional LWE problem. In this paper they use a version where the error distribution only outputs fixed weight errors, and call it xLPN. Security is preserved as LPN can be reduced to xLPN.

Then, the commitment to a message $\boldsymbol{m} \in \mathbb{F}_2^n$ is $\boldsymbol{A}(\boldsymbol{r}||\boldsymbol{m}) \oplus \boldsymbol{e}$, where $\boldsymbol{r} \in \mathbb{F}_2^{l-n}$ is a random binary vector and $\boldsymbol{e} \in \mathbb{F}_2^k$ is again a random low-weight error. The most relevant feature of this commitment scheme is that it allows the committer to prove that they know a valid opening of a commitment, or that the hidden messages of different commitments satisfy any bitwise relation. To do this kind of proofs they present a dual version of the Stern protocol from [116], this time with the proper masking elements to achieve the zero-knowledge property that was not obtained in the dual version of [120].

Linear relations are proven by running in parallel three executions of the knowledge of opening protocol modifying the elements masking the messages so that they also satisfy the same linear relation. This simple strategy is going to be used too by [124, 19, 17] and us.

In order to prove any bitwise relation, $\boldsymbol{m}_3 = \boldsymbol{m}_1 \circ \boldsymbol{m}_2$, they first extend the messages to a size four times larger. For each index j in messages $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ we have a pair of bits $(\boldsymbol{m}_1[j], \boldsymbol{m}_2[j])$. They append at the end three additional pairs flipping the first, the second or both bits $\left(\overline{\boldsymbol{m}_1[j]}, \boldsymbol{m}_2[j]\right)$, $\left(\boldsymbol{m}_1[j], \overline{\boldsymbol{m}_2[j]}\right)$, $\left(\overline{\boldsymbol{m}_1[j]}, \overline{\boldsymbol{m}_2[j]}\right)$. We call $\widetilde{\boldsymbol{m}}_1, \widetilde{\boldsymbol{m}}_2$ to this extended versions and define $\widetilde{\boldsymbol{m}}_3 = \widetilde{\boldsymbol{m}}_1 \circ \widetilde{\boldsymbol{m}}_2$. Notice that since each possible pair of bits appears the same amount of times once we permute these messages the result does not reveal anything about the original ones. Following the same strategy as the original Stern protocol some answers reveal the permutation but keep the messages masked, while some others prove $\pi(\widetilde{\boldsymbol{m}}_3) = \pi(\widetilde{\boldsymbol{m}}_1) \circ \pi(\widetilde{\boldsymbol{m}}_2)$, and only a combination of all the three answers gives us soundness.

Notice the strategy is always the same, we expand the secrets so that we are able to completely randomize them in different ways (permuting or adding masking elements) that still preserve some properties, allowing us to remove one of the randomization elements to reveal that one property holds while the other kind of randomization hides everything else.

The previous scheme only allowed commitments to binary messages, but its ideas were soon adapted to the lattice setting by Xie *et al.* in [124]. This article transforms the commitment scheme of [64] into a lattice proposal using a dual version of the techniques from [76] (presenting new ideas since it is not a trivial translation).

Observe the RLWE problem is the right choice for this kind of constructions, even if the proofs are slightly more cumbersome, as it admits any message $m \in R_q$ as a secret, treating the small error $e \leftarrow_R \chi$ separately. This is not the case with the ISIS problem, where the only secret is restricted to have small norm, preventing us from computing arbitrary operations with the messages preserving this condition.

The most difficult part is proving multiplicative relations. Authors first present a simple inefficient version directly following the ideas of [64]. In order to be able to randomize the secrets they extend them with a q^2 overhead so that each pair $(a, b) \in \mathbb{Z}_q \times \mathbb{Z}_q$ appears the same number of times.

They can reduce the overhead to 4 first decomposing the secrets in binary and then using the same idea as before. However, this requires repeating the proof $\log(q)^2$ times, as we have decomposed each message in $\log(q)$ binary messages, and we have to multiply them. The binary decomposition is never disclosed, because only a commitment to each binary message has to be published, so a linear relation has to be used to show that these commitments to binary vectors are indeed commitments to a binary decomposition of the original committed message.

Binding property is obtained as two valid openings would imply the existence of a lattice vector shorter than the shortest lattice vector.

Notice here that a binary decomposition was feasible when we were talking about the error term since it is bounded and needs only a small overhead. A binary decomposition for products of unbounded polynomials requires this $\log(q)^2$ overhead. Our greatest contribution is going to be a newly designed proof that allows us to avoid such overhead.

This prolific area has recently been mostly replaced by Fiat-Shamir with aborts proposals, but we however think that it is still interesting to see how far can we optimize the original Stern's idea. In order to make faithful comparisons it is also important to be committed to strict formal proofs of security that can be quantitatively used to securely instantiate the lattice-based schemes belonging to this family, so that the current performance of this alternative can be properly evaluated.

3.3 Lattice-Based Commitment Scheme

Now we can define our proposal for a lattice-based commitment scheme. Any message $m \in R_q$ can be encoded as the coordinates of a point in an ideal lattice defined by $\boldsymbol{a} \in R_q^k$. To hide this lattice point $\boldsymbol{a}m \in \mathcal{L}(\boldsymbol{a})$ we add a RLWE sample from another lattice $\boldsymbol{b}r + \boldsymbol{e}$, where $\boldsymbol{b} \in R_q^k$ defines this other lattice $\mathcal{L}(\boldsymbol{b})$, the randomness $r \leftarrow_{\text{R}} R_q$ is chosen uniformly at random and the error term $\boldsymbol{e} \leftarrow_{\text{R}} \chi^k$ is chosen from the appropriate bounded discrete Gaussian distribution.

This structure $\boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ is used by Benhamouda *et al.* in [19], and it is very similar to the one proposed by Xie *et al.* in [124]. We however define a different set of conditions designed to guarantee security and be able to latter design the companion ZKPoKs of valid openings, linear and multiplicative relations.

The degree of the polynomial $n = 2^\kappa$ is a power of two, usually choosing $\kappa = 9$ or $\kappa = 10$. The modulus q is a prime number such that $q \equiv 2d + 1 \mod 4d$ for another power of two d smaller than n (so that we can apply the efficient multiplication algorithm discussed in Chapter 2). Integer k would be the multiplicative overhead (the length of $\boldsymbol{a}$ as a vector of polynomials). And finally, as error distribution χ we would use a bounded discrete Gaussian distribution $D_{\sigma,B}$ of parameter σ and bound yet another power of two B. That is, the probability of each coefficient would be proportional to that of a Gaussian distribution of standard deviation σ conditioned to be in $[-B, B)$, as defined in Section 1.3.1. Recall we abuse notation and just write $\|\boldsymbol{e}\|_\infty \leq B$.

Remember a commitment scheme is defined by a tuple of three algorithms Gen, Com and Ver, as defined in Definition 1.6. While the commitment algorithm Com we present in this chapter is formally the same as the one that was introduced in [19] the verification Ver algorithm is much simpler and satisfies the standard definitions, as our proofs of openings and relations do not require any relaxation (the set of valid openings is exactly the set of openings obtained following the commitment algorithm, which was not the case in [19]). Even if the commitment key $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_{\text{R}} (R_q^k)^2$ is equivalent and follows again a uniform distribution in $(R_q^k)^2$ the public parameters $pp := (n, q, d, k, \sigma, B)$ are defined in a different manner and have to satisfy less strict properties, so the key generation algorithm Gen is also different. For that reason the following proposal shares structure with [19], but is a different construction as a commitment scheme.

3.3.1 The Commitment Scheme

The following is a secure lattice-based commitment scheme under the assumption that the decisional-RLWE is hard if the parameters are properly chosen. We define it here and prove its security in Section 3.3.2.

- Gen: the generator algorithm takes a security parameter 1^λ and outputs public parameters $pp := (n, q, d, k, \sigma, B)$, a public key $pk := (\boldsymbol{a}, \boldsymbol{b}) \in (R_q^k)^2$, where $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ and the noise distribution χ is defined as $D^n_{\sigma,B}$.

  $(pk := (\boldsymbol{a}, \boldsymbol{b}); pp := (n, q, d, k, \sigma, B)) \leftarrow_{\text{R}} \mathsf{Gen}(1^\lambda)$

- Com: the commitment algorithm takes as input public parameters $pp := (n, q, d, k, \sigma, B)$, a message $m \in R_q$ and a public key $pk := (\boldsymbol{a}, \boldsymbol{b}) \in (R_q^k)^2$, and produces a commitment $\boldsymbol{c} := \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ and an opening $o := (r, \boldsymbol{e})$, where $r \leftarrow_{\text{R}} R_q$ and $\boldsymbol{e} \leftarrow_{\text{R}} \chi^k$.

  $(\boldsymbol{c} := \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}, o := (r, \boldsymbol{e})) \leftarrow_{\text{R}} \mathsf{Com}(m; pk := (\boldsymbol{a}, \boldsymbol{b}), pp)$

- Ver: the verification algorithm takes as input public parameters $pp := (n, q, d, k, \sigma, B)$, a commitment $\boldsymbol{c} \in R_q^k$, a message $m \in R_q$, an opening $o := (r, \boldsymbol{e})$ with $r \in R_q$ and $\boldsymbol{e} \in R_q^k$, and a public key $pk := (\boldsymbol{a}, \boldsymbol{b}) \in (R_q^k)^2$, and accepts if $\boldsymbol{c} \overset{?}{=} \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ and $\|\boldsymbol{e}\|_\infty \overset{?}{\leq} B$, or rejects otherwise.

  $\mathsf{Ver} : \{(\boldsymbol{c}, m, o; pk, pp)\} \to \{\text{accept}, \text{reject}\}$

3.3.2 Security Proofs of the Commitment Scheme

Now we can see what conditions are sufficient for the commitment we have just described in Section 3.3.1 to be secure.

Proposition 3.1 (Security of the commitment scheme)**.** *Provided that Gen outputs public parameters $pp := (n, q, d, k, \sigma, B)$ satisfying the following conditions the previously defined commitment scheme is secure, i.e., satisfies the standard notions of correctness, binding, and hiding.*

- $$q \equiv 2d + 1 \pmod{4d}, \tag{3.1}$$
- $$k \geq \frac{\lambda + 2n\log(q)}{n(\log(q)/d - \log(4B - 1))}, \tag{3.2}$$
- $$d < \frac{\log(q)}{\log(4B - 1)}, \tag{3.3}$$
- $$\mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,D^n_\sigma}) \geq \lambda, \tag{3.4}$$
- $$\frac{(B-1)^2}{2\sigma^2}\log(e) - \log(n \cdot k) \geq \lambda + 1. \tag{3.5}$$

Proof. We can check that all standard properties are verified, the commitment has correctness, it is perfectly binding except with negligible probability in the generation of the public key, and it is computationally hiding under the assumption that the decisional-RLWE is a hard problem.

- **Correctness**: if everybody is honest the verifier accepts the opening.

  It immediately follows by the definitions of Com and Ver. The structure validated by the verifier (a lattice point with the message as coordinates $\boldsymbol{a}m$ plus a RLWE sample $\boldsymbol{b}r + \boldsymbol{e}$) is exactly the same used to build the commitment in the first place, and the additional condition on the norm of the noise term $\boldsymbol{e}$ is ensured by the fact that we are already using as error distribution a truncated discrete Gaussian so that this condition is always satisfied.

  Notice that, as we are going to see when discussing the hiding property, the σ parameter is going to be chosen so that the probability of the tails from D_σ^n is negligible in the security parameter λ, so one could alternatively use non-truncated discrete Gaussians at the expense of having correctness except with negligible probability. We prefer to define the scheme with perfect correctness and deal with the bounded tails when studying the hiding property because it makes the analysis slightly simpler.

- **Binding**: a commitment can only be correctly opened to one message.

  We can guarantee it similarly than [19], using a counting argument to ensure that no two valid openings exist for a single commitment except with a very small probability in the sampling of the commitment key. Provided that our commitment definition does not need any kind of relaxation we are able to provide a simpler proof with tighter bounds. The argument is also generalized to the fact that $x^n + 1$ could factorize into d irreducible polynomials, for d any power of two smaller or equal than n, instead of only considering the case $d = 2$. We also directly bound the undesired probability by a specific $2^{-\lambda}$, rather than proving that it is asymptotically negligible in n, which helps the subsequent work of finding suitable parameters for a particular target security level.

  The binding property proof of [19] relies on the algebraic structure of $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. This structure heavily depends on the modulus q, as it has an impact on the factorization of $x^n + 1$, which is irreducible over the integers but splits into two irreducible polynomials of degree $n/2$ when $q \equiv 3 \pmod 8$ [115].

  However, as we want to take advantage of the efficient multiplication algorithm described in Chapter 2 we have to use a different condition on the modulus.

We have seen that

$$q \equiv 2d + 1 \pmod{4d} \qquad \text{(eq. (3.1))}$$

implies via Theorem 2.32 that $x^n + 1$ splits into d polynomials $\prod_{j=0}^{d-1}\left(x^{n/d} - \alpha_j\right)$ where each $x^{n/d} - \alpha_j$ is irreducible modulo q.

We can similarly use this alternative attribute to prove the binding property for our construction. Furthermore, we want to prove that, with overwhelming probability over the choice of $\boldsymbol{a}, \boldsymbol{b} \in R_q^k$, we have that

$$\left.\begin{array}{l}\text{accept} \leftarrow \mathsf{Ver}(\boldsymbol{c}, m', o'; pk, pp)\\ \text{accept} \leftarrow \mathsf{Ver}(\boldsymbol{c}, m'', o''; pk, pp)\end{array}\right\} \implies m' = m''.$$

Two accepted openings to the same commitment would be

$$\boldsymbol{c} = \boldsymbol{a}m' + \boldsymbol{b}r' + \boldsymbol{e}',$$
$$\boldsymbol{c} = \boldsymbol{a}m'' + \boldsymbol{b}r'' + \boldsymbol{e}''.$$

We have that $\boldsymbol{a}(m' - m'') + \boldsymbol{b}(r' - r'') + (\boldsymbol{e}' - \boldsymbol{e}'') = 0$, and we can rename the differences and write $\boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$, where $m \neq 0$ if $m' \neq m''$ and the elements from $\boldsymbol{e}$ belong to the interval $[-2B + 1, 2B - 1]$ if $\|\boldsymbol{e}'\|_\infty, \|\boldsymbol{e}''\|_\infty \leq B$. However, we are going to see that, with overwhelming probability over the choice of $\boldsymbol{a}$ and $\boldsymbol{b}$, and provided that $q \equiv 2d + 1 \pmod{4d}$, d is small enough and k is large enough, there are no $m, r \in R_q$ and $\boldsymbol{e} \in R_q^k$ such that $\boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e} = 0$ holds, $\boldsymbol{e}$ is small and $m \neq 0$.

For fixed m, r and $\boldsymbol{e}$ we count the proportion of pairs $(\boldsymbol{a}, \boldsymbol{b})$ for which the equality holds. Then, in order to estimate the overall probability of choosing a pair $(\boldsymbol{a}, \boldsymbol{b})$ such that there exists a solution, we use a union bound adding up all previous probabilities for any possible triplet $(m, r, \boldsymbol{e})$. We finally see that it is negligible if parameters are carefully selected.

As we said, if we consider fixed m, r and $\boldsymbol{e}$, for each $\boldsymbol{b}$ we have $\boldsymbol{a}m = -\boldsymbol{b}r - \boldsymbol{e}$. In each of the k components $a_i m = -b_i r - e_i$. $q \equiv 2d + 1 \pmod{4d}$ implies that $x^n + 1$ splits into d irreducible polynomials $p_1(x), p_2(x), \ldots, p_d(x)$ of degree n/d. We know that $m \not\equiv 0 \mod x^n + 1$, therefore, $m \not\equiv 0 \mod p_j(x)$ for some $j \in \{1, \ldots, d\}$.

For this case we know that choosing different a_i we have that $a_i m$ takes at least $q^{n/d}$ different values $\pmod{p_j(x)}$. There are $q^{n/d}$ equivalence classes $\pmod{p_j(x)}$ and only one of them is $-b_i r - e_i \pmod{p_j(x)}$, therefore at most

$1/q^{n/d}$ of the possible a_i hold the equation. As this is independently true for each i, we have that the probability of $(\boldsymbol{a}, \boldsymbol{b})$ to fit the equation for these particular m, r and $\boldsymbol{e}$ is at most $1/q^{nk/d}$.

If we want to consider the possibility that there exists a solution, we can bound the total probability with a union bound. There are less than q^n possible m, q^n possible r and $(4B-1)^{nk}$ possible $\boldsymbol{e}$. Therefore, if $(\boldsymbol{a}, \boldsymbol{b}) \leftarrow_{\mathrm{R}} (R_q^k)^2$ and we want to impose that the commitment is binding except with a probability smaller than $2^{-\lambda}$ we have

$$\Pr_{(\boldsymbol{a},\boldsymbol{b})}\left[\exists m, r, \boldsymbol{e} \;\middle|\; \begin{array}{c} \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e} = 0 \\ \boldsymbol{e} \in [-2B+1, 2B-1]^k \\ m \neq 0 \end{array}\right] \leq \frac{q^{2n}(4B-1)^{nk}}{q^{nk/d}} \leq 2^{-\lambda}.$$

Taking logarithms, the condition that has to be satisfied is

$$n(2\log(q) + k(\log(4B-1) - \log(q)/d)) \leq -\lambda.$$

If we ensure $(\log(4B-1) - \log(q)/d)$ is negative then we only need to choose

$$k \geq \frac{\lambda + 2n\log(q)}{n(\log(q)/d - \log(4B-1))}. \tag{eq. (3.2)}$$

Notice here the first condition just imposes the maximum value of d we can still use, preserving the binding property,

$$d < \frac{\log(q)}{\log(4B-1)}. \tag{eq. (3.3)}$$

There is an important trade-off in the result of Theorem 2.32, the more factors the faster the multiplication of ring elements can be implemented (the optimum is achieved when $x^n + 1$ splits into linear factors, and we can apply FFT multiplication), but achieving the binding property becomes harder. Considering that, we might end up with a larger commitment size or, if d is too large, we might end up not being able to find a secure set of parameters at all.

The smaller the error bound B gets the more options we have for this trade-off. As larger bounds also increase the size of the proof, our goal would be to find the smaller bound B that still ensures the hiding property.

- **Hiding**: a well constructed commitment $\boldsymbol{c}$ does not leak any relevant information about the message m.

It is computationally hiding as $\boldsymbol{b}r + \boldsymbol{e}$ are k RLWE samples, indistinguishable from independent uniformly random polynomials under the decisional $\mathrm{RLWE}_{n\,q\,k\,D^n_{\sigma\,B}}$ assumption. Any adversary able to break the hiding property would then also be able to solve the $\mathrm{RLWE}_{n\,q\,k\,D^n_{\sigma\,B}}$ problem (as seen in Proposition 1.49). Notice that we choose parameters so that the probability that $\boldsymbol{e} \leftarrow_{\mathrm{R}} D^{nk}_{\sigma}$ has $\|\boldsymbol{e}\|_\infty > B$ is negligible and then original and conditioned probability distributions are statistically indistinguishable.

First of all we have to check that the underlying computational problem is hard enough. We denote by $\mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,\chi})$ the number of security bits, i.e. $2^{\mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,\chi})}$ would be a lower bound on the number of operations required to solve the problem. One can then study the asymptotic behavior, as we know the currently most efficient algorithms for the usual instantiations of the RLWE problem are still exponential, and just ensure that q and χ are chosen so that the usual reductions can be applied to get $\mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,\chi}) \in \Omega(\lambda)$.

For practical purposes the asymptotics are not enough because we do care about the constant factors that might determine if the scheme can or cannot be broken in practice. For that reason we directly estimate bitsec using Albrecht *et al.* Lattice Estimator* [8].

Let us take a moment to discuss how the noise terms are treated. On the one hand, regarding the hiding property, we are concerned with the indistinguishability of RLWE samples from uniform polynomials and, as we have seen in Section 1.3.7, we mainly have reductions from the decisional-RLWE problem to presumably hard lattice problems when the noise distribution χ is a Gaussian, and discrete Gaussians are the most well-studied noise distributions. On the other hand, regarding both correctness and binding, we can see that is not enough. The verification algorithm usually boils down to check that the commitment was correctly computed using the revealed randomness r and $\boldsymbol{e}$. We then found two issues that arise from the noisy nature of the lattice problems. A *correctly computed* commitment would imply that the error term was sampled from χ^k, but any vector of polynomials $\boldsymbol{v} \in R_q^k$ could be, even if with a very small probability, a sample from χ^k. Nevertheless, the fact that $\boldsymbol{e}$ was small played an essential role when we proved that the equation $\boldsymbol{c} = \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ had a unique solution for m, implying the binding property (without this restriction finding multiple openings would be trivial just isolating $\boldsymbol{e}$). The only possibility we have to check it was correctly computed is to verify some condition on $\boldsymbol{e}$ that is fulfilled with overwhelming probability when $\boldsymbol{e}$ does

* Commit `f9f4b3c69d5be6df2c16243e8b1faa80703f020c` from `github.com/malb/lattice-estimator`

come from a discrete Gaussian and, at the same time, ensures a unique solution exists so that the binding property holds. This checkable property is just a bound on the infinity norm $\|\boldsymbol{e}\|_\infty \leq B$, as we have seen.

What we have left is to ensure that the decisional-RLWE problem remains hard even if we use a slightly different noise distribution because we condition the infinity norms to be bounded by B. As we have seen in Proposition 1.53 indistinguishability is almost preserved if we substitute one distribution, just increasing the bound on the advantage by the SD. We have also seen in Proposition 1.55 that the SD when we condition to an event is the probability of that event not happening, and how to bound probabilities and distances between truncated or not multivariate discrete Gaussians in Propositions 1.63 and 1.64.

We can particularize the definition of the advantage of an adversary against the hiding property, the ability they would have distinguishing two commitments to two different messages chosen by themself. More formally

$$\underset{\mathsf{hid}}{Adv}(\mathcal{A}) = \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} ((\boldsymbol{a},\boldsymbol{b}); pp := (n,q,d,k,\sigma,B)) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1((\boldsymbol{a},\boldsymbol{b}), pp, 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ r \leftarrow_{\mathrm{R}} R_q,\ \boldsymbol{e} \leftarrow_{\mathrm{R}} D^{nk}_{\sigma,B} \\ \boldsymbol{c} = \boldsymbol{a} m_b + \boldsymbol{b} r + \boldsymbol{e} \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(\boldsymbol{c}, (\boldsymbol{a},\boldsymbol{b}), pp, aux, 1^\lambda) \end{array} \right] - 1/2 \right|$$

and we can apply Corollary 1.57 to relate it to the probability of the truncated Gaussian tails and the RLWE problem with a standard non-truncated discrete Gaussian distribution

$$\leq \Pr\left[\|\boldsymbol{e}\|_\infty > B \;\middle|\; \boldsymbol{e} \leftarrow_{\mathrm{R}} D^{nk}_{\sigma} \right]$$
$$+ \left| \Pr\left[b_{\mathcal{A}} = b \;\middle|\; \begin{array}{c} ((\boldsymbol{a},\boldsymbol{b}); pp := (n,q,d,k,\sigma,B)) \leftarrow_{\mathrm{R}} \mathsf{Gen}(1^\lambda) \\ (m_0, m_1, aux) \leftarrow_{\mathrm{R}} \mathcal{A}_1((\boldsymbol{a},\boldsymbol{b}), pp, 1^\lambda), \\ b \leftarrow_{\mathrm{R}} \{0,1\}, \\ r \leftarrow_{\mathrm{R}} R_q,\ \boldsymbol{e} \leftarrow_{\mathrm{R}} D^{nk}_{\sigma} \\ \boldsymbol{c} = \boldsymbol{a} m_b + \boldsymbol{b} r + \boldsymbol{e} \\ b_{\mathcal{A}} \leftarrow_{\mathrm{R}} \mathcal{A}_2(\boldsymbol{c}, (\boldsymbol{a},\boldsymbol{b}), pp, aux, 1^\lambda) \end{array} \right] - 1/2 \right|$$

which we know by Proposition 1.49 that is equivalent to the advantage of a similarly powerful (performs the same plus a constant number of operations) adversary $\mathcal{B}$ against the decisional $\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}$ problem

$$\leq \Pr\left[\|\boldsymbol{e}\|_\infty > B \mid \boldsymbol{e} \leftarrow_{\mathrm{R}} D_\sigma^{nk}\right] + \max_{\mathcal{B}}\left\{Adv_{\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}}(\mathcal{B})\right\}.$$

Observe we cannot apply here the usual strategy of bounding each of the two terms by some $\epsilon/2$ so that the total advantage is bounded by ϵ, because we cannot really be such precise for the second term. The assumption is computational, we can either prove only asymptotic claims (the advantage $Adv_{\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}}$ for polynomial adversaries is negligible in λ for sufficiently large parameters, but the actual constants depend on the polynomial that bounds the running time of the adversary) or make it so solving it with overwhelming probability takes 2^λ operations and expect that there is no strange trade-off between computations and success probability, and therefore any algorithm with a running time significantly lower than 2^λ would only achieve a sufficiently small success probability. The first part is simpler, as it is inherently a probability condition, and can be directly addressed to ensure it is below a certain threshold $2^{-\lambda}$. This approach is not entirely satisfactory, but it is the best we can do to mix computational assumptions with statistical statements.

We follow the second procedure ensuring $\max_{\mathcal{B}}\left\{Adv_{\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}}(\mathcal{B})\right\}$ is insignificant by imposing

$$\mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}) \geq \lambda, \qquad \text{(eq. (3.4))}$$

and using the auxiliary functions that were discussed in Section 1.4.4 to find sufficient conditions for bounding the tail probability

$$\Pr\left[\ \|\boldsymbol{e}\|_\infty \leq B\ \middle|\ \boldsymbol{e} \leftarrow_{\mathrm{R}} D_\sigma^{nk}\right] \geq 1 - 2^{-\lambda}. \qquad \text{(eq. (3.5'))}$$

We explicitly defined vecBoundedPrFromBoundedPr in Proposition 1.64 so that if we find a such that $\Pr\left[e \notin [-B, B) \mid e \leftarrow_{\mathrm{R}} D_\sigma\right] \leq 2^{-a}$ then

$$-\log\left(\Pr\left[\ \|\boldsymbol{e}\|_\infty > B\ \middle|\ \boldsymbol{e} \leftarrow_{\mathrm{R}} D_\sigma^{nk}\right]\right) \geq \mathsf{vecBoundedPrFromBoundedPr}\,(a, nk)\,,$$

therefore it would be sufficient to ensure that

$$\mathsf{vecBoundedPrFromBoundedPr}\,(a, n \cdot k) \geq \lambda.$$

We precisely defined boundedPr in Proposition 1.63 so that

$$\log\left(\Pr\left[\,|e| > B \,\middle|\, e \leftarrow_{\mathrm{R}} D_\sigma\right]\right) \leq -\mathsf{boundedPr}(\sigma, B).$$

Combining it with the previous inequality we find that it is sufficient to ask for

$$\mathsf{vecBoundedPrFromBoundedPr}\left(\mathsf{boundedPr}(\sigma, B-1), n \cdot k\right) \geq \lambda,$$

as it implies eq. (3.5′).

Recall that we use $B-1$ as the event we want to consider is given by the error terms belonging to $[-B, B)$, which is ensured if they belong to $[-(B-1), B-1]$. A more tight analysis of this probability would not significantly change the final parameters.

Expanding this last expression what we get is

$$\frac{(B-1)^2}{2\sigma^2}\log(e) - \log(n \cdot k) \geq \lambda + 1, \qquad \text{(eq. (3.5))}$$

that is what we asked in the first place. □

We have defined a commitment that is perfectly binding with overwhelming probability over the sampling of the public key. In the following section we are going to see that some steps of the proofs for the soundness properties, and some computations conducted by the verifier of the ZKPoKs that we are going to present, rely on the perfectly binding property of the commitment. We can safely assume that the involved public keys define a binding commitment because the contrary only happens with negligible probability, bellow the $2^{-\lambda}$ threshold we have established. Any adversary with a success probability significantly greater than the soundness error would still obtain a success probability sufficiently above that error if only public keys defining binding commitments are used, because we are conditioning to an event that happens except with a probability lower than $2^{-\lambda}$, and the success probability could only decrease by such amount (and recall the standard soundness definition does not request the success probability of the extractor to be exactly the same of the advantage of the adversary, it only has to be proportional).

In practice, we might even ensure that some condition implied by the binding property holds, discarding the pair of keys in the overwhelmingly unlikely event it did not hold. In that case a rigorous analysis would then need to reevaluate the hiding property, as we would again be considered a distribution slightly different from $\mathcal{R}_{s,\chi}$, because we would be substituting the distribution that samples the $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors of polynomials from the public key. Once again we could see that the success probability of any adversary could only be incremented at most by $2^{-\lambda}$. This distribution is only used once, at the time of the key generation, unlike the error sampling from a truncated Gaussian that is used every time a commitment is computed. In any case, if one wants to meticulously bound all terms that come from statistical discussions by $2^{-\lambda}$, it would always be possible to rewrite the conditions ensuring that the commitment is binding except with a probability smaller than $2^{-(\lambda+1)}$ and analogously ensuring that the probability of the truncated tails of the discrete Gaussian is again below $2^{-(\lambda+1)}$, so that both bounds add up to $2^{-\lambda}$.

3.4 Interactive Zero-Knowledge Proofs of Knowledge

A commitment scheme is a fairly simple primitive that can be easily instantiated from even more basic primitives such as one-way functions. The potential of the proposed scheme arises from the fact that its usefulness as a building block for more complex protocols can be broadened with several ZKPoKs.

The opening ZKPoK that we are going to introduce in Section 3.4.1 allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the fact that they know a valid opening for a given commitment without revealing any information about the opening itself during the interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$.

This kind of proof is interesting because showing the commitments are well-formed is usually the first step of more intricate protocols that then manipulate the commitments or work with the committed values (for example using *commit-and-proof* strategies). It is also relevant because it serves as foundation for the other two ZKPoKs presented in Sections 3.4.2 and 3.4.3. In these two the prover is able to demonstrate knowledge of valid openings of three commitments to three messages $m_1, m_2, m_3 \in R_q$ satisfying $m_3 = \lambda_1 m_1 + \lambda_2 m_2$ for given $\lambda_1, \lambda_2 \in R_q$ or $m_3 = m_1 \cdot m_2$ respectively, again without revealing any additional information.

Notice that our commitment scheme is not homomorphic, that is, the sum of two commitments does not always form a valid commitment to the sum of the committed values because the norm of the added noise terms might exceed the bound. This is even more true for a linear combination of two commitments as the error term of the sum would also be a linear combination of the original error terms, losing any control on its bound. Nevertheless, one can always build a third commitment to this linear combination sampling fresh randomness and prove the linear relation holds using the corresponding ZKPoK. Even more delicate is the product case, but we can still manage to do the same, build a fresh commitment to the product and prove that the relation is satisfied, taking care of the crossed terms that would appear when multiplying the secret messages while masked.

3.4.1 Knowledge of a Valid Opening

We first propose an Interactive Honest-Verifier Zero-Knowledge Proof of Knowledge of a valid opening for the commitment presented before. The difficult part is to prove that the error term is small enough, for which we adapt Stern-based protocols to this particular RLWE based commitment. The main idea is to prove that coefficients of $\boldsymbol{e}$ are small by proving they have a binary decomposition with the desired limited length. While the SDP and the ISIS problem are very similar, in order to prove that the commitment has been constructed with a RLWE sample we need several auxiliary elements. What we obtain is a 5-move protocol with a soundness error of $(q+1)/2q$, really close to $1/2$ as q is usually a large prime.

Let $\boldsymbol{a} := (a_1, \dots, a_k), \boldsymbol{b} := (b_1, \dots, b_k) \in R_q^k$, a message $m \in R_q$, a random element r in R_q and noise $\boldsymbol{e} \in R_q^k$ a vector of polynomials with norm smaller than B. We want to prove knowledge of a valid opening for the commitment $\boldsymbol{c} := \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$.

Notice the first task we encounter characterizing the shortness of $\boldsymbol{e}$ by the length of its binary decomposition is to fix how to define this binary decomposition giving that we use $\left\{-\lfloor q/2 \rfloor, \dots, \lfloor q/2 \rfloor\right\}$ as representatives for $\mathbb{Z}_q$. To avoid any nuisance with negative numbers, a vector of polynomials $\boldsymbol{e} \in R_q^k$ with small coefficients, all of them in the interval $[-B, B)$, can be transformed into another vector of non-negative small coefficients by adding B to each of its coefficients. Let $\mathbf{B}$ be the vector of k polynomials from R_q^k that has all its coefficients equal to the bound B, so $\boldsymbol{e} + \boldsymbol{B}$ has all its coefficients positive.

Then we can decompose $\boldsymbol{e} + \boldsymbol{B}$ into $\log(B) + 1$ binary vectors of size nk, with k blocks of n bits, that represent the jth bits of the binary decomposition of the coefficients of each of the polynomials. We would like to be able to randomize them to hide all information but the fact that the norm of the original polynomials is bounded by B. To do so we first extend each vector doubling its size and adding one block of 0 followed by one block of 1 so that they have the same number of each. That way permuting vectors would completely hide any information (as it was done in the original Stern protocol with fixed Hamming weight vectors).

We denote by expand the function that takes as input a vector of polynomials $\boldsymbol{e}$ with all its coefficients in $[-B, B)$ and outputs $\log(B) + 1$ vectors $\boldsymbol{e}'_j$ following the previously described procedure.

We denote by $\mathfrak{B}_n \subset \{0, 1\}^{2n}$ the set of binary strings of length $2n$ with exactly n zeroes and n ones, so that correctly computed $\boldsymbol{e}'_j$ belong to $\mathfrak{B}_{nk}$.

This extension is an adaptation of the idea from Ling *et al.* in [76] to the dual ring setting (we shift the error to only have 0's and 1's, while their protocol also included −1's, this way we only have a factor two overhead instead of a factor three).

Then we also need to define an auxiliary function that allows us to go back from vectors to polynomials. Let ϕ be the function that takes as input a long vector of size $2nk$ from $\mathbb{Z}_q^{2nk}$, discards the second half, divides it in k blocks and outputs a vector of k polynomials in R_q^k that has as coefficients the elements from each of the blocks.

That way if $\{\boldsymbol{e}'_j\}_j \leftarrow \mathsf{expand}(\boldsymbol{e})$ then $\boldsymbol{e} = \phi(\sum_{j=0}^{\log(B)} 2^j \boldsymbol{e}'_j) - \boldsymbol{B}$.

With this notation we can define an interactive protocol to prove knowledge of a valid opening for commitment $\boldsymbol{c}$. The complex structure of the commitment scheme requires more subtle details than the original Stern proposal, but the underlying intuition is the same. We want to prove knowledge of some elements $m, r, \{\boldsymbol{e}'_j\}_j$, of some masking elements $\pi \in \mathfrak{S}_{2nk}$, $\mu, \rho \in R_q$, $\boldsymbol{f} \in \mathbb{Z}_q^{2nk}$ and of a vector of polynomials $\boldsymbol{y} \in R_q^k$ such that:

(a) $\pi_j(\boldsymbol{e}'_j) \in \mathfrak{B}_{nk}$,

(b) $\boldsymbol{y} = \boldsymbol{a}\mu + \boldsymbol{b}\rho + \boldsymbol{f}$,

(c) $\boldsymbol{y} + \boldsymbol{c} = \boldsymbol{a}(\mu + m) + \boldsymbol{b}(\rho + r) + (\boldsymbol{f} + \boldsymbol{e})$, where $\boldsymbol{e} = \phi(\sum_j 2^j \boldsymbol{e}'_j) - \boldsymbol{B}$.

All three properties imply knowledge of a valid opening for the commitment. However, provided that the masking elements are chosen uniformly at random, just from any two of them no information about the opening could be obtained, as the masking elements hide the opening elements and permutations π_j completely hide the information about each $\boldsymbol{e}'_j$ (in an analogous way as the original Stern protocol). In order to improve efficiency we can add one more round where we ask the verifier for an additional challenge $\alpha \in \mathbb{Z}_q$ and then prove only these two properties:

(a) $\pi_j(\boldsymbol{e}'_j) \in \mathfrak{B}_{nk}$,

(b′) $\boldsymbol{y} + \alpha\boldsymbol{c} = \boldsymbol{a}(\mu + \alpha m) + \boldsymbol{b}(\rho + \alpha r) + (\boldsymbol{f} + \alpha\boldsymbol{e})$, where $\boldsymbol{e} = \phi(\sum_j 2^j \boldsymbol{e}'_j) - \boldsymbol{B}$.

To prove these properties we follow the general *commit-and-choose* strategy where the prover starts committing to the relevant elements and then the verifier chooses which or how these commitments should be opened. As these elements were committed in the first round (using an auxiliary commitment scheme) before α was chosen we can ensure with high probability that property (b′) implies both properties (b) and (c). This is an adaptation of the idea used in [38] and allows us to reduce the soundness error from $2/3$ to almost $1/2$.

With this intuition in mind we can provide our Protocol 3.1 for proving knowledge of valid openings. Let $(\mathsf{aCom}, \mathsf{aVer})$ denote an auxiliary commitment scheme (one could of course use the same commitment scheme we are analyzing, but since we do not need any special property nor the existence of companion ZKPoKs we might as well use a simpler and more efficient construction).

The prover $\mathcal{P}$ chooses $\log(B)+1$ permutations $\pi_0, \dots, \pi_{\log(B)} \leftarrow_{\text{R}} \mathfrak{S}_{2nk}$, $\log(B)+1$ random vectors $\boldsymbol{f}_0, \dots, \boldsymbol{f}_{\log(B)} \leftarrow_{\text{R}} \mathbb{Z}_q^{2nk}$ and 2 random polynomials $\mu, \rho \leftarrow_{\text{R}} R_q$.

The polynomials μ, ρ will be used to mask m and r respectively, while the permutations π_j and the vectors $\boldsymbol{f}_j$ will be used to mask the expanded error terms $\{\boldsymbol{e}'_j\}_j \leftarrow \mathsf{expand}(\boldsymbol{e})$ (the previously introduced $\boldsymbol{f}$ would be $\phi(\sum_j 2^j \boldsymbol{f}_j)$). If the verifier chooses to remove the masking vectors $\boldsymbol{f}_j$ we could show that the shortness property (a) holds, while removing the permutations we would be able to show that the commitment had the proper structure (b′).

The prover $\mathcal{P}$ starts computing the following commitments,

$$\begin{aligned} c_1 &\leftarrow_{\text{R}} \mathsf{aCom}\big(\{\pi_j\}_j, \boldsymbol{a}\mu + \boldsymbol{b}\rho + \phi(\textstyle\sum_j 2^j \boldsymbol{f}_j)\big), \\ c_2 &\leftarrow_{\text{R}} \mathsf{aCom}\big(\{\pi_j(\boldsymbol{f}_j)\}_j, \{\pi_j(\boldsymbol{e}'_j)\}_j\big). \end{aligned}$$

The prover sends these commitments to the verifier. The verifier $\mathcal{V}$ chooses an integer $\alpha \in \mathbb{Z}_q$ and sends it to the prover as first challenge. Then the prover computes

$$\boldsymbol{g}_j := \pi_j(\boldsymbol{f}_j + \alpha \boldsymbol{e}'_j).$$

The prover sends $\{\boldsymbol{g}_j\}_j$ to the verifier. The verifier $\mathcal{V}$ continues choosing a random bit $b \leftarrow_{\text{R}} \{0,1\}$ and sends it to the prover. Based on this second challenge the prover opens the first or the second commitment.

Observe that we have called π to the first elements committed in c_1, but a malicious prover could claim that something different is the committed value when presenting an opening. Provided that the auxiliary commitment is secure, it would be discovered by the verifier when it fails the auxiliary commitment verification. As we cannot assume the prover would be honest and to point out that subtlety, as we said, we are going to denote by $\widetilde{\pi}$ the element that $\mathcal{P}$ discloses if the protocol says it should be π, as it cannot be assumed $\widetilde{\pi} = \pi$, and analogously for any other element.

Case $b = 0$.

– $\mathcal{P}$ reveals $\{\widetilde{\pi}_j := \pi_j\}_j, \widetilde{\boldsymbol{y}} := \boldsymbol{a}\mu + \boldsymbol{b}\rho + \phi(\sum_j 2^j \boldsymbol{f}_j), \widetilde{s} := \rho + \alpha r$ and the opening of c_1 to $(\{\widetilde{\pi}_j\}_j, \widetilde{\boldsymbol{y}})$.

– $\mathcal{V}$ checks c_1. They also compute $\boldsymbol{z} := \widetilde{\boldsymbol{y}} + \alpha(\boldsymbol{c} + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s} - \phi(\sum_j 2^j \widetilde{\pi}_j^{-1}(\boldsymbol{g}_j))$ and check that $\boldsymbol{z} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$, i.e., that there exists a $t \in R_q$ such that $\boldsymbol{z} = \boldsymbol{a}t$.

Case $b = 1$.

– $\mathcal{P}$ reveals $\{\widetilde{\boldsymbol{e}}'_j := \pi_j(\boldsymbol{e}'_j)\}_j$ and the opening of c_2 to $(\{\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j\}_j, \{\widetilde{\boldsymbol{e}}'_j\}_j)$.

– $\mathcal{V}$ checks c_2 and that each $\widetilde{\boldsymbol{e}}'_j$ belongs to $\mathfrak{B}_{nk}$.

$$\mathrm{ZKP}\left[\, m, r, \boldsymbol{e} \;\middle|\; \boldsymbol{c} = \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}, \quad \|\boldsymbol{e}\|_\infty \le B \,\right] \tag{3.6}$$

Protocol 3.1 Knowledge of a Valid Opening

```
$\mathcal{P}((\boldsymbol{a},\boldsymbol{b}),\boldsymbol{c}; m, r, \boldsymbol{e})$                                   $\mathcal{V}((\boldsymbol{a},\boldsymbol{b}),\boldsymbol{c})$
1:  $\boldsymbol{e}'_0, \ldots, \boldsymbol{e}'_{\log(B)} \leftarrow \mathsf{expand}(\boldsymbol{e})$
2:  $\pi_0, \ldots, \pi_{\log(B)} \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$
3:  $\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{\log(B)} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$
4:  $\mu, \rho \leftarrow_{\mathrm{R}} R_q$
5:  $(c_1, o_1) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_j\}_j, \boldsymbol{a}\mu + \boldsymbol{b}\rho + \phi(\sum_j 2^j \boldsymbol{f}_j))$
6:  $(c_2, o_2) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_j(\boldsymbol{f}_j)\}_j, \{\pi_j(\boldsymbol{e}'_j)\}_j)$
                                   $\xrightarrow{c_1, c_2}$
7:                                                                          $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_q$
                                   $\xleftarrow{\alpha}$
8:  for $j \in 0, \ldots, \log(B)$ do
9:      $\boldsymbol{g}_j := \pi_j(\boldsymbol{f}_j + \alpha \boldsymbol{e}'_j)$
                                   $\xrightarrow{\{\boldsymbol{g}_j\}_j}$
10:                                                                         $b \leftarrow_{\mathrm{R}} \{0,1\}$
                                   $\xleftarrow{b}$
11: if $b = 0$ then
12:     for $j \in 0, \ldots, \log(B)$ do
13:         $\widetilde{\pi}_j := \pi_j$
14:     $\widetilde{\boldsymbol{y}} := \boldsymbol{a}\mu + \boldsymbol{b}\rho + \phi(\sum_j 2^j \boldsymbol{f}_j)$
15:     $\widetilde{s} := \rho + \alpha r$
16:     $\widetilde{o} := o_1$
17:     $ans := (\{\widetilde{\pi}_j\}_j, \widetilde{\boldsymbol{y}}, \widetilde{s}, \widetilde{o})$
18: if $b = 1$ then
19:     for $j \in 0, \ldots, \log(B)$ do
20:         $\widetilde{\boldsymbol{e}}'_j := \pi_j(\boldsymbol{e}'_j)$
21:     $\widetilde{o} := o_2$
22:     $ans := (\{\widetilde{\boldsymbol{e}}'_j\}_j, \widetilde{o})$
                                   $\xrightarrow{ans}$
23:                                                                         if $b = 0$ then
24:                                                   $\overset{?}{\mathsf{aVer}}(c_1, (\{\widetilde{\pi}_j\}_j, \widetilde{\boldsymbol{y}}), \widetilde{o})$
25:                   $\boldsymbol{z} := \widetilde{\boldsymbol{y}} + \alpha(\boldsymbol{c} + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s} - \phi(\sum_j 2^j \widetilde{\pi}_j^{-1}(\boldsymbol{g}_j)) \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
26:                                                                         if $b = 1$ then
27:                                    $\overset{?}{\mathsf{aVer}}(c_2, (\{\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j\}_j, \{\widetilde{\boldsymbol{e}}'_j\}_j), \widetilde{o})$
28:                                                            for $j \in 0, \ldots, \log(B)$ do
29:                                                                      $\widetilde{\boldsymbol{e}}'_j \overset{?}{\in} \mathfrak{B}_{nk}$
```

Completeness

If $\mathcal{P}$ knows a valid witness and both the prover and the verifier correctly follow the protocol then the verifier always accepts at the end, as all relations hold by construction.

The commitment verifications come from the correctness of the auxiliary commitment scheme and the fact that the $\boldsymbol{g}_j$ were correctly computed using the previously committed $\boldsymbol{f}_j$ and $\boldsymbol{e}'_j$ (the fact that this $\boldsymbol{g}_j$ have been sent before the second challenge is received implies that the same elements have to be used to prove all properties). The vector $\boldsymbol{z}$ does belong to $\mathcal{L}(\boldsymbol{a})$ as it is equal to $\boldsymbol{a}(\mu + \alpha m)$ if the prover has followed the protocol (this is only possible because $\boldsymbol{e}$ is small and therefore $\phi(\sum_j 2^j \boldsymbol{e}'_j) - \boldsymbol{B} = \boldsymbol{e}$ and cancels out the error terms). Finally, $\boldsymbol{e}'_j \in \mathfrak{B}_{nk}$ if the original $\boldsymbol{e}$ had all its coefficients in $[-B, B)$ and the $\boldsymbol{e}'_j$ have been computed using expand.

Unlike Fiat-Shamir with aborts ZKPoKs here we have standard completeness as an honest prover will always succeed responding and there is no abort probability.

Soundness

Observe with each of the challenges $b = 1$ or $b = 0$ we check statements related to the conditions (a) and (b′).

If a (possibly malicious) prover $\mathcal{P}^*$ is able to provide accepted answers to δ rounds of interaction with an honest verifier $\mathcal{V}$ with probability $\left({(q+1)}/{2q}\right)^{\delta} + \epsilon$, were ϵ is non-negligible, then they are able to efficiently extract a witness with probability proportional to ϵ.

Recall the definition of soundness for an interactive protocol between a prover and a verifier $\langle \mathcal{P}, \mathcal{V} \rangle$ given in Definition 1.8. We want that for any $x \notin \mathfrak{R}$ the success probability of any malicious prover $\mathcal{P}^*$ should be small as

$$\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = \text{accept}\right] \le \epsilon(x).$$

That is the goal of our proofs, as we want to convince the verifier of the validity of the statements, but we can also ensure that we could extract a witness using the knowledge-soundness Definition 1.10, that can be seen to imply soundness.

In many occasions, soundness is then proved via showing Knowledge-Soundness because the proof satisfies yet another property, called special soundness, from which we need the multiple-moves version.

Definition 3.2 (($k_1, \ldots, k_\mu$)-Special Soundness as in [10]). A $(2\mu+1)$-move public-coin protocol is $(k_1, \ldots, k_\mu)$-special sound if there is an efficient algorithm that on input

$\prod_{j=1}^{\mu} k_j$ accepting transcripts

$$\left\{(x, a, \{c_{i_1,\dots,i_j}\}_{j=1}^{\mu}, \{b_{i_1,\dots,i_j}\}_{j=1}^{\mu})\right\}_{i_j \in \{1,\dots,k_j\}\ \forall j \in \{1,\dots,\mu\}},$$

such that $c_{i_1,\dots,i_{j-1},i_j} \neq c_{i_1,\dots,i_{j-1},i'_j}$ when $i_j \neq i'_j$, outputs a witness w such that $(x, w) \in \mathfrak{R}$.

Observe this definition is a fine-graded generalization of the usual k-special soundness for 3-move protocols where one just needs k transcripts (a, c_j, b_j) with $c_j \neq c_{j'}$. Nevertheless, when working with multiple-move protocols asking just for a number of transcripts does not fully capture the structure of the proof, as the possibilities branch out with each different challenge and the tree structure of the relevant conversations plays a relevant role if one wants to obtain tight proofs (see Figure 1 of [11] for a nice visualization of the tree structure).

Our proposal has been designed so that a single iteration of our opening protocol is $(2, 2)$-special sound, as we just need four transcripts with α and α' as the first challenge and both $b = 0$ and $b = 1$ as the second for each of them.

Proposition 3.3 ((2, 2)-Special Soundness of the Opening Protocol 3.1)**.** *The opening protocol described in Protocol 3.1 satisfies the definition of* $(2, 2)$*-Special Soundness.*

Proof. If we can find commitments c_1, c_2, two α, α' and $\boldsymbol{g}_j, \boldsymbol{g}'_j$ that induce accepted answers for both $b = 0$ and $b = 1$ then we can extract a valid witness.

Define $\Delta_\alpha = \alpha - \alpha' \neq 0$. On the one hand, the binding property of c_1, c_2 ensures that openings to $\widetilde{\pi}_j$, $\widetilde{\boldsymbol{y}}$ and $\widetilde{\boldsymbol{e}}'_j$ are fixed (for the sake of simplifying the proof let us assume here that the auxiliary commitment is perfectly binding, what happens otherwise is going to be later discussed).

$$\begin{aligned}
\boldsymbol{z} &= \widetilde{\boldsymbol{y}} + \alpha(\boldsymbol{c} + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s} - \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\boldsymbol{g}_j)) \\
\boldsymbol{z}' &= \widetilde{\boldsymbol{y}} + \alpha'(\boldsymbol{c} + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}' - \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\boldsymbol{g}'_j)) \\
\Delta_\alpha \boldsymbol{c} &= (\boldsymbol{z} - \boldsymbol{z}') + \boldsymbol{b}(\widetilde{s} - \widetilde{s}') + \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\boldsymbol{g}_j - \boldsymbol{g}'_j)) - \Delta_\alpha \boldsymbol{B} \\
\boldsymbol{c} &= \Delta_\alpha^{-1}(\boldsymbol{z} - \boldsymbol{z}') + \boldsymbol{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\Delta_\alpha^{-1}(\boldsymbol{g}_j - \boldsymbol{g}'_j))) - \boldsymbol{B}
\end{aligned}$$

On the other hand, the claimed commitment to $\pi_j(\boldsymbol{f}_j)$ is now opened to both $\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j$ and $\boldsymbol{g}'_j - \alpha'\widetilde{\boldsymbol{e}}'_j$, so the binding property of the auxiliary commitment scheme now ensures an equality.

$$\begin{aligned}
\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j &= \boldsymbol{g}'_j - \alpha'\widetilde{\boldsymbol{e}}'_j \\
\widetilde{\boldsymbol{e}}'_j &= \Delta_\alpha^{-1}(\boldsymbol{g}_j - \boldsymbol{g}'_j)
\end{aligned}$$

So we can substitute it in the previous expression.

$$\boldsymbol{c} = \Delta_\alpha^{-1}(\boldsymbol{z} - \boldsymbol{z}') + \boldsymbol{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\boldsymbol{e}}'_j)) - \boldsymbol{B}$$

We know $\Delta_\alpha^{-1}(\boldsymbol{z} - \boldsymbol{z}') \in \mathcal{L}(\boldsymbol{a})$ because $\boldsymbol{z}, \boldsymbol{z}' \in \mathcal{L}(\boldsymbol{a})$, but it is still left to detail how to check if a point $\boldsymbol{z}$ belongs to $\mathcal{L}(\boldsymbol{a})$, the ideal lattice defined by $\boldsymbol{a}$, and how to recover its coordinates (the polynomial $t \in R_q$ such that the vector of polynomials is $\boldsymbol{a}t$). In order to efficiently do so, we use one property of the commitment key that has not been mentioned yet, but that can be deduced from the binding property of the commitment scheme.

Lemma 3.4. If $(\boldsymbol{a}, \boldsymbol{b}) \in R_q^k \times R_q^k$ defines a commitment key for a perfectly binding commitment scheme in a ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ such that $x^n + 1$ splits in d irreducible $p_j(x)$ modulo q for $j \in \{1, \dots, d\}$, then there exist d indexes i_j such that the component $a_{i_j} \in R_q$ is invertible modulo $p_j(x)$.

Proof. This can be directly proved by contradiction. Assume it is not the case and for some $j \in \{1, \dots, d\}$ we have that $a_i \equiv 0$ modulo $p_j(x)$ for all $i \in \{1, \dots, k\}$.

Then we can define m_0 such that $m_0 \equiv 0 \mod p_{j'}(x)$ for all $j' \neq j$ and $m_0 \equiv 1 \mod p_j(x)$. By construction $m_0 \neq 0$ but $\boldsymbol{a}m_0 = 0$ and, therefore, we could open any commitment to any message m to a different message $m + m_0$, breaking the binding property of the commitment scheme. □

We explicitly check this condition (and abort if it is not satisfied) in the key generation protocol, but it will hold except with negligible probability if the used set of parameters is secure. We also output the specific indexes i_j and the inverses ${a_{i_j}}^{-1} \mod p_j$ as auxiliary parameters during the key generation protocol (even if omitted during the text, it is implicit that participants receiving the public key $(\boldsymbol{a}, \boldsymbol{b})$ also get these auxiliary elements precomputed).

With that property in mind now we can see that, provided a vector of polynomials $\boldsymbol{z} \in R_q^k$ if there exists a polynomial t such that $\boldsymbol{z} = \boldsymbol{a} \cdot t$ this t should verify that $z_{i_j} \equiv a_{i_j} t \mod p_j(x)$, and we could completely determine it from $t \equiv {a_{i_j}}^{-1} z_{i_j} \mod p_j(x)$.

Since we have already restricted the possible candidates for t to one single polynomial, we would just need to check that for every $i \in \{1, \dots, k\}$ and every $j \in \{1, \dots, d\}$ it holds that $z_i \equiv a_i {a_{i_j}}^{-1} z_{i_j} \mod p_j(x)$. Using again the invertibility of the a_{i_j}, to verify $\boldsymbol{z} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$ it is sufficient to just check $a_{i_j} z_i \overset{?}{\equiv} a_i z_{i_j} \mod p_j(x)$, without ever explicitly computing the t.

Only the extractor needs to use these auxiliary public parameters to compute the involved t and t' from $\boldsymbol{z}$ and $\boldsymbol{z}'$. Then it could finally write

$$\boldsymbol{c} = \boldsymbol{a}\Delta_\alpha^{-1}(t - t') + \boldsymbol{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\textstyle\sum_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\boldsymbol{e}}'_j)) - \boldsymbol{B}.$$

As these elements come from accepted answers we know that $\widetilde{\boldsymbol{e}}'_j \in \mathfrak{B}_{nk} \subset \{0, 1\}^{2nk}$ and therefore $\phi(\sum_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\boldsymbol{e}}'_j)) - \boldsymbol{B}$ has norm smaller than B.

Then $(\Delta_\alpha^{-1}(\widetilde{t} - \widetilde{t}'), \Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}'), \phi(\sum_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{e}_j')) - \boldsymbol{B})$ is a valid opening. □

Observe the definition is only rigorously satisfied if the auxiliary commitment is perfectly binding, as we have explicitly used that two openings to the same commitment must be equal. Nevertheless, it is still possible to take advantage of the property if we instead use a computationally binding commitment. In that case, if we have used rewindable access to a prover $\mathcal{P}^*$ to obtain the 4 accepting conversations with the requested structure with a non-negligible probability, the commitment openings would still be unique with overwhelming probability, and we could still use the previous strategy to extract a valid witness. Recall the definition does not ask the extractor success probability to be exactly the difference between the success probability of $\mathcal{P}^*$ and κ, as we are satisfied with it being decreased only in a polynomial amount (and a computationally binding commitment would only decrease it negligibly).

We have mentioned the possibility of using rewindable access to a prover to get the accepting conversations, but to see how $(2, 2)$-Special Soundness implies Knowledge-Soundness it remains to specify how to efficiently obtain such tree of conversations. A general tight efficient extractor was first defined in [10] and [11] proves (Theorem 3 in their paper) that a $(k_1, \ldots, k_\mu)$-special sound protocol is knowledge sound with knowledge error

$$\kappa = 1 - \prod_{j=1}^{\mu} \frac{|C_j| - k_j + 1}{|C_j|}. \tag{3.7}$$

That means a single repetition of the opening interactive protocol would be knowledge sound with knowledge error

$$\kappa = 1 - \frac{q - 2 + 1}{q} \cdot \frac{2 - 2 + 1}{2} = 1 - \frac{q - 1}{2q} = \frac{q + 1}{2q}.$$

It corresponds to the probability of correctly guessing at least one of the challenges, which is a usual property of this kind of protocols.

Of course such bound on the cheating probability is unreasonable for any meaningful practical case. The general strategy is to repeat the same protocol δ independent times, significantly reducing the bound on the cheating probability to something as low as we are comfortable with. It is however significantly more involved to extract the useful accepting conversations from δ parallel repetitions of a $(k_1, \ldots, k_\mu)$-special sound protocol (we specifically need to avoid exploring the whole possible execution tree as the complexity would then increase exponentially in the number of repetitions).

Fortunately, the really novel contribution of [11] (presented in their Theorem 4) is that the parallel repetition of a $(k_1, \ldots, k_\mu)$-special sound protocol δ times is knowledge sound with knowledge error κ^δ (when the δ parallel repetitions effectively reduce the knowledge error from κ to κ^δ they call it a strong parallel repetition result). Moreover, the extractor would only require a number of rewinds proportional to $\prod_{i=1}^{\mu} k_i$ (proportional to 4 in the opening protocol).

It is interesting to just mention how [11] achieves its results. Their main contribution is a novel extractor for a k-special sound Σ-protocol (that is, with just three moves), that can then be naturally generalized to multi-round protocols, parallel repetitions, or both at the same time. This new extractor can be analyzed using a more fine-graded measure of the quality of the prover than just its expected success probability, given by the minimum expected probability of the prover when some challenges have been removed from the challenge set. This new definition comes naturally from the fact that, to be able to extract a witness from a k-special sound protocol, we might try to sequentially obtain different accepting conversations with challenges different from the ones for which we have already obtained a valid conversation, therefore we are only interested in the success probability when the prover is challenged with a new challenge (removing the previous challenges from the challenge set), captured by this new notion.

It allows them to easily define the extractor in a recursive way, aborting if the first attempt fails and recursively extracting $(k-1)$ accepting conversations excluding the previous challenge for which we already have a valid conversation.

Their approach then fails and aborts in the first step with probability $1-\epsilon$, and only continues with probability ϵ (the success probability of the prover). This allows the extractor to have $\mathcal{O}(1/\epsilon)$ attempts to obtain the next $(k-1)$ accepting conversations while keeping the expected running time constant (it is proportional to $1/\epsilon$ with probability ϵ or constant otherwise). With that strategy, the success probability is proportional to $\epsilon - \kappa$, as required by the standard definition, while keeping constant the expected number of prover rewinds.

Furthermore, they also provide clever bounds for the success probability of the extractor applied to the parallel repetitions. Instead of ensuring that we can use one of the repetitions to extract a witness and focusing on it, or trying every $i \in \{1, \ldots, \delta\}$ but bounding its probability with the probability of a single i, they show the probability of being successful on at least one of the repetitions. This allows them to obtain an efficient extractor while obtaining a strong parallel repetition result, effectively reducing a knowledge-soundness error κ to κ^δ.

Using it we are able to ensure that the opening protocol satisfies Knowledge-Soundness and therefore soundness as introduced in Definitions 1.8 and 1.10.

Zero-Knowledge

In order to prove the protocol satisfies the honest verifier zero-knowledge property we are going to show the existence of a simulator $\mathcal{S}$ that, given a pair of challenges (α, b), outputs a conversation following the same distribution as the ones one would obtain from honest provers and verifiers.

This ensures that only the fact that the protocol was followed in order convinced the verifier that the prover knows the witness (as seen in the soundness property), but the conversation itself does not reveal any additional information about the secret because someone can indeed produce similar (computationally indistinguishable) transcripts using the simulator without any knowledge of the secret.

We introduce here notation $\widehat{a}$ to refer to the simulated element that plays the role of some element a so that we can then use both notations to prove that distributions involving $\widehat{a}$ are identically distributed to those involving the honestly computed a.

Case $b = 0$.

$$\widehat{t}, \widehat{s} \leftarrow_{\mathrm{R}} R_q, \quad \widehat{\boldsymbol{g}}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}, \quad \widehat{\pi}_j \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$$

$$c_1 \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\widehat{\pi}_j\}_j, \boldsymbol{a}\widehat{t} + \boldsymbol{b}\widehat{s} + \phi(\textstyle\sum_j 2^j \widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j)) - \alpha(\boldsymbol{c} + \boldsymbol{B}))$$

$\mathcal{S}$ reveals $\{\widehat{\boldsymbol{g}}_j\}_j, \{\widetilde{\pi}_j = \widehat{\pi}_j\}_j, \widetilde{\boldsymbol{y}} = \boldsymbol{a}\widehat{t} + \boldsymbol{b}\widehat{s} + \phi(\sum_j 2^j \widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j)) - \alpha \boldsymbol{c}, \widetilde{s} = \widehat{s}$.

Indistinguishable from a real conversation with the same $\pi_j = \widehat{\pi}_j$ and where $\mu = \widehat{t} - \alpha m$, $\rho = \widehat{s} - \alpha r$ and $\boldsymbol{f}_j = \widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j) - \alpha \boldsymbol{e}'_j$.

$$\begin{aligned}
\boldsymbol{g}_j &= \pi_j(\boldsymbol{f}_j + \alpha \boldsymbol{e}'_j) \\
&= \pi_j(\widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j)) + \pi_j(\alpha \boldsymbol{e}'_j - \alpha \boldsymbol{e}'_j) \\
&= \widehat{\boldsymbol{g}}_j
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{a}\mu + \boldsymbol{b}\rho + \phi(\textstyle\sum_j 2^j \boldsymbol{f}_j) &= \boldsymbol{a}(\widehat{t} - \alpha m) + \boldsymbol{b}(\widehat{s} - \alpha r) + \phi(\textstyle\sum_j 2^j (\widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j) - \alpha \boldsymbol{e}'_j)) \\
&= \boldsymbol{a}\widehat{t} + \boldsymbol{b}\widehat{s} + \phi(\textstyle\sum_j 2^j \widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j)) - \alpha(\boldsymbol{a}m + \boldsymbol{b}r + \phi(\textstyle\sum_j 2^j \boldsymbol{e}'_j)) \\
&= \boldsymbol{a}\widehat{t} + \boldsymbol{b}\widehat{s} + \phi(\textstyle\sum_j 2^j \widehat{\pi}_j^{-1}(\widehat{\boldsymbol{g}}_j)) - \alpha(\boldsymbol{c} + \boldsymbol{B})
\end{aligned}$$

Case $b = 1$.

$$\widehat{\boldsymbol{e}}'_j \leftarrow_{\mathrm{R}} \mathfrak{B}_{nk}, \quad \widehat{\boldsymbol{f}}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}, \quad \widehat{\pi}_j \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$$

$$c_2 \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\widehat{\pi}_j(\widehat{\boldsymbol{f}}_j)\}_j, \{\widehat{\pi}_j(\widehat{\boldsymbol{e}}'_j)\}_j)$$

$$\widehat{\boldsymbol{g}}_j = \widehat{\pi}_j(\widehat{\boldsymbol{f}}_j + \alpha \widehat{\boldsymbol{e}}'_j)$$

$\mathcal{S}$ reveals $\{\widehat{\boldsymbol{g}}_j\}_j$, $\{\widehat{\boldsymbol{e}}'_j = \widehat{\pi}_j(\widehat{\boldsymbol{e}}'_j)\}_j$. Equivalent to an honest conversation were π_j is such that $\pi_j(\boldsymbol{e}'_j) = \widehat{\pi}_j(\widehat{\boldsymbol{e}}'_j)$ and $\boldsymbol{f}_j = \pi_j^{-1}(\widehat{\pi}_j(\widehat{\boldsymbol{f}}_j))$.

$$\begin{aligned} \boldsymbol{g}_j &= \pi_j(\boldsymbol{f}_j + \alpha \boldsymbol{e}'_j) \\ &= \pi_j(\pi_j^{-1}(\widehat{\pi}_j(\widehat{\boldsymbol{f}}_j))) + \alpha\widehat{\pi}_j(\widehat{\boldsymbol{e}}'_j) \\ &= \widehat{\pi}_j(\widehat{\boldsymbol{f}}_j + \alpha\widehat{\boldsymbol{e}}'_j) \\ &= \widehat{\boldsymbol{g}}_j \end{aligned}$$

Notice that in both cases simulated conversations follow the same distribution as honest conversations. All omitted commitments can be computed as commitments to 0, which makes them indistinguishable from honestly computed commitments by the hiding property of the auxiliary commitment scheme. The elements $\pi_j, \mu, \rho, \boldsymbol{f}_j$ or just $\pi_j, \boldsymbol{f}_j$ in the first or the second case would follow precisely a uniform distribution as they would do in a real conversation (the permutation of uniformly random elements or the addition to uniformly random elements follows a uniform distribution and the fact that $\pi_j(\boldsymbol{e}'_j)$ has to be $\widehat{\pi}_j(\widehat{\boldsymbol{e}}'_j)$ is no restriction because $\widehat{\pi}_j$ and $\widehat{\boldsymbol{e}}'_j$ are uniformly distributed again).

3.4.2 Knowledge of a Linear Relation

The feasibility of proving that different committed elements are related is the most important feature of the presented commitment scheme, as then it can be used as building block for more complex cryptographic applications.

The linear structure of the commitment scheme aides the design of a linear relation proof. Given a secure public key $\boldsymbol{a}, \boldsymbol{b} \in R_q^k$ and valid commitments $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3 \in R_q^k$ with their respective openings to three messages $m_1, m_2, m_3 \in R_q$ such that $m_3 = \lambda_1 m_1 + \lambda_2 m_2$ for known $\lambda_1, \lambda_2 \in R_q$, the prover would like to convince the verifier of the fact that they not only know such valid openings but that the committed values m_i indeed satisfy the linear relation.

From now on, in this subsection and the next, index i will belong to $\{1, 2, 3\}$. The goal of the prover would then be to prove knowledge of $m_1, m_2, m_3 \in R_q$, $r_1, r_2, r_3 \in R_q$ and $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3 \in R_q^k$ so that $\boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + \boldsymbol{e}_i$, ensuring that the error terms have coefficients with norm bounded by B and that the committed values satisfy the linear relation with coefficients λ_1 and λ_2.

The main strategy consists on repeating three times in parallel the ZKPoK of a valid opening, for each of the three commitments. Then, an additional verification is added to check the linear relation.

The *shortness* of the error terms is not compatible with linear transformations, so it has to be proven independently decomposing again $\{e'_{ij}\}_j \leftarrow \mathsf{expand}(e_i)$ so that

$$c_i = a m_i + b r_i + \phi(\textstyle\sum_j 2^j e'_{ij}) - B.$$

To demonstrate that the committed messages satisfy the linear relation we just have to choose the message masking elements holding the same linear relation. μ_3 is computed as $\lambda_1\mu_1 + \lambda_2\mu_2$, and in case $b = 0$ the verifier just needs to check whether $z_3 = \lambda_1 z_1 + \lambda_2 z_2$. Provided $z_i = a(\mu_i + \alpha m_i)$ and the challenge α has been chosen by the verifier when the μ_i already form part of one of the auxiliary commitments we are going to be able to deduce that this implies the linear relation holds for the committed messages.

The full interaction between an honest prover $\mathcal{P}$ and a verifier $\mathcal{V}$ is completely specified in Protocol 3.2.

Completeness

Completeness comes from the completeness property of the opening protocol, as most of the computations are completely equivalent. The only new verification we should review is the linearity verification $z_3 \overset{?}{=} \lambda_1 z_1 + \lambda_2 z_2$.

From the completeness analysis of the opening protocol we already know that if the prover was honest $z_i = a(\mu_i + \alpha m_i)$. If the statement is true then $m_3 = \lambda_1 m_1 + \lambda_2 m_2$, and if the prover has followed the protocol then $\mu_3 = \lambda_1\mu_1 + \lambda_2\mu_2$. Combining both we get that the verification should pass, and the linear protocol is also complete.

Soundness

If a (possibly malicious) prover $\mathcal{P}^*$ is able to provide accepted answers to δ rounds of interaction with an honest verifier $\mathcal{V}$ with probability $\left({(q+1)}/{2q}\right)^\delta + \epsilon$, where ϵ is non-negligible, then they are able to efficiently extract a witness.

We can extend the $(2, 2)$-special soundness from the opening protocol to the linear proof. Applying the same arguments from four valid conversations with challenges $\alpha \neq \alpha'$ and $b = 0$ and $b = 1$ for both cases we could obtain three valid openings

$$\left\{(\Delta_\alpha^{-1}(\widetilde{t}_i - \widetilde{t}'_i), \Delta_\alpha^{-1}(\widetilde{s}_i - \widetilde{s}'_i), \phi(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\widetilde{e}'_{ij})) - B)\right\}_i.$$

$$\mathrm{ZKP}\left[\, m_i, r_i, \boldsymbol{e}_i \;\middle|\; \boldsymbol{c}_i = \boldsymbol{a} m_i + \boldsymbol{b} r_i + \boldsymbol{e}_i, \quad \|\boldsymbol{e}_i\|_\infty \le B, \quad m_3 = \lambda_1 m_1 + \lambda_2 m_2 \,\right] \tag{3.8}$$

Protocol 3.2 Knowledge of a Linear Relation

```
$\mathcal{P}((\boldsymbol{a},\boldsymbol{b}), \{\boldsymbol{c}_i\}_i, (\lambda_1,\lambda_2); \{m_i, r_i, \boldsymbol{e}_i\}_i)$                    $\mathcal{V}((\boldsymbol{a},\boldsymbol{b}), \{\boldsymbol{c}_i\}_i, (\lambda_1,\lambda_2))$
1:  for $i \in 1,2,3$ do
2:      $\{\boldsymbol{e}'_{ij}\}_j \leftarrow \mathsf{expand}(\boldsymbol{e}'_i)$
3:      $\pi_{i0}, \ldots, \pi_{i\log(B)} \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$
4:      $\boldsymbol{f}_{i0}, \ldots, \boldsymbol{f}_{i\log(B)} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$
5:  $\mu_1, \mu_2, \rho_1, \rho_2, \rho_3 \leftarrow_{\mathrm{R}} R_q$
6:  $\mu_3 := \lambda_1\mu_1 + \lambda_2\mu_2$
7:  $(c_1, o_1) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_{ij}\}_{i,j}, \{\boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})\}_i)$
8:  $(c_2, o_2) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_{ij}(\boldsymbol{f}_{ij})\}_{i,j}, \{\pi_{ij}(\boldsymbol{e}'_{ij})\}_{i,j})$
                                        $\xrightarrow{c_1, c_2}$
9:                                                                          $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_q$
                                        $\xleftarrow{\alpha}$
10: for $i \in 1,2,3$ do
11:     for $j \in 0,\ldots,\log(B)$ do
12:         $\boldsymbol{g}_{ij} := \pi_{ij}(\boldsymbol{f}_{ij} + \alpha\boldsymbol{e}'_{ij})$
                                        $\xrightarrow{\{\boldsymbol{g}_{ij}\}_{i,j}}$
13:                                                                         $b \leftarrow_{\mathrm{R}} \{0,1\}$
                                        $\xleftarrow{b}$
14: if $b = 0$ then
15:     for $i \in 1,2,3$ do
16:         $\widetilde{\boldsymbol{y}}_i := \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})$
17:         $\widetilde{s}_i := \rho_i + \alpha r_i$
18:         for $j \in 0,\ldots,\log(B)$ do
19:             $\widetilde{\pi}_{ij} := \pi_{ij}$
20:     $\widetilde{o} := o_1$
21:     $ans := (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{y}}_i\}_i, \{\widetilde{s}_i\}_i, \widetilde{o})$
22: if $b = 1$ then
23:     for $i \in 1,2,3$ do
24:         for $j \in 0,\ldots,\log(B)$ do
25:             $\widetilde{\boldsymbol{e}}'_{ij} := \pi_{ij}(\boldsymbol{e}'_{ij})$
26:     $\widetilde{o} := o_2$
27:     $ans := (\{\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}, \widetilde{o})$
                                        $\xrightarrow{ans}$
28:                                                                         if $b = 0$ then
29:                                                     $\overset{?}{\mathsf{aVer}}(c_1, (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{y}}_i\}_i), \widetilde{o})$
30:                                                                             for $i \in 1,2,3$ do
31:                 $\boldsymbol{z}_i := \widetilde{\boldsymbol{y}}_i + \alpha(\boldsymbol{c}_i + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}_i - \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\boldsymbol{g}_{ij})) \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
32:                                                                 $\boldsymbol{z}_3 \overset{?}{=} \lambda_1\boldsymbol{z}_1 + \lambda_2\boldsymbol{z}_2$
33:                                                                         if $b = 1$ then
34:                                         $\overset{?}{\mathsf{aVer}}(c_2, (\{\boldsymbol{g}_{ij} - \alpha\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}), \widetilde{o})$
35:                                                                             for $i \in 1,2,3$ do
36:                                                             for $j \in 0,\ldots,\log(B)$ do
37:                                                                     $\widetilde{\boldsymbol{e}}'_{ij} \overset{?}{\in} \mathfrak{B}_{nk}$
```

We have to be aware that in general linearity of some lattice points $z_i \in \mathcal{L}(\boldsymbol{a})$ does not always imply that the same relation holds for some coordinates of these points t_i such that $z_i = \boldsymbol{a}t_i$. That is the case because R_q is not an integral domain. Nevertheless, this property holds whenever $\boldsymbol{a}$ is part of the public key of a binding commitment. We can prove it using a similar argument than the one we saw before with Lemma 3.4.

Proposition 3.5 (Linearity of lattice points implies linearity of its coordinates if the commitment is binding)**.** *If $(\boldsymbol{a}, \boldsymbol{b}) \in R_q^k \times R_q^k$ defines a commitment key for a perfectly binding commitment scheme in a ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ then $z_3 = \lambda_1 z_1 + \lambda_2 z_2$ for $z_i = \boldsymbol{a}t_i \in R_q^k$ implies $t_3 = \lambda_1 t_1 + \lambda_2 t_2 \in R_q$.*

Proof. Assume by contradiction the statement is false and $z_3 = \lambda_1 z_1 + \lambda_2 z_2$ but $t_3 \neq \lambda_1 t_1 + \lambda_2 t_2$.

Then we could define $m_0 = t_3 - \lambda_1 t_1 - \lambda_2 t_2$. By assumption, we know $m_0 \neq 0$, but $\boldsymbol{a}m_0 = \boldsymbol{a}t_3 - \lambda_1 \boldsymbol{a}t_1 - \lambda_2 \boldsymbol{a}t_2 = z_3 - \lambda_1 z_1 - \lambda_2 z_2 = 0$.

This implies we could open any commitment to a message $m \in R_q$ to a different value $m + m_0$, contradicting the perfectly binding property of the commitment scheme. Our first assumption could not be correct, and the statement is therefore true. □

From the linear relation that the verifier would check if $b = 0$ involving the z_i or the z_i' we can therefore obtain that $\widetilde{t_3} = \lambda_1\widetilde{t_1} + \lambda_2\widetilde{t_2}$ and similarly that $\widetilde{t_3'} = \lambda_1\widetilde{t_1'} + \lambda_2\widetilde{t_2'}$. As a result, we have that the required linear relation holds

$$\Delta_\alpha^{-1}(\widetilde{t_3} - \widetilde{t_3'}) = \lambda_1\Delta_\alpha^{-1}(\widetilde{t_1} - \widetilde{t_1'}) + \lambda_2\Delta_\alpha^{-1}(\widetilde{t_2} - \widetilde{t_2'}).$$

That means a single repetition of the linear interactive protocol would be knowledge sound with knowledge error $\kappa = {}^{(q+1)}/_{2q}$. We could again repeat it δ times to reduce this knowledge error to κ^δ.

Zero-Knowledge

The same simulator for Protocol 3.1 works repeated 3 times in parallel, with the only exception that in case $b = 0$ we randomly choose $\widehat{t_1}, \widehat{t_2} \leftarrow_R R_q$ but $\widehat{t_3}$ would be computed as $\widehat{t_3} = \lambda_1\widehat{t_1} + \lambda_2\widehat{t_2}$.

This would be indistinguishable from a real conversation where $\mu_i = \widehat{t_i} + \alpha m_i$ and this last condition ensures that μ_3 would be $\lambda_1\mu_1 + \lambda_2\mu_2$ as it would have been in a real conversation between an honest prover and a verifier.

3.4.3 Knowledge of a Multiplicative Relation

In this subsection we present the main contribution of this chapter, an efficient proof of knowledge of a multiplicative relation. That is, index i belongs again to $\{1, 2, 3\}$, and we have $\boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + \boldsymbol{e}_i$ three valid commitments where $m_3 = m_1 \cdot m_2$. We want to prove knowledge of valid openings for the commitments $\boldsymbol{c}_i$ satisfying this relation.

If we mask the messages $(m_1 + \mu_1)$ and $(m_2 + \mu_2)$ with random $\mu_1, \mu_2 \leftarrow_{\mathrm{R}} R_q$, as we did before (let us omit the α challenge for the moment), and then multiply them, some crossed terms appear: $(m_1 + \mu_1)(m_2 + \mu_2) = m_3 + (m_1\mu_2 + m_2\mu_1) + \mu_1\mu_2$. Following the notation from [19] we define $m_+ := m_1\mu_2 + m_2\mu_1$ and $m_\times := \mu_1\mu_2$.

In order to be able to verify the multiplicative relation we also have to prove knowledge of honestly computed m_+ and $m_\times$, once again masking them with random polynomials μ_+ and $\mu_\times$.

If we want to get $m_3 = m_1 \cdot m_2$ we then have to prove a relation between masked elements and masking polynomials involving two challenges $\alpha, \beta \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ chosen by the verifier (now we need to use a different challenge for the third commitment as it plays a different role in the product verification). In [19] the authors use a challenge to prove the relation, while [38] introduces the challenge to reduce the soundness error of each round as we did in Section 3.4.1. The particular requirements of our proofs, where we try to achieve both goals at the same time, imply that we need a much more involved analysis in order to prove the soundness of this strategy. As we have mentioned, this efficient interactive protocol to prove knowledge of a valid opening for commitments $\boldsymbol{c}_i$ holding the multiplicative relation is what gives a special value to the commitment design.

The prover $\mathcal{P}$ starts expanding the error terms $\{\boldsymbol{e}'_{ij}\}_j \leftarrow \mathsf{expand}(\boldsymbol{e}_i)$. Then $\mathcal{P}$ chooses $3(\log(B) + 1)$ permutations $\pi_{i0}, \ldots, \pi_{i\log(B)} \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$ and $3(\log(B) + 1)$ random vectors $\boldsymbol{f}_{i0}, \ldots, \boldsymbol{f}_{i\log(B)} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$ that will be used to mask these errors.

To hide the messages and the randomness, $\mathcal{P}$ also samples 6 random polynomials $\mu_1, \mu_2, \mu_3, \rho_1, \rho_2, \rho_3 \leftarrow_{\mathrm{R}} R_q$. Next, $\mathcal{P}$ computes the crossed terms $m_\times := \mu_1\mu_2$ and $m_+ := \mu_1 m_2 + \mu_2 m_1$. Then they choose 2 additional random polynomials $\mu_\times, \mu_+ \leftarrow_{\mathrm{R}} R_q$ that will be used to hide these crossed terms while verifying they have been correctly computed.

Ultimately, $\mathcal{P}$ computes the following commitments,

$$\begin{aligned}
c_1 &\leftarrow_{\mathrm{R}} \mathsf{aCom}\Big(\{\pi_{ij}\}_{i,j}, \{\boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\textstyle\sum_j 2^j \boldsymbol{f}_{ij})\}_i\Big),\\
c_2 &\leftarrow_{\mathrm{R}} \mathsf{aCom}\Big(\mu_3, \mu_\times, \mu_+\Big),\\
c_3 &\leftarrow_{\mathrm{R}} \mathsf{aCom}\Big(\{\pi_{ij}(\boldsymbol{f}_{ij})\}_{i,j}, \{\pi_{ij}(\boldsymbol{e}'_{ij})\}_{i,j}\Big),\\
c_4 &\leftarrow_{\mathrm{R}} \mathsf{aCom}\Big(\mu_\times + m_\times, \mu_+ + m_+\Big).
\end{aligned}$$

Observe besides the usual two commitments regarding the structure and the shortness of the noise terms we now have two additional commitments involving the masked crossed terms and the masking elements relevant for the product verification.

The prover sends these commitments to the verifier. The verifier $\mathcal{V}$ chooses a pair of integers, $(\alpha, \beta) \leftarrow_{\mathrm{R}} \mathbb{Z}_q^2$, and sends it to the prover.

Let us define the following auxiliary constants to simplify notation,

$$\delta_i := \begin{cases} \alpha, & \text{for } i \in \{1,2\},\\ \beta, & \text{for } i \in \{3\}. \end{cases}$$

This way we can use the generic challenge δ_i when working with any of the three commitments without making any distinction.

Now $\mathcal{P}$ computes and sends the usual vectors of polynomials $\boldsymbol{g}_{ij}$, with the error terms hidden with both $\boldsymbol{f}_{ij}$ and π_{ij}, together with an additional commitment involving both the challenges and the multiplicative masking elements, that will be of paramount importance to prove the product relation.

$$\begin{aligned}
\boldsymbol{g}_{ij} &= \pi_{ij}(\boldsymbol{f}_{ij} + \delta_i \boldsymbol{e}'_{ij}),\\
c_5 &\leftarrow_{\mathrm{R}} \mathsf{aCom}\Big((\beta\mu_\times) + \alpha(\beta\mu_+) + \alpha^2(\mu_3)\Big).
\end{aligned}$$

The verifier $\mathcal{V}$ chooses a bit $b \leftarrow_{\mathrm{R}} \{0,1\}$ and sends it as second challenge.

Case $b = 0$.

- $\mathcal{P}$ reveals $\{\widetilde{\pi}_{ij} := \pi_{ij}\}_{i,j}$, $\{\widetilde{\boldsymbol{y}}_i := \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})\}_i$, $\widetilde{t}_\times := \mu_\times + m_\times$, $\widetilde{t}_+ := \mu_+ + m_+$ and $\{\widetilde{s}_i := \rho_i + \delta_i r_i\}_i$. The prover could also compute $\widetilde{t}_i := \mu_i + \delta_i m_i$, but does not send these polynomials as the verifier can compute them as the coordinates of $\boldsymbol{z}_i := \widetilde{\boldsymbol{y}}_i + \delta_i(\boldsymbol{c}_i + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}_i - \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\boldsymbol{g}_{ij})) \in \mathcal{L}(\boldsymbol{a})$. Then $\mathcal{P}$ sends openings of commitments c_1 to $(\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{y}}_i\}_i)$, c_4 to $(\widetilde{t}_\times, \widetilde{t}_+)$ and c_5 to $\beta\widetilde{t}_\times + \alpha\beta\widetilde{t}_+ + \alpha^2\widetilde{t}_3 - \beta\widetilde{t}_1\widetilde{t}_2$.
- $\mathcal{V}$ checks that $\widetilde{\boldsymbol{y}}_i + \delta_i(\boldsymbol{c}_i + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}_i - \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\boldsymbol{g}_{ij})) \in \mathcal{L}(\boldsymbol{a})$ and writes them as $\boldsymbol{a}\widetilde{t}_i$. Then $\mathcal{V}$ checks c_1, c_4 and c_5.

Case $b = 1$.

– $\mathcal{P}$ reveals $\{\widetilde{e}'_{ij} := \pi_{ij}(e'_{ij})\}_{i,j}$, $\widetilde{\mu}_3 := \mu_3$, $\widetilde{\mu}_\times := \mu_\times$, $\widetilde{\mu}_+ := \mu_+$ and openings of commitments c_2 to $(\widetilde{\mu}_3, \widetilde{\mu}_\times, \widetilde{\mu}_+)$, c_3 to $(\{g_{ij} - \delta_i \widetilde{e}'_{ij}\}_{i,j}, \{\widetilde{e}'_{ij}\}_{i,j})$ and c_5 to $(\beta\widetilde{\mu}_\times) + \alpha(\beta\widetilde{\mu}_+) + \alpha^2(\widetilde{\mu}_3)$.

– $\mathcal{V}$ checks c_2, c_3, c_5 and that each $\widetilde{e}'_{ij}$ belongs to $\mathfrak{B}_{nk}$.

The multiplicative relation Protocol 3.3 can also be seen as three parallel executions of Protocol 3.1, this time taking into account the crossed terms and the multiplicative relation.

The distinctive part of this protocol is that the fifth commitment can be opened using either the crossed masking elements μ_3, μ_+ and $\mu_\times$ that were already committed in c_2 or using the masked crossed elements $\mu_\times + m_\times$ and $\mu_+ + m_+$ that were committed in c_4 together with the masked messages $t_i = \mu_i + \delta_i m_i$.

The fact that c_2 and c_4 were committed before the challenges from the verifier were chosen will allow us to verify that the prover has not cheated with the crossed terms and as a consequence the verifier will be convinced of $m_3 = m_1 \cdot m_2$, as we are going to see when proving soundness.

Completeness

Completeness is again mostly inherited from the opening protocol. The new commitment c_2 is directly opened to the committed values and the verification passes if the auxiliary commitment has correctness. The same happens with the new commitment c_4, as $\widetilde{t}_\times$ and $\widetilde{t}_+$ are how the honest prover denotes $\mu_\times + m_\times$ and $\mu_+ + m_+$.

Only commitment c_5 requires our full attention. If $b = 1$ it is open to a polynomial computed in the same way $\mathcal{P}$ did when creating it. However, if $b = 0$ the proposed computation is different, and we need to check that the result is indeed the same, provided that the prover responses have been computed following the protocol.

Recall the $\widetilde{t}_i$ such that $z_i = a\widetilde{t}_i$ are unique and efficiently computable if the commitment key (a, b) defines a perfectly binding commitment (and this happens with overwhelming probability).

$$\begin{gathered}
\beta\widetilde{t}_\times + \alpha\beta\widetilde{t}_+ + \alpha^2\widetilde{t}_3 - \beta\widetilde{t}_1\widetilde{t}_2 = \\
= \beta(\mu_\times + m_\times) + \alpha\beta(\mu_+ + m_+) + \alpha^2(\mu_3 + \beta m_3) - \beta(\mu_1 + \alpha m_1)(\mu_2 + \alpha m_2) \\
= \beta(\mu_\times + m_\times - \mu_1\mu_2) + \alpha\beta(\mu_+ + m_+ - \mu_1 m_2 - \mu_2 m_1) + \alpha^2(\mu_3 + \beta(m_3 - m_1 m_2)) \\
= (\beta\mu_\times) + \alpha(\beta\mu_+) + \alpha^2(\mu_3)
\end{gathered}$$

$$\mathrm{ZKP}\left[m_i, r_i, \boldsymbol{e}_i \;\middle|\; \boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + \boldsymbol{e}_i,\ \|\boldsymbol{e}_i\|_\infty \le B,\ m_3 = m_1 \cdot m_2 \right] \tag{3.9}$$

Protocol 3.3 KNOWLEDGE OF A MULTIPLICATIVE RELATION

$\mathcal{P}((\boldsymbol{a},\boldsymbol{b}), \boldsymbol{c}_i; m_i, r_i, \boldsymbol{e}_i)$ $\mathcal{V}((\boldsymbol{a},\boldsymbol{b}), \boldsymbol{c}_i)$

```
1: for $i \in 1,2,3$ do
2:     $\{\boldsymbol{e}'_{ij}\}_j \leftarrow \mathsf{expand}(\boldsymbol{e}_i)$
3:     $\pi_{i0}, \ldots, \pi_{i\log(B)} \leftarrow_{\mathrm{R}} \mathfrak{S}_{2nk}$
4:     $\boldsymbol{f}_{i0}, \ldots, \boldsymbol{f}_{i\log(B)} \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$
5:     $\mu_i, \rho_i \leftarrow_{\mathrm{R}} R_q$
6: $m_\times := \mu_1\mu_2, \quad m_+ := \mu_1 m_2 + \mu_2 m_1$
7: $\mu_\times, \mu_+ \leftarrow_{\mathrm{R}} R_q$
8: $(c_1, o_1) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_{ij}\}_{i,j}, \{\boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})\}_i)$
9: $(c_2, o_2) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\mu_3, \mu_\times, \mu_+)$
10: $(c_3, o_3) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\{\pi_{ij}(\boldsymbol{f}_{ij})\}_{i,j}, \{\pi_{ij}(\boldsymbol{e}'_{ij})\}_{i,j})$
11: $(c_4, o_4) \leftarrow_{\mathrm{R}} \mathsf{aCom}(\mu_\times + m_\times, \mu_+ + m_+)$
                              $\xrightarrow{c_1, c_2, c_3, c_4}$
12:                                                            $\alpha, \beta \leftarrow_{\mathrm{R}} \mathbb{Z}_q$
                              $\xleftarrow{\delta_1 := \alpha,\quad \delta_2 := \alpha,\quad \delta_3 := \beta}$
13: $(c_5, o_5) \leftarrow_{\mathrm{R}} \mathsf{aCom}((\beta\mu_\times) + \alpha(\beta\mu_+) + \alpha^2(\mu_3))$
14: for $i \in 1,2,3$ and $j \in 0, \ldots, \log(B)$ do
15:     $\boldsymbol{g}_{ij} := \pi_{ij}(\boldsymbol{f}_{ij} + \delta_i \boldsymbol{e}'_{ij})$
                              $\xrightarrow{c_5, \{\boldsymbol{g}_{ij}\}_{i,j}}$
16:                                                            $b \leftarrow_{\mathrm{R}} \{0,1\}$
                              $\xleftarrow{b}$
17: if $b = 0$ then
18:     for $i \in 1,2,3$ do
19:         $\widetilde{\boldsymbol{y}}_i := \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})$
20:         for $j \in 0, \ldots, \log(B)$ do
21:             $\widetilde{\pi}_{ij} := \pi_{ij}$
22:     $\widetilde{t}_\times := \mu_\times + m_\times,\ \widetilde{t}_+ := \mu_+ + m_+,\ \widetilde{s}_i := \rho_i + \delta_i r_i,\quad \widetilde{o}_1 := o_1,\ \widetilde{o}_4 := o_4,\ \widetilde{o}_5 := o_5$
23:     $ans := (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{y}}_i\}_i, \widetilde{t}_\times, \widetilde{t}_+, \{\widetilde{s}_i\}_i, \widetilde{o}_1, \widetilde{o}_4, \widetilde{o}_5)$
24: if $b = 1$ then
25:     for $i \in 1,2,3$ and $j \in 0, \ldots, \log(B)$ do
26:         $\widetilde{\boldsymbol{e}}'_{ij} := \pi_{ij}(\boldsymbol{e}'_{ij})$
27:     $\widetilde{\mu}_3 := \mu_3,\ \widetilde{\mu}_\times := \mu_\times,\ \widetilde{\mu}_+ := \mu_+,\quad \widetilde{o}_2 := o_2,\ \widetilde{o}_3 := o_3,\ \widetilde{o}_5 := o_5$
28:     $ans := (\{\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}, \widetilde{\mu}_3, \widetilde{\mu}_\times, \widetilde{\mu}_+, \widetilde{o}_2, \widetilde{o}_3, \widetilde{o}_5)$
                              $\xrightarrow{ans}$
29:                                                            if $b = 0$ then
30:                                    $\mathsf{aVer}(c_1, (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{y}}_i\}_i), \widetilde{o}_1) \overset{?}{}$
31:                                    $\mathsf{aVer}(c_4, (\widetilde{t}_\times, \widetilde{t}_+), \widetilde{o}_4) \overset{?}{}$
32:                                    for $i \in 1,2,3$ do
33:                 $\boldsymbol{z}_i := \widetilde{\boldsymbol{y}}_i + \delta_i(\boldsymbol{c}_i + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}_i - \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\boldsymbol{g}_{ij})) \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
34:     Let $\widetilde{t}_i \in R_q$ s.t. $\boldsymbol{z}_i = \boldsymbol{a}\widetilde{t}_i$,  $\overset{?}{\mathsf{aVer}}(c_5, \beta\widetilde{t}_\times + \alpha\beta\widetilde{t}_+ + \alpha^2\widetilde{t}_3 - \beta\widetilde{t}_1\widetilde{t}_2, \widetilde{o}_5)$
35:                                                            if $b = 1$ then
36:                                    $\overset{?}{\mathsf{aVer}}(c_2, (\widetilde{\mu}_3, \widetilde{\mu}_\times, \widetilde{\mu}_+), \widetilde{o}_2)$
37:                                    $\overset{?}{\mathsf{aVer}}(c_3, (\{\boldsymbol{g}_{ij} - \alpha\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}, \{\widetilde{\boldsymbol{e}}'_{ij}\}_{i,j}), \widetilde{o}_3)$
38:                                    $\overset{?}{\mathsf{aVer}}(c_5, (\beta\widetilde{\mu}_\times) + \alpha(\beta\widetilde{\mu}_+) + \alpha^2(\widetilde{\mu}_3), \widetilde{o}_5)$
39:                                    for $i \in 1,2,3$ and $j \in 0, \ldots, \log(B)$ do
40:                                        $\widetilde{\boldsymbol{e}}'_{ij} \overset{?}{\in} \mathfrak{B}_{nk}$
```

Soundness

If a (possibly malicious) prover $\mathcal{P}^*$ is able to provide accepted answers to δ rounds of interaction with an honest verifier $\mathcal{V}$ with probability $\left((q^2+3q-2)/(2q^2)\right)^{\delta}+\epsilon$, where ϵ is non-negligible, then they are able to efficiently extract a witness.

The multiplicative protocol is more interesting than the previous ones, as we need six pairs (β,α_1), (β,α_2), (β,α_3) and (β',α'_1), (β',α'_2), (β',α'_3), with all α_i different and all α'_i different too, for which there are accepting transcripts for both $b=0$ and $b=1$. This structure is that of a $(2,3,2)$-special sound 7-move protocol, considering β as the first challenge, α as the second and b as the third. Of course, our 5-move protocol can be artificially transformed into a 7-move protocol just sending β as first challenge, waiting for an empty response from the prover, sending α as a second challenge and finally b as a third one. Since these two protocols are by all means equivalent any property that we can deduce when interpreting it as a $(2,3,2)$-special sound 7-move interactive protocol would also be true for our 5-move multiplicative protocol (this is only done for the sake of simplifying the proof, in any real instantiation both α and β could be sent at the same time).

In order to be able to refer to each of the transcripts we are going to use a notation enumerating all 6 pairs of conversations with a superscript. Let the six first challenge pairs be $(\alpha^{(1)},\beta^{(1)})$, $(\alpha^{(2)},\beta^{(2)})$, $(\alpha^{(3)},\beta^{(3)})$, $(\alpha^{(4)},\beta^{(4)})$, $(\alpha^{(5)},\beta^{(5)})$, $(\alpha^{(6)},\beta^{(6)})$, with all $\alpha^{(l)}$ different for $l\in\{1,2,3\}$, all $\alpha^{(l)}$ different for $l\in\{4,5,6\}$ and $\beta^{(1)}=\beta^{(2)}=\beta^{(3)}\neq\beta^{(4)}=\beta^{(5)}=\beta^{(6)}$.

The binding property of all auxiliary commitments ensures that openings to the same elements are equal. Therefore, we have fixed $\widetilde{\pi}_{ij}$, $\widetilde{\boldsymbol{y}}_i$, $\widetilde{\mu}_3$, $\widetilde{\mu}_\times$, $\widetilde{\mu}_+$, $\widetilde{\boldsymbol{e}}'_{ij}$, $\widetilde{t}_\times$ and $\widetilde{t}_+$, and there is no need to introduce superscripts for these elements (once again this is always true if the auxiliary commitment scheme is perfectly binding or reduces the success probability of the extractor by a negligible amount if it is only computationally binding).

For each pair $(\alpha^{(l)},\beta^{(l)})$ we have $\boldsymbol{g}_{ij}^{(l)}$ that do depend on the challenge. This is also the case for the $\widetilde{s}_i^{(l)}$

We know that $\boldsymbol{z}_i^{(l)}:=\widetilde{\boldsymbol{y}}_i+\delta_i^{(l)}(\boldsymbol{c}_i+\boldsymbol{B})-\boldsymbol{b}_i\widetilde{s}_i^{(l)}-\phi(\sum_j 2^j\widetilde{\pi}_{ij}^{-1}(\boldsymbol{g}_{ij}^{(l)}))\in\mathcal{L}(\boldsymbol{a})$ and call $\widetilde{t}_i^{(l)}$ to its uniquely defined coordinates. Let l and l' in $\{1,2,3,4,5,6\}$ such that $\Delta_{\delta_i}:=\delta_i^{(l)}-\delta_i^{(l')}\neq 0$. Then we will be able to compute valid openings of $\boldsymbol{c}_i$.

$$\begin{aligned}
\boldsymbol{a}\widetilde{t}_i^{(l)} &= \widetilde{\boldsymbol{y}}_i+\delta_i^{(l)}(\boldsymbol{c}_i+\boldsymbol{B})-\boldsymbol{b}\widetilde{s}_i^{(l)}-\phi\left(\textstyle\sum_j 2^j\widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l)}\right)\right)\\
\boldsymbol{a}\widetilde{t}_i^{(l')} &= \widetilde{\boldsymbol{y}}_i+\delta_i^{(l')}(\boldsymbol{c}_i+\boldsymbol{B})-\boldsymbol{b}\widetilde{s}_i^{(l')}-\phi\left(\textstyle\sum_j 2^j\widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l')}\right)\right)\\
\Delta_{\delta_i}\boldsymbol{c}_i &= \boldsymbol{a}\left(\widetilde{t}_i^{(l)}-\widetilde{t}_i^{(l')}\right)+\boldsymbol{b}\left(\widetilde{s}_i^{(l)}-\widetilde{s}_i^{(l')}\right)+\phi\left(\textstyle\sum_j 2^j\widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l)}-\boldsymbol{g}_{ij}^{(l')}\right)\right)-\Delta_{\delta_i}\boldsymbol{B}
\end{aligned}$$

$$\begin{aligned}\boldsymbol{c}_i = \boldsymbol{a}\left(\Delta_{\delta_i}^{-1}\left(\widetilde{t}_i^{(l)} - \widetilde{t}_i^{(l')}\right)\right) + \boldsymbol{b}\left(\Delta_{\delta_i}^{-1}\left(\widetilde{s}_i^{(l)} - \widetilde{s}_i^{(l')}\right)\right) \\ + \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\Delta_{\delta_i}^{-1}\left(\boldsymbol{g}_{ij}^{(l)} - \boldsymbol{g}_{ij}^{(l')}\right)\right)\right) - \boldsymbol{B}\end{aligned}$$

As we did with the opening proof we can use that the opening to $\widetilde{\boldsymbol{f}}_{ij}$ is unique to relate the $\boldsymbol{g}_{ij}$ to the permuted extended errors $\widetilde{\boldsymbol{e}}_{ij}$

$$\begin{aligned}\boldsymbol{g}_{ij}^{(l)} - \delta_i^{(l)}\widetilde{\boldsymbol{e}}'_{ij} &= \boldsymbol{g}_{ij}^{(l')} - \delta_i^{(l')}\widetilde{\boldsymbol{e}}'_{ij} \\ \widetilde{\boldsymbol{e}}'_{ij} &= \Delta_{\delta_i}^{-1}\left(\boldsymbol{g}_{ij}^{(l)} - \boldsymbol{g}_{ij}^{(l')}\right)\end{aligned}$$

and replace them in the previous equation to obtain a valid opening.

$$\boldsymbol{c}_i = \boldsymbol{a}\left(\Delta_{\delta_i}^{-1}\left(\widetilde{t}_i^{(l)} - \widetilde{t}_i^{(l')}\right)\right) + \boldsymbol{b}\left(\Delta_{\delta_i}^{-1}\left(\widetilde{s}_i^{(l)} - \widetilde{s}_i^{(l')}\right)\right) + \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\widetilde{\boldsymbol{e}}'_{ij}\right)\right) - \boldsymbol{B}.$$

As these elements come from accepted answers we know that $\widetilde{\boldsymbol{e}}'_{ij} \in \mathfrak{B}_{nk} \subset \{0,1\}^{2nk}$ and therefore $\phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\widetilde{\boldsymbol{e}}'_{ij})) - \boldsymbol{B}$ has norm smaller than B. This implies that $(\Delta_{\delta_i}^{-1}(\widetilde{t}_i^{(l)} - \widetilde{t}_i^{(l')}), \Delta_{\delta_i}^{-1}(\widetilde{s}_i^{(l)} - \widetilde{s}_i^{(l')}), \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\widetilde{\boldsymbol{e}}'_{ij})) - \boldsymbol{B})$ are valid openings.

The same way we introduced notation for elements a possible malicious prover discloses or elements the simulator outputs we now use $\overline{a}$ to denote the extracted element we intend to prove plays the role of a.

We know that the opened messages do not depend on (l) and (l'), because the commitment scheme is perfectly binding. Therefore, we can call them ($\overline{m}_i := \Delta_{\delta_i}^{-1}(\widetilde{t}_i^{(l)} - \widetilde{t}_i^{(l')}), \overline{r}_i^{(l)} := \Delta_{\delta_i}^{-1}(\widetilde{s}_i^{(l)} - \widetilde{s}_i^{(l')}), \overline{\boldsymbol{e}}_i := \phi(\sum_j 2^j \widetilde{\pi}_{ij}^{-1}(\widetilde{\boldsymbol{e}}'_{ij})) - \boldsymbol{B}$) without any superscript on the message.

It only remains to prove that $\overline{m}_3 = \overline{m}_1 \cdot \overline{m}_2$. To do so, we have committed to the crossed terms masking elements in c_5, and we have seen in the completeness section that the alternative opening involving the $\widetilde{t}_i$ (and therefore the messages m_i) works if $m_3 = m_1 \cdot m_2$. To prove the other direction, the fact that if both openings for each of the 6 pairs of transcripts are valid then $\overline{m}_3 = \overline{m}_1 \cdot \overline{m}_2$, we use the usual technique ensuring that everything else was determined before the challenges were chosen and that the statement holds for different challenges.

From the elements involved in the openings of c_5 only $\widetilde{t}_1^{(l)}$, $\widetilde{t}_2^{(l)}$ and $\widetilde{t}_3^{(l)}$ might depend on (l) (and were not determined before the challenges were chosen). We can define $\overline{\mu}_i^{(l)} := \widetilde{t}_i^{(l)} - \delta_i^{(l)}\overline{m}_i$ and $\overline{\rho}_i^{(l)} := \widetilde{s}_i^{(l)} - \delta_i^{(l)}\overline{r}_i^{(l)}$, and study their dependency on l.

Claim 3.6. *These newly defined $\overline{\mu}_i^{(l)}$ do not depend on l, as we have $\overline{\mu}_i^{(l)} = \overline{\mu}_i^{(l')}$ for any pair l and l'.*

Proof. Assume that we have l and l' such that $\overline{\mu}_i^{(l)} \neq \overline{\mu}_i^{(l')}$.

We could rewrite the expression of $\boldsymbol{a}\widetilde{t}_i^{(l)}$ in terms of these new variables.

$$\begin{aligned}
\boldsymbol{a}\widetilde{t}_i^{(l)} &= \widetilde{\boldsymbol{y}}_i + \delta_i^{(l)}(\boldsymbol{c}_i + \boldsymbol{B}) - \boldsymbol{b}\widetilde{s}_i^{(l)} - \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l)}\right)\right)\\
\boldsymbol{a}\left(\overline{\mu}_i^{(l)} + \delta_i^{(l)}\overline{m}_i\right) &= \widetilde{\boldsymbol{y}}_i + \delta_i^{(l)}\left(\boldsymbol{a}\overline{m}_i + \boldsymbol{b}\overline{r}_i^{(l)} + \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\widetilde{\boldsymbol{e}}'_{ij}\right)\right)\right)\\
&\quad - \boldsymbol{b}\left(\overline{\rho}_i^{(l)} + \delta_i^{(l)}\overline{r}_i^{(l)}\right) - \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l)}\right)\right)\\
\boldsymbol{a}\overline{\mu}_i^{(l)} + \boldsymbol{b}\overline{\rho}_i^{(l)} &= \widetilde{\boldsymbol{y}}_i - \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\boldsymbol{g}_{ij}^{(l)} - \delta_i^{(l)}\widetilde{\boldsymbol{e}}'_{ij}\right)\right)
\end{aligned}$$

Notice that $\boldsymbol{g}_{ij}^{(l)} - \delta_i^{(l)}\widetilde{\boldsymbol{e}}'_{ij}$ is open to $\widetilde{\boldsymbol{f}}_{ij}$, that was committed before $\alpha^{(l)}$ and $\beta^{(l)}$ were chosen and therefore does not depend on l,

$$\boldsymbol{a}\overline{\mu}_i^{(l)} + \boldsymbol{b}\overline{\rho}_i^{(l)} = \widetilde{\boldsymbol{y}}_i - \phi\left(\textstyle\sum_j 2^j \widetilde{\pi}_{ij}^{-1}\left(\widetilde{\boldsymbol{f}}_{ij}\right)\right).$$

Since the right-hand side does not depend on l nor l' from two equations we get

$$\boldsymbol{a}\left(\overline{\mu}_i^{(l)} - \overline{\mu}_i^{(l')}\right) + \boldsymbol{b}\left(\overline{\rho}_i^{(l)} - \overline{\rho}_i^{(l')}\right) = \boldsymbol{0}.$$

We can again use that $(\boldsymbol{a}, \boldsymbol{b})$ define a perfectly binding commitment scheme. Any valid commitment $\boldsymbol{c} = \boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ could be opened to $(m + \overline{\mu}_i^{(l)} - \overline{\mu}_i^{(l')}, r + \overline{\rho}_i^{(l)} - \overline{\rho}_i^{(l')}, \boldsymbol{e})$, and that would break the binding property if $\overline{\mu}_i^{(l)} \neq \overline{\mu}_i^{(l')}$.

Therefore, $\overline{\mu}_i$ does not depend on (l), and we can omit the superscript. □

We can also define $\overline{m}_\times := \widetilde{t}_\times - \widetilde{\mu}_\times$, $\overline{m}_+ := \widetilde{t}_+ - \widetilde{\mu}_+$. This time there is no dependence with l as the elements were previously committed. With all these discussions now we are ready to prove the relation $\overline{m}_3 = \overline{m}_1 \cdot \overline{m}_2$.

From the two openings of the c_5 commitment scheme in each pair of conversations with the same initial challenges $(\alpha^{(l)}, \beta^{(l)})$ we get the following equation

$$\alpha^{(l)^2}(\widetilde{\mu}_3) + \alpha^{(l)}(\beta^{(l)}\widetilde{\mu}_+) + (\beta^{(l)}\widetilde{\mu}_\times) = \beta^{(l)}\widetilde{t}_\times + \alpha^{(l)}\beta^{(l)}\widetilde{t}_+ + \alpha^{(l)^2}\widetilde{t}_3^{(l)} - \beta^{(l)}\widetilde{t}_1^{(l)}\widetilde{t}_2^{(l)}.$$

We can expand the right-hand side to see how the dependency on l only appears in the challenges $\alpha^{(l)}$ and $\beta^{(l)}$ and get

$$\beta^{(l)}(\widetilde{\mu}_\times + \overline{m}_\times) + \alpha^{(l)}\beta^{(l)}(\widetilde{\mu}_+ + \overline{m}_+) + \alpha^{(l)^2}(\overline{\mu}_3 + \beta^{(l)}\overline{m}_3) - \beta^{(l)}(\overline{\mu}_1 + \alpha^{(l)}\overline{m}_1)(\overline{\mu}_2 + \alpha^{(l)}\overline{m}_2).$$

From the equivalence between the left-hand side and the expanded version of the right-hand side, reordering the elements, we can finally get the following expression

$$\begin{aligned}&\alpha^{(l)^2}(\widetilde{\mu}_3 - \overline{\mu}_3 + \beta^{(l)}(\overline{m}_1\overline{m}_2 - \overline{m}_3))\\ &+\alpha^{(l)}(\beta^{(l)}(\overline{\mu}_1\overline{m}_2 + \overline{\mu}_2\overline{m}_1 - \overline{m}_+))\\ &+(\beta^{(l)}(\overline{\mu}_1\overline{\mu}_2 - \overline{m}_\times)) = 0.\end{aligned}$$

If we restrict ourselves to the cases with equal β we can see this expression as a two degree polynomial in α (the coefficients were committed before the challenges were chosen), that is equal to 0 for three evaluations $\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}$ or $\alpha^{(4)}, \alpha^{(5)}, \alpha^{(6)}$. This implies that it is the 0 polynomial and that all its coefficients are 0, providing us with the equalities $\widetilde{\mu}_3 - \overline{\mu}_3 + \beta^{(l)}(\overline{m}_1\overline{m}_2 - \overline{m}_3) = 0$. Given that this equality is satisfied by two different $\beta^{(l)} \neq \beta^{(l')}$, we have that $(\beta^{(l)} - \beta^{(l')})(\overline{m}_1\overline{m}_2 - \overline{m}_3) = 0$ and finally $\overline{m}_3 = \overline{m}_1\overline{m}_2$ as we wanted to prove. The relation holds for the extracted witness.

That means, using eq. (3.7), that a single repetition of the interactive multiplicative protocol would be knowledge sound with knowledge error

$$\kappa = 1 - \frac{q-2+1}{q} \cdot \frac{q-3+1}{q} \cdot \frac{2-2+1}{2} = \frac{q^2+3q-2}{2q^2},$$

that can be reduced to κ^δ with δ parallel repetitions. In this case the extractor from [11] would require 12 rewinds, because we are considering it as a $(2, 3, 2)$-special-sound protocol.

Zero-Knowledge

Zero-knowledge can be proved again explicitly describing a simulator $\mathcal{S}$ that outputs conversations indistinguishable from honest ones without any knowledge of the witness. The multiplicative protocol is much more complex than the previous ones, but we have included sufficiently many masking elements to make it possible.

Case $b = 0$.

$$\begin{aligned}&\widehat{t}_i, \widehat{s}_i \leftarrow_R R_q, \quad \widehat{t}_\times, \widehat{t}_+ \leftarrow_R R_q\\ &\widehat{g}_{ij} \leftarrow_R \mathbb{Z}_q^{2nk}, \quad \widehat{\pi}_{ij} \leftarrow_R \mathfrak{S}_{2nk}\\ &c_1 \leftarrow_R \mathsf{aCom}\big(\{\widehat{\pi}_{ij}\}_{i,j}, \{\boldsymbol{a}\widehat{t}_i + \boldsymbol{b}\widehat{s}_i + \phi(\textstyle\sum_j 2^j \widehat{\pi}_{ij}^{-1}(\widehat{g}_{ij})) - \delta_i(\boldsymbol{c}_i + \boldsymbol{B})\}_i\big)\\ &c_4 \leftarrow_R \mathsf{aCom}\big(\widehat{t}_\times, \widehat{t}_+\big)\\ &c_5 \leftarrow_R \mathsf{aCom}\big(\beta\widehat{t}_\times + \alpha\beta\widehat{t}_+ + \alpha^2\widehat{t}_3 - \beta\widehat{t}_1\widehat{t}_2\big)\end{aligned}$$

$\mathcal{S}$ reveals $\{\widehat{\boldsymbol{g}}_{ij}\}_{i,j}$, $\{\widetilde{\pi}_{ij} = \widehat{\pi}_{ij}\}_{i,j}$, $\{\widetilde{\boldsymbol{y}}_i = \boldsymbol{a}\widehat{t}_i + \boldsymbol{b}\widehat{s}_i + \phi(\sum_j 2^j \widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij})) - \delta_i(\boldsymbol{c}_i + \boldsymbol{B})\}_i$, $\widetilde{t}_\times, \widetilde{t}_+, \{\widetilde{s}_i\}_i$.

Indistinguishable from a real conversation with the same $\pi_{ij} = \widehat{\pi}_{ij}$ and where $\mu_i = \widehat{t}_i - \delta_i m_i$, $\mu_\times = \widehat{t}_\times - m_\times$, $\mu_+ = \widehat{t}_+ - m_+$, $\rho_i = \widehat{s}_i - \delta_i r_i$ and $\boldsymbol{f}_{ij} = \widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij}) - \delta_i \boldsymbol{e}'_{ij}$.

We can check every element would then be the one outputted by the simulator.

$$\begin{aligned}
\boldsymbol{g}_{ij} &= \pi_{ij}(\boldsymbol{f}_{ij} + \delta_i \boldsymbol{e}'_{ij}) \\
&= \pi_{ij}(\widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij})) \\
&= \widehat{\boldsymbol{g}}_{ij} \\
\widetilde{\pi}_{ij} &= \widehat{\pi}_{ij} \\
\boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\textstyle\sum_j 2^j \boldsymbol{f}_{ij}) &= \\
&= \boldsymbol{a}(\widehat{t}_i - \delta_i m_i) + \boldsymbol{b}(\widehat{s}_i - \delta_i r_i) + \phi(\textstyle\sum_j 2^j (\widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij}) - \delta_i \boldsymbol{e}'_{ij})) \\
&= \boldsymbol{a}\widehat{t}_i + \boldsymbol{b}\widehat{s}_i + \phi(\textstyle\sum_j 2^j \widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij})) - \delta_i(\boldsymbol{a}m_i + \boldsymbol{b}r_i + \phi(\textstyle\sum_j 2^j \boldsymbol{e}'_{ij})) \\
&= \boldsymbol{a}\widehat{t}_i + \boldsymbol{b}\widehat{s}_i + \phi(\textstyle\sum_j 2^j \widehat{\pi}_{ij}^{-1}(\widehat{\boldsymbol{g}}_{ij})) - \delta_i(\boldsymbol{c}_i + \boldsymbol{B}) \\
\widetilde{t}_\times &= \mu_\times + m_\times \\
&= \widehat{t}_\times \\
\widetilde{t}_+ &= \mu_+ + m_+ \\
&= \widehat{t}_+ \\
\widetilde{s}_i &= \rho_i + \delta_i r_i \\
&= \widehat{s}_i
\end{aligned}$$

Case $b = 1$.

$$\begin{aligned}
&\widehat{\mu}_3, \widehat{\mu}_\times, \widehat{\mu}_+ \leftarrow_R R_q, \quad \widehat{\boldsymbol{e}}'_{ij} \leftarrow_R \mathfrak{B}_{nk} \\
&\widehat{\boldsymbol{f}}_{ij} \leftarrow_R \mathbb{Z}_q^{2nk}, \quad \widehat{\pi}_{ij} \leftarrow_R \mathfrak{S}_{2nk} \\
&c_2 \leftarrow_R \mathsf{aCom}(\widehat{\mu}_3, \widehat{\mu}_\times, \widehat{\mu}_+) \\
&c_3 \leftarrow_R \mathsf{aCom}(\{\widehat{\pi}_{ij}(\widehat{\boldsymbol{f}}_{ij})\}_{i,j}, \{\widehat{\pi}_{ij}(\widehat{\boldsymbol{e}}'_{ij})\}_{i,j}) \\
&c_5 \leftarrow_R \mathsf{aCom}(\beta\widehat{\mu}_\times + \alpha\beta\widehat{\mu}_+ + \alpha^2\widehat{\mu}_3) \\
&\widehat{\boldsymbol{g}}_{ij} = \widehat{\pi}_{ij}(\widehat{\boldsymbol{f}}_{ij} + \delta_i \widehat{\boldsymbol{e}}'_{ij})
\end{aligned}$$

$\mathcal{S}$ reveals $\{\widehat{g}_{ij}\}_{i,j}$, $\{\widetilde{e}'_{ij} = \widehat{\pi}_{ij}(\widehat{e}'_{ij})\}_{i,j}$, $\widetilde{\mu}_3 = \widehat{\mu}_3$, $\widetilde{\mu}_\times = \widehat{\mu}_\times$, $\widetilde{\mu}_+ = \widehat{\mu}_+$.

Equivalent to an honest conversation with equal $\mu_3 = \widehat{\mu}_3, \mu_\times = \widehat{\mu}_\times, \mu_+ = \widehat{\mu}_+$ and were π_{ij} is such that $\pi_{ij}(e'_{ij}) = \widehat{\pi}_{ij}(\widehat{e}'_{ij})$ and $f_{ij} = \pi_{ij}^{-1}(\widehat{\pi}_{ij}(\widehat{f}_{ij}))$. In this case the only elements not immediate are the g_{ij}.

$$\begin{aligned}
g_{ij} &= \pi_{ij}(f_{ij} + \delta_i e'_{ij}) \\
&= \pi_{ij}(\pi_{ij}^{-1}(\widehat{\pi}_{ij}(\widehat{f}_{ij}))) + \delta_i \widehat{\pi}_{ij}(\widehat{e}'_{ij}) \\
&= \widehat{\pi}_{ij}(\widehat{f}_{ij} + \delta_i \widehat{e}'_{ij}) \\
&= \widehat{g}_{ij}
\end{aligned}$$

Notice again that simulated conversations follow the proper distributions, as they are equivalent to honest conversations where the elements follow the expected distributions.

3.5 Conclusions

To sum up, we have proposed a new protocol for proving linear and multiplicative relations between secret elements hidden inside RLWE samples. The direct applications are new zero-knowledge proofs for proving knowledge of the evaluations of arithmetic circuits with committed inputs.

Jain *et al.* [64] commitment scheme was the first proposal of this kind, based on the LPN problem, that is, their commitment scheme only allowed to commit to $\{0,1\}^n$ messages. Xie *et al.* [124] proposed exact Stern-based proofs for lattice-based commitments, but they had a factor $\log(q)^2$ overhead to the messages. We are able to build exact proofs with a constant factor overhead, thus further improving efficiency. Besides that, our scheme is compatible with the techniques that reduce the soundness error to $1/2$, and as a consequence it requires fewer repetitions to achieve the same confidence level. Several constructions using Xie *et al.* zero-knowledge proofs for relations between committed messages (as the lattice-based Attributed Based Signature scheme for unbounded circuits [47]) could benefit from this improvement directly replacing their proofs with our proposal.

Our scheme can be directly compared to the one proposed by Benhamouda *et al.* [19]. While their proofs do not require repetitions our proposal achieves the same security level with smaller commitments, as we do not generalize the definition of opening of the commitment. It is also more robust and easy to implement, as in our protocol the prover is always able to answer with a valid response, without any abort probability. Finally, we require a significantly smaller modulus q for our construction to be sound. This implies that our schemes can still be used as a building block in larger protocols where it would be much less efficient (or even unfeasible) to increase the modulus q for the whole protocol. That could be the case for electronic voting, where heavy ZKPoKs could be performed on some servers, but votes have to be encrypted using resource constrained voting devices. More detailed comparisons and cost analysis, along with the differences between this version and the proposal published in [87] can be found in Appendix 3.A.

We think that these properties represent a major improvement on constructions based on Stern's protocol and might be useful in applications that heavily require this kind of proofs. We also think that our ideas are flexible enough to be applied as building blocks for other different constructions besides commitment schemes.

Appendices 3

3.A Differences with the Published Version

As we have already mentioned the commitment scheme and the interactive ZKPoKs introduced in this chapter were previously presented in the 17th IMA International Conference on Cryptography and Coding, celebrated in Oxford in 2019 [87]. Nevertheless, this chapter has significant differences with the version published in [87], as we have complemented its asymptotic security proofs with concrete security estimations. Besides that, we have also further improved some demonstrations justifying missing details and obtaining more tight results. The fundamental differences are going to be described in this appendix.

The original version in [87] established some asymptotic conditions on the parameters using the same kind of bounds used in [19] for the Benhamouda *et al.* commitment scheme. This way, looking at the theoretical inequalities that the parameters have to satisfy it was possible to directly compare both constructions without actually calculating the secure parameters. The comparison is well detailed in [87], showing that we obtain a smaller commitment size for the same n and q (because our scheme requires a smaller overhead k), and most importantly it can be instantiated with a much smaller q. Conversely, [19] obtains smaller proof sizes, mostly because they do not need to repeat the protocols to obtain negligible soundness (although the fact that the running time of their proofs depends on the secret would require an overhead to prevent side-channel timing attacks). It is also interesting the theoretical comparison with the scheme from Xie *et al.* [124], provided that we reduce the size of the proofs by a factor $\log^2(q)$ given that we do not need to binary decompose the messages, and also reduce the number of repetitions by reducing the soundness error from $2/3$ to almost $1/2$.

However, these conditions on the parameters, that allow theoretical comparisons and work for proving asymptotic security, are sometimes far from tight and not that useful when it comes to searching the most efficient possible secure set of parameters. For that matter, the security proofs have been revamped. To do so we have removed some constants, introduced new ones, and replaced some conditions with others that play the same role but allow a tighter analysis.

It is important to emphasize that this is one of the main challenges when instantiating real world cryptographic protocols from a theoretic proposal. Proving security from a theoretical point of view requires asymptotic proofs, as its goal is ensuring that undesired events occur with as low probability as we desire provided some parameters are large enough. These proofs guarantee the existence of parameters for any security level (under certain hypothesis), but do not inform of the particular parameters given that they might not explicitly describe how large these parameters should be, and have to be generic enough to work with any set of parameters.

Claiming a specific security level λ is something different, as the constants hidden in the asymptotic relations play a really important role, and we have been able to tailor some inequalities to the specific set of parameters we would like to work with.

Both [87, 19] use $\sigma \in \mathcal{O}\left(n^{3/4}\right)$ as error parameter, but this is not useful for instantiating the commitment. On the one hand, fixing a particular relation $\sigma = c \cdot n^{3/4}$ might require absurdly large n for the properties to hold for some c, or, on the other hand, it might produce far from tight inequalities. Explicitly computing how large n has to be, or how tight are the proofs, is not straightforward in general.

Asymptotic relations are defined for functions, but, in order to study the best σ for a given n, a different analysis is required. It is also relevant to notice that the original proofs used as an assumption the hardness of the underlying RLWE problem, but did not discuss how the relations among the parameters affect this hardness. This again has to be specifically consider when instantiating the commitment.

For these reasons some constrains and the proofs from this chapter differ from its counterparts in [87]. Let us list the main differences.

Remove intermediate constant γ. Consequently, we no longer make use of the intermediate constant γ controlling the relation between q and n, imposing $q \geq n^{\gamma}$. The original version [87] inherited it from [19], and it was useful to compare both schemes from a theoretical point of view, but not for choosing parameters. We have instead explicitly checked the relations between q and n for each security property.

Decoupling of noise bound from n introducing new bound B. Both [19] and [87] bounded the error term by n, so that σ could be asymptotically defined with respect to n and security proofs would then be more straightforward. As we have mentioned, choosing specific parameters requires computing actual probabilities, and not its asymptotic behavior. Doing so we found out that it is possible to choose much tighter bounds on the error terms preserving all the other properties. For that matter, we have introduced from the very beginning a new parameter B, intended to be a much tighter upper bound to the norm of the error term. In the following chapter we are going to see that this was indeed a right presumption, because we can obtain secure sets with $B \ll n$.

Alternative constraint of the modulus q. Besides redoing the proofs for security considerations, we also change another condition, regarding the modulus q, to allow further optimizations. To improve the efficiency of polynomial multiplications, we use the partial FFT approach that has been described in Chapter 2.

For this reason we have imposed $q \equiv 2d + 1 \mod 4d$ so that $x^n + 1$ factorizes into d irreducible polynomials in $\mathbb{Z}_q[x]$. Notice that when $d = 2$, which is the most efficient case as we will discuss, this translates into $q \equiv 5 \mod 8$, while the original proposals [19, 87] required $q \equiv 3 \mod 8$, which also ensure by a different theorem that $x^n + 1$ factorizes into two different irreducible polynomials of degree $n/2$ in $\mathbb{Z}_q[x]$.

Missing details. Notice that the original soundness proof from [87] implicitly assumes that given $z \in \mathcal{L}(\boldsymbol{a})$ one can compute $t \in R_q$ such that $z = \boldsymbol{a}t$. This might not be true for any lattice, but we have formally showed in Section 3.4.1 that it is true if $\boldsymbol{a}$ is the first vector of polynomials from the public key $(\boldsymbol{a}, \boldsymbol{b})$ of the commitment scheme under the conditions that imply the binding property of the commitment. Something analogous happens in Section 3.4.2 about the fact that the linearity of some $z_i \in \mathcal{L}(\boldsymbol{a})$ implies the linearity of the coordinates $t_i \in R_q$ such that $z_i = \boldsymbol{a}t_i$, which was not explicitly proven in [87]. The fact that $(\boldsymbol{a}, \boldsymbol{b})$ defines a binding commitment is once again used in Section 3.4.3 to prove soundness of the multiplicative relation. This was proved differently in [87], only obtaining soundness except with a negligible probability on the sampling of the public keys (slightly different to the negligible probability of the commitment not being binding).

Standard soundness. To be able to prove soundness, the published version [87] used the notion of k-special soundness that we informally defined in Section 1.2.4 (see [87] for the formal definition). This property states that a witness can be extracted from k accepting conversations with different challenges. Notice that this does not immediate provide knowledge-soundness, as one has to build the extractor that efficiently gets these k conversations.

We could say, as it is done in [87], that $q + 2$ transcripts, with different pairs of challenges in $\mathbb{Z}_q \times \{0, 1\}$, ensure by the pigeonhole principle the existence of the four transcripts that we really need (with α as the first challenge in two of them, $\alpha' \neq \alpha$ in the other two, and both $b = 0$ and $b = 1$ for each first challenge), but it seems an overkill. That is because the k-special-soundness definition from [87] does not fully capture the nature of multi-round protocols.

Many proposals in the literature do not even analyze how the extractor would work. As it was mentioned in [87] the size of the potential execution tree considering all possible challenges would be of the order of q^{δ}. The [87] soundness proof already partially addressed this issue, providing an efficient witness extractor sufficient to ensure soundness of the interactive protocol.

In [87] we managed to prove that, provided oracle access to a malicious prover that produces accepting conversations with probability $\epsilon \geq ((q+1)/2q)^{\delta}$, let us call $\kappa = (q+1)/2q$, there exists an efficient extractor that outputs a valid witness with probability $2(\epsilon - \kappa^{\delta})^3/27$. It was sufficient to prove soundness of the interactive proof for which it is sufficient to see that a witness can be extracted with positive probability, but it does not satisfy the standard notion of knowledge-soundness with knowledge error κ^{δ}, that would require the probability to be proportional to $(\epsilon - \kappa^{\delta})$, while that proof only achieves a relaxed version where we have a cubic loss in the success probability. The standard definition also asks for an extractor that, given oracle access to any successful enough malicious prover, outputs a witness with sufficient probability. The proof from [87] proves the existence of such an extractor that works with each prover, but that extractor could be different each time. That is because that proof shows that it is sufficient to focus on a single individual thread of the δ repetitions, however the index of that thread could be different for each prover. This could be naïvely addressed, and the extractor can be made universal by choosing that index at random (at the expense of dividing by δ the guaranteed success probability), or trying it for every index (at the expense of increasing its computational cost by a factor δ).

This proposed extractor would work in a running time just proportional to q (or to δq to comply with the standard definition as we have seen). Once a useful node in the execution tree (an initial message for which the prover can correctly answer to sufficiently many challenges) is found, all the $q+2$ transcripts have to be obtained rewinding the prover. That is still a significant work regarding that, as we have said, only 4 of them are really relevant, given that the witness can be obtained from 4 transcripts.

We have further improved this analysis in Section 3.4.1, using the $(k_1, \ldots, k_{\mu})$-Special Soundness from Definition 3.2, because it is much more suitable for protocols with more than 3 moves, as it is our case. Using it we have been able to prove that the protocols satisfy the standard definition of knowledge-soundness, without the relaxations to the definition that were necessary in [87].

Chapter 4

Commitment and NIZKPoKs Implementation

In this chapter we continue the work from the previous one. We explain how to obtain Non-Interactive Zero-Knowledge Proofs of Knowledge for the same statements, valid openings and linear or multiplicative relations among committed elements. Then, we prove them secure and showcase an implementation of both the commitment and the proofs. We develop a procedure to obtain optimal secure sets of parameters and evaluate the performance of the scheme showing both the size of the commitments and the proofs, and the running time of all the defined algorithms.

4.1 Introduction

The aim of this chapter is to analyze the practical performance of the commitment scheme and the proofs defined in the previous chapter. We believe it is important to bridge the gap between theoretical proposals and actual implementations, at least providing a working prototype so that we can fully address the most relevant practical issues that might arise in a real use case.

In order to make the analysis as complete as possible, we have decided to implement non-interactive variants of the ZKPoKs presented so far. First of all, the benchmarks we can perform with our implementation of a non-interactive proof will more closely resemble a real scenario than a simulated interactive protocol running both parties in a single computer, estimating or omitting the communication delays. We will then obtain more meaningful results following this approach. Secondly, non-interactive proofs can be universally verifiable, meaning that the same proof computed once can be verified by any verifier without requesting the prover to do any additional computation, so it is relevant to analyze this scenario too.

4.1.1 Structure of the Chapter

The non-interactive versions can be obtained in the ROM from the interactive protocols applying the Fiat-Shamir transform, substituting the random challenges by pseudorandom elements computed from the previous messages. We first devote Section 4.2 to introduce the modifications needed to get the NIZKPoKs versions and include the pseudocode that would be used for the implementations.

Then in Section 4.3 we formally prove knowledge-soundness, zero-knowledge and completeness for the non-interactive variants. While zero-knowledge and completeness are almost directly inherited from the interactive versions the soundness property requires a much more involved discussion. The notion of soundness itself is of a different nature when dealing with non-interactive proofs.

In order to be able to apply known security reductions we start in Section 4.3.1 with a technical proof showing that we can assume, without loss of generality, that all challenges from an interactive proof with multiple moves come from the same challenge set. Recall it is not our case, as first challenges are integers from $C_1 := \mathbb{Z}_q$ and second challenges are bits from $C_2 := \{0, 1\}$, so we have to see how can we assume that challenges are just random seeds from a single challenge set $\mathcal{S}$ from which the integers and bits are then obtained via pseudorandom functions.

Once we have a single challenge set, we formally obtain provable soundness applying existing Fiat-Shamir theorems considering both the ROM and the *Quantum Random Oracle Model* (QROM) in Section 4.3.2. We furthermore see in Section 4.3.3 what conditions should we impose if instead of provable security we settle choosing parameters so that the current best known attack strategy is still essentially unsuccessful. The probability of these attacks is bounded in this section, while the design of the explicit attacks is sketched here and later included in Appendix 4.A.

Finally, Sections 4.3.4 and 4.3.5 address the zero-knowledge and completeness properties so that we have a full security proof.

The implementation of the commitment scheme and the proofs has been a joint work with Sergi Rovira, it is available in a public GitHub repository [90], and the details about the implementation design paired with explanations about how to deal with the main computational tasks are included in Section 4.4.

As important as the security proofs and the actual implementation is the instantiation of the scheme, given in Section 4.5. By that we mean the process of finding a secure set of parameters as efficient as possible from which we can define a specific instance of the commitment scheme and the companion proofs. It is not a direct task considering that this kind of schemes have many conditions on the parameters (as the ones already defined in Chapter 3 and the new ones we are going to introduce in Section 4.3), that make the parameter space difficult to visualize. It is even harder considering that for some of the conditions we do not even have an explicit expression, and we are limited to use a black box approach as we do when evaluating the difficulty of the RLWE problem using the Lattice Estimator tool from Albrecht *et al.* [8].

We remark here that asymptotic analyses might fail for this task, because the intermingled relations might impose that secure sets of parameters only exist for parameters in a given interval (we are going to see that is what happens with the modulus q for a given n, as we are only able to find secure sets with q above and below certain thresholds). For example, any property that holds provided that q is sufficiently large could imply that no secure instantiations exist because it might be the case that q has to be such large that some other condition cannot be satisfied. For this reason we need tight analyses on the security conditions and robust strategies to find the *best* parameter sets (the script we have used to find these parameters is again uploaded to the GitHub repository [90] and also included in Appendix 4.B).

Ultimately, we present in Section 4.6 multiple benchmarks regarding both running time and size of the commitment, opening, linear and multiplicative zero-knowledge proofs. Thanks to the fact that we have meticulously studied the security conditions and instantiation procedure we can be confident that the presented plots provide insightful information about the efficiency of Stern-based lattice-based NIZKPoKs. Moreover, we also include several tables with a selected number of parameter sets to provide specific results showcasing the best sets regarding size of both the commitment and the proofs, running time of each algorithm and commitment size to message size ratio. We believe these extensive benchmarks are much more comprehensive than what could be extrapolated from a table with only two or three example parameter sets. Additional results for non-practical sets of parameters are also included in Appendix 4.C for completeness.

4.2 Non-Interactive Zero-Knowledge Proofs of Knowledge

In this section, we describe the implementation approach and the small modifications we have introduced while adapting the interactive protocols from Chapter 3 into non-interactive proof systems. Moreover, in Section 4.2.1 we have included the full description of each of the proofs and verifications, taking into account the tools that we have used to instantiate them.

Transform the Interactive protocols into Non-Interactive ones. We have transformed, via Fiat-Shamir, the original Interactive-ZKPoKs into NIZKPoKs, computing the challenges from a CSPRNG seeded with the previous elements from the conversation. To make the non-interactive version knowledge sound with a negligible soundness error, the number of repetitions of the protocols, denoted by δ_{OL} in the opening and linear proofs and by δ_M in the multiplicative one, has to be increased. In particular, we have to explicitly work with parallel repetitions and denote the dependency on the iteration index, something we have omitted till now, because it is of paramount importance that the challenges depend on previous elements from all the parallel repetitions.

Split the key generation algorithm. The standard theoretic definition of cryptographic primitives usually assumes the generator Gen to take as input just the security parameter in unary 1^λ. A commitment scheme was defined that way in Definition 1.6, and we followed that approach in Section 3.3.1. However, the public parameters pp that define the sizes, lattice dimension, modulus or noise distribution play a different role than the public key $pk := (\boldsymbol{a}, \boldsymbol{b}) \in R_q^k$.

From a practical point of view these are two different tasks. Choosing a set of parameters pp usually involves more decisions than just the security level defined by 1^λ. It is often the case that one can fiddle with the parameters preserving the security level to obtain different trade-offs. We are going to show in Section 4.5 that we are free to choose the (λ, n, q, d) that better fit our needs, and the rest of the parameters can be computed so that performance of both the commitment and the proofs is optimized. A parameter generator script is available in the GitHub repository [90], and the relevant part of the code is included in Appendix 4.B.

Nevertheless, the final user typically uses a precomputed set of parameters and just generates the key. For that reason we define a specific key generator KeyGen algorithm that already takes the set of parameters as input and outputs the public key (including the related auxiliary elements discussed before).

$$pk \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(1^\lambda, pp)$$

This approach does not compromise security because the properties that the parameters have to satisfy in order to ensure it are publicly verifiable and do not depend on the specific procedure that has been used to obtain them.

On the contrary, the actual public key $(\boldsymbol{a}, \boldsymbol{b})$ could be adversarially generated to break the security, as the reductions assume it has been obtained from a uniform distribution. For that matter its generation is defined as a separated algorithm, allowing the final users to reuse the pp but resample the pk as many times as they want rerunning KeyGen, so that they trust the keys have been honestly generated.

Permutation encoding. In order to save space the prover does not encode the permutation itself but a seed from which it is pseudorandomly defined. Furthermore, we choose to define just one master seed for every iteration from which all the other seeds for this iteration are derived, using SHAKE-128 as XOF. The procedure used to expand the random seed into a pseudorandomly uniform permutation follows the Fisher-Yates algorithm and is going to be detailed in Section 4.4.2.

Auxiliary commitment instantiation. To instantiate the auxiliary commitment scheme we have chosen to follow the approach presented in Section 1.2.5, using SHA3-256 as a hash function modeled as a random oracle, computing $\mathsf{aCom}(m, o)$ as SHA3-256($m \| o$), with $o \leftarrow_{\text{R}} \{0, 1\}^{\lambda'}$, where $\lambda' := 8 \cdot \lceil \lambda/8 \rceil$ for convenience, so that the opening has an integer number of bytes.

4.2.1 Pseudocode

Before detailing the protocols, we require extra notation and auxiliary functions. Let Hash and XOF be a hash function and an extendable output function, respectively, as defined in Definitions 1.12 and 1.13. Let π_τ be a function that takes as input a vector of integers and permutes its elements using a pseudorandom permutation derived from the seed $\tau \in \{0, 1\}^{8\lceil \lambda/8 \rceil}$.

We will call $\tau_i \in \{0, 1\}^{8\lceil \lambda/8 \rceil}$ the master seed for the ith iteration of the δ repetitions and expand it into $\log(B) + 1$ seeds $\tau_{ij} \in \{0, 1\}^{8\lceil \lambda/8 \rceil}$ using XOF.

We denote by PRN a function that takes as input a seed and outputs vectors of δ pseudorandom integers from $\mathbb{Z}_q$ (or 2δ integers in the multiplicative case where we need twice as many challenges). Analogously, we denote by PRB a function that takes as input a seed and outputs δ pseudorandom bits.

We include here the pseudocode for the NIZKPoKs, even if it heavily replicates the interactive versions, to have them as a reference for a reader interested in understanding the code published in the repository [90] or developing its own implementation of a similar proof. The structure is however different from the interactive versions, as we now have separated algorithms for the prover $\mathcal{P}$ creating the whole proof on its own and for $\mathcal{V}$ verifying its validity without the need of any interaction. In order to facilitate the comparisons with the interactive versions we have kept right aligned the pseudorandom deterministic computation of the challenges via a XOF and a pseudorandom generator (but it is only a visual choice and these steps are still computed by the prover). We have also make explicit how the permutations are also computed pseudorandomly from a seed that has to be sampled by $\mathcal{P}$. Finally, we have used a hash function as auxiliary commitment, and we explicitly include when the openings should be sampled.

Algorithms 4.1, 4.3 and 4.4 and Algorithms 4.2, 4.5 and 4.6 respectively show the tasks of $\mathcal{P}$ and $\mathcal{V}$ to create and verify NIZKPoKs of a valid opening to commitment $\boldsymbol{c}$ with public key $(\boldsymbol{a}, \boldsymbol{b})$, a linear or a multiplicative relation (that is, the committed messages m_1, m_2 and m_3 from commitments $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ either satisfy $m_3 = \lambda_1 m_1 + \lambda_2 m_2$ for given $\lambda_1, \lambda_2 \in R_q$ or $m_3 = m_1 \cdot m_2$).

Algorithm 4.1 NON-INTERACTIVE PROOF OF A VALID OPENING

```
1:  $\{\boldsymbol{e}'_j\}_j \leftarrow \mathsf{expand}(\boldsymbol{e})$
2:  **for** $i \in 1, \dots, \delta_{OL}$ **do**
3:      $\tau_i \leftarrow \{0,1\}^{8\lceil \lambda/8 \rceil}$
4:      $\{\tau_{ij}\}_j \leftarrow \mathsf{XOF}(\tau_i)$
5:      $\{\boldsymbol{f}_{ij}\}_j \leftarrow_{\text{R}} \mathbb{Z}_q^{2nk}$
6:      $\mu_i, \rho_i \leftarrow_{\text{R}} R_q$
7:      $o_{1i}, o_{2i} \leftarrow_{\text{R}} \{0,1\}^{8\lceil \lambda/8 \rceil}$
8:      $\boldsymbol{y}_i := \boldsymbol{a}\mu_i + \boldsymbol{b}\rho_i + \phi(\sum_j 2^j \boldsymbol{f}_{ij})$
9:      $c_{1i} \leftarrow \mathsf{Hash}(\tau_i \| \boldsymbol{y}_i \| o_{1i})$
10:     $c_{2i} \leftarrow \mathsf{Hash}(\{\pi_{\tau_{ij}}(\boldsymbol{f}_{ij})\}_j \| \{\pi_{\tau_{ij}}(\boldsymbol{e}'_j)\}_j \| o_{2i})$

11:                                      $\text{seed}_1 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\boldsymbol{c}\|\{c_{1i}, c_{2i}\}_i)$
12:                                      $\{\alpha_i\}_i \leftarrow \mathsf{PRN}(\text{seed}_1)$

13: **for** $i \in 1, \dots, \delta_{OL}$ **do**
14:     **for** $j \in 0, \dots, \log(B)$ **do**
15:         $\boldsymbol{g}_{ij} := \pi_{\tau_{ij}}(\boldsymbol{f}_{ij} + \alpha_i \boldsymbol{e}'_j)$

16:                                      $\text{seed}_2 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\boldsymbol{c}\|\text{seed}_1\|\{\boldsymbol{g}_{ij}\}_{ij})$
17:                                      $\{b_i\}_i \leftarrow \mathsf{PRB}(\text{seed}_2)$

18: **for** $i \in 1, \dots, \delta_{OL}$ **do**
19:     **if** $b_i = 0$ **then**
20:         $s_i := \rho_i + \alpha_i r$
21:     **else if** $b_i = 1$ **then**
22:         **for** $j \in 0, \dots, \log(B)$ **do**
23:             $\widetilde{\boldsymbol{e}}'_{ij} := \pi_{\tau_{ij}}(\boldsymbol{e}'_j)$
24: **return** $\{c_{i1}, c_{i2}\}_i, \{\boldsymbol{g}_{ij}\}_{ij}, \{(\tau_i, \boldsymbol{y}_i, s_i, o_{1i})\}_{i \text{ s.t. } b_i=0}, \{(\{\widetilde{\boldsymbol{e}}'_{ij}\}_j, o_{2i})\}_{i \text{ s.t. } b_i=1}$
```

Algorithm 4.2 NON-INTERACTIVE VERIFICATION OF A VALID OPENING

```
1:  $\text{seed}_1 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\boldsymbol{c}\|\{c_{1i}, c_{2i}\}_i)$
2:  $\text{seed}_2 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\boldsymbol{c}\|\text{seed}_1\|\{\boldsymbol{g}_{ij}\}_{ij})$
3:  $\{\alpha_i\}_i \leftarrow \mathsf{PRN}(\text{seed}_1),\ \{b_i\}_i \leftarrow \mathsf{PRB}(\text{seed}_2)$
4:  **for** $i \in 1, \dots, \delta_{OL}$ **do**
5:      **if** $b_i = 0$ **then**
6:          $\{\tau_{ij}\}_j \leftarrow \mathsf{XOF}(\tau_i)$
7:          $c_{1i} \overset{?}{=} \mathsf{Hash}(\tau_i \| \boldsymbol{y}_i \| o_{1i})$
8:          $\boldsymbol{z}_i := \boldsymbol{y}_i + \alpha_i(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s_i - \phi(\sum_j 2^j \pi_{\tau_{ij}}^{-1}(\boldsymbol{g}_{ij}))$
9:          $\boldsymbol{z}_i \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
10:     **else if** $b_i = 1$ **then**
11:         $c_{2i} \overset{?}{=} \mathsf{Hash}\big(\{\boldsymbol{g}_{ij} - \alpha_i \widetilde{\boldsymbol{e}}'_{ij}\}_j \| \{\widetilde{\boldsymbol{e}}'_{ij}\}_j \| o_{2i}\big)$
12:         **for** $j \in 0, \dots, \log(B)$ **do**
13:             $\widetilde{\boldsymbol{e}}'_{ij} \overset{?}{\in} \mathfrak{B}_{nk}$
```

Algorithm 4.3 Non-Interactive Proof of a Linear Relation

```
1:  for $h \in 1,2,3$ do
2:      $\{e'_{hj}\}_j \leftarrow \mathsf{expand}(e_h)$
3:  for $i \in 1,\ldots,\delta_{OL}$ do
4:      for $h \in 1,2,3$ do
5:          $\tau_{hi} \leftarrow \{0,1\}^{8\lceil\lambda/8\rceil}$
6:          $\{\tau_{hij}\}_j \leftarrow \mathsf{XOF}(\tau_{hi})$
7:          $\{f_{hij}\}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$
8:      $\mu_{1i}, \mu_{2i}, \{\rho_{hi}\}_h \leftarrow_{\mathrm{R}} R_q$
9:      $\mu_{3i} := \lambda_1\mu_{1i} + \lambda_2\mu_{2i}$
10:     $o_{1i}, o_{2i} \leftarrow_{\mathrm{R}} \{0,1\}^{8\lceil\lambda/8\rceil}$
11:     for $h \in 1,2,3$ do
12:         $y_{hi} := a\mu_{hi} + b\rho_{hi} + \phi(\sum_j 2^j f_{hij})$
13:     $c_{1i} \leftarrow \mathsf{Hash}(\{\tau_{hi}\}_h \| \{y_{hi}\}_h \| o_{1i})$
14:     $c_{2i} \leftarrow \mathsf{Hash}(\{\pi_{\tau_{hij}}(f_{hij})\}_{hj} \| \{\pi_{\tau_{hij}}(e'_{hj})\}_{hj} \| o_{2i})$

15:                              $\mathrm{seed}_1 \leftarrow \mathsf{XOF}(a\|b\|\{c_h\}_h\|\lambda_1\|\lambda_2\|\{c_{1i}, c_{2i}\}_i)$
16:                                                       $\{\alpha_i\}_i \leftarrow \mathsf{PRN}(\mathrm{seed}_1)$

17: for $i \in 1,\ldots,\delta_{OL}$ do
18:     for $h \in 1,2,3$ do
19:         for $j \in 0,\ldots,\log(B)$ do
20:             $g_{hij} := \pi_{\tau_{hij}}(f_{hij} + \alpha_i e'_{hj})$

21:                    $\mathrm{seed}_2 \leftarrow \mathsf{XOF}\left(a\|b\|\{c_h\}_h\|\lambda_1\|\lambda_2\|\mathrm{seed}_1\|\{g_{hij}\}_{hij}\right)$
22:                                                       $\{b_i\}_i \leftarrow \mathsf{PRB}(\mathrm{seed}_2)$

23: for $i \in 1,\ldots,\delta_{OL}$ do
24:     if $b_i = 0$ then
25:         for $h \in 1,2,3$ do
26:             $s_{hi} := \rho_{hi} + \alpha_i r_{hi}$
27:     else if $b_i = 1$ then
28:         for $h \in 1,2,3$ do
29:             for $j \in 0,\ldots,\log(B)$ do
30:                 $\widetilde{e}'_{hij} := \pi_{\tau_{hij}}(e'_{hj})$
31: return $\{c_{1i}, c_{2i}\}_i, \{g_{hij}\}_{hij}$,
        $\left\{\{\tau_{hi}\}_h, \{y_{hi}\}_h, \{s_{hi}\}_h, o_{1i}\right\}_{i \text{ s.t. } b_i=0}, \left\{\{\widetilde{e}'_{hij}\}_{hj}, o_{2i}\right\}_{i \text{ s.t. } b_i=1}$
```

Algorithm 4.4 Non-Interactive Proof of a Multiplicative Relation

```
1: for $h \in 1,2,3$ do
2:     $\{e'_{hj}\}_j \leftarrow \mathsf{expand}(e_h)$
3: for $i \in 1,\dots,\delta_M$ do
4:     for $h \in 1,2,3$ do
5:         $\tau_{hi} \leftarrow \{0,1\}^{8\lceil\lambda/8\rceil}$
6:         $\{\tau_{hij}\}_j \leftarrow \mathsf{XOF}(\tau_{hi})$
7:         $\{f_{hij}\}_j \leftarrow_{\mathrm{R}} \mathbb{Z}_q^{2nk}$
8:         $\mu_{hi}, \rho_{hi} \leftarrow_{\mathrm{R}} R_q$
9:         $y_{hi} := a\mu_{hi} + b\rho_{hi} + \phi(\sum_j 2^j f_{hij})$
10:    $\mu_{\times i}, \mu_{+i} \leftarrow_{\mathrm{R}} R_q$
11:    $m_{\times i} := \mu_{1i}\mu_{2i}, \quad m_{+i} := \mu_{1i}m_2 + \mu_{2i}m_1$
12:    $o_{1i}, o_{2i}, o_{3i}, o_{4i}, o_{5i} \leftarrow_{\mathrm{R}} \{0,1\}^{8\lceil\lambda/8\rceil}$
13:    $c_{1i} \leftarrow \mathsf{Hash}(\{\tau_{hi}\}_h \| \{y_{hi}\}_h \| o_{1i})$
14:    $c_{2i} \leftarrow \mathsf{Hash}(\mu_{3i} \| \mu_{\times i} \| \mu_{+i} \| o_{2i})$
15:    $c_{3i} \leftarrow \mathsf{Hash}(\{\pi_{\tau_{hij}}(f_{hij})\}_{hj} \| \{\pi_{\tau_{hij}}(e'_{hj})\}_{hj} \| o_{3i})$
16:    $c_{4i} \leftarrow \mathsf{Hash}(\mu_{\times i} + m_{\times i} \| \mu_{+i} + m_{+i} \| o_{4i})$
17:                                  $\mathrm{seed}_1 \leftarrow \mathsf{XOF}(a \| b \| \{c_h\}_h \| \{c_{1i}, c_{2i}, c_{3i}, c_{4i}\}_i)$
18:                                  $\{\alpha_i, \beta_i\}_i \leftarrow \mathsf{PRN}(\mathrm{seed}_1)$
19: for $i \in 1,\dots,\delta_M$ do
20:    $\gamma_{1i} := \alpha_i, \quad \gamma_{2i} := \alpha_i, \quad \gamma_{3i} := \beta_i$
21:    $c_{i5} \leftarrow \mathsf{Hash}(\beta\mu_{\times i} + \alpha_i\beta_i\mu_{+i} + {\alpha_i}^2\mu_{3i} \| o_{i5})$
22:    for $h \in 1,2,3$ do
23:        for $j \in 0,\dots,\log(B)$ do
24:            $g_{hij} := \pi_{\tau_{hij}}(f_{hij} + \gamma_{hi}e'_{hj})$
25:                                  $\mathrm{seed}_2 \leftarrow \mathsf{XOF}\left(a \| b \| \{c_h\}_h \| \mathrm{seed}_1 \| \{c_{5i}\}_i \| \{g_{hij}\}_{hij}\right)$
26:                                  $\{b_i\}_i \leftarrow \mathsf{PRB}(\mathrm{seed}_2)$
27: for $i \in 1,\dots,\delta_M$ do
28:    if $b_i = 0$ then
29:        $t_{\times i} := \mu_{\times i} + m_{\times i}, \quad t_{+i} := \mu_{+i} + m_{+i}$
30:        for $h \in 1,2,3$ do
31:            $s_{hi} := \rho_{hi} + \gamma_{hi}r_h$
32:    else if $b_i = 1$ then
33:        for $h \in 1,2,3$ do
34:            for $j \in 0,\dots,\log(B)$ do
35:                $\widetilde{e}'_{hij} := \pi_{\tau_{hij}}(e'_{hj})$
36: return $\{c_{1i}, c_{2i}, c_{3i}, c_{4i}, c_{5i}\}_i, \{g_{hij}\}_{hij},$
        $\left\{(\{\tau_{hi}\}_h, \{y_{hi}\}_h, t_{\times i}, t_{+i}, \{s_{hi}\}_h, o_{1i}, o_{4i}, o_{5i})\right\}_{i \text{ s.t. } b_i=0}$
        $\left\{(\{\widetilde{e}'_{hij}\}_{hj}, \mu_{3i}, \mu_{\times i}, \mu_{+i}, o_{2i}, o_{3i}, o_{5i})\right\}_{i \text{ s.t. } b_i=1}$
```

Algorithm 4.5 Non-Interactive Verification of a Linear Relation

```
1: $\text{seed}_1 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\{\boldsymbol{c}_h\}_h\|\lambda_1\|\lambda_2\|\{c_{1i}, c_{2i}\}_i)$
2: $\text{seed}_2 \leftarrow \mathsf{XOF}\left(\boldsymbol{a}\|\boldsymbol{b}\|\{\boldsymbol{c}_h\}_h\|\lambda_1\|\lambda_2\|\text{seed}_1\|\{\boldsymbol{g}_{hij}\}_{hij}\right)$
3: $\{\alpha_i\}_i \leftarrow \mathsf{PRN}(\text{seed}_1),\ \{b_i\}_i \leftarrow \mathsf{PRB}(\text{seed}_2)$
4: for $i \in 1, \dots, \delta_{OL}$ do
5:     if $b_i = 0$ then
6:         for $h \in 1, 2, 3$ do
7:             $\{\tau_{hij}\}_j \leftarrow \mathsf{XOF}(\tau_{hi})$
8:             $\boldsymbol{z}_{hi} := \boldsymbol{y}_{hi} + \alpha_i(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_{hi} - \phi(\sum_j 2^j \pi^{-1}_{\tau_{hij}}(\boldsymbol{g}_{hij}))$
9:             $\boldsymbol{z}_{hi} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
10:            $c_{1i} \overset{?}{=} \mathsf{Hash}\left(\{\tau_{hi}\}_h\|\{\boldsymbol{y}_{hi}\}_h\|o_{1i}\right)$
11:            $\boldsymbol{z}_{3i} \overset{?}{=} \lambda_1\boldsymbol{z}_{1i} + \lambda_2\boldsymbol{z}_{2i}$
12:    else if $b_i = 1$ then
13:        $c_{2i} \overset{?}{=} \mathsf{Hash}(\{\boldsymbol{g}_{hij} - \alpha_i\widetilde{\boldsymbol{e}}'_{hij}\}_{hj}\|\{\widetilde{\boldsymbol{e}}'_{hij}\}_{hj}\|o_{2i})$
14:        for $h \in 1, 2, 3$ do
15:            for $j \in 0, \dots, \log(B)$ do
16:                $\widetilde{\boldsymbol{e}}'_{hij} \overset{?}{\in} \mathfrak{B}_{nk}$
```

Algorithm 4.6 Non-Interactive Verification of a Multiplicative Relation

```
1: $\text{seed}_1 \leftarrow \mathsf{XOF}(\boldsymbol{a}\|\boldsymbol{b}\|\{\boldsymbol{c}_h\}_h\|\{c_{1i}, c_{2i}, c_{3i}, c_{4i}\}_i)$
2: $\text{seed}_2 \leftarrow \mathsf{XOF}\left(\boldsymbol{a}\|\boldsymbol{b}\|\{\boldsymbol{c}_h\}_h\|\text{seed}_1\|\{c_{5i}\}_i\|\{\boldsymbol{g}_{hij}\}_{hij}\right)$
3: $\{\alpha_i, \beta_i\}_i \leftarrow \mathsf{PRN}(\text{seed}_1),\ \{b_i\}_i \leftarrow \mathsf{PRB}(\text{seed}_2)$
4: for $i \in 1, \dots, \delta_M$ do
5:     $\gamma_{1i} := \alpha_i, \quad \gamma_{2i} := \alpha_i, \quad \gamma_{3i} := \beta_i$
6:     if $b_i = 0$ then
7:         for $h \in 1, 2, 3$ do
8:             $\boldsymbol{z}_{hi} := \boldsymbol{y}_{hi} + \gamma_{hi}(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_{hi} - \phi(\sum_j 2^j \pi^{-1}_{\tau_{hij}}(\boldsymbol{g}_{hij}))$
9:             $\boldsymbol{z}_{hi} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
10:            Let $t_{hi} \in R_q$ s.t. $\boldsymbol{z}_{hi} = \boldsymbol{a}t_{hi}$
11:        $c_{1i} \overset{?}{=} \mathsf{Hash}(\{\tau_{hi}\}_h\|\{\boldsymbol{y}_{hi}\}_h\|o_{1i})$
12:        $c_{4i} \overset{?}{=} \mathsf{Hash}(t_{\times i}\|t_{+i}\|o_{4i})$
13:        $c_{5i} \overset{?}{=} \mathsf{Hash}(\beta_i t_{\times i} + \alpha_i\beta_i t_{+i} + {\alpha_i}^2 t_{3i} - \beta_i t_{1i} t_{2i}\|o_{5i})$
14:    else if $b_i = 1$ then
15:        $c_{2i} \overset{?}{=} \mathsf{Hash}(\mu_{3i}\|\mu_{\times i}\|\mu_{+i}\|o_{2i})$
16:        $c_{3i} \overset{?}{=} \mathsf{Hash}(\{\boldsymbol{g}_{hij} - \gamma_{hi}\widetilde{\boldsymbol{e}}'_{hij}\}_{hj}\|\{\widetilde{\boldsymbol{e}}'_{hij}\}_{hj}\|o_{3i})$
17:        $c_{5i} \overset{?}{=} \mathsf{Hash}(\beta_i\mu_{\times i} + \alpha_i\beta_i\mu_{+i} + {\alpha_i}^2\mu_{3i}\|o_{5i})$
18:        for $h \in 1, 2, 3$ do
19:            for $j \in 0, \dots, \log(B)$ do
20:                $\widetilde{\boldsymbol{e}}'_{hij} \overset{?}{\in} \mathfrak{B}_{nk}$
```

4.3 Non-Interactive Security Proofs

In this section we analyze and prove the security properties of the NIZKPoKs. The conditions defined in Chapter 3 for their interactive counterparts are inherited, but the (knowledge) soundness needs to be carefully revisited because of the different nature of this property when dealing with non-interactive proof systems.

We were able to bound the knowledge error for a single interaction of the opening and linear protocols with $\kappa_{OL} = (q+1)/2q$, and also for the multiplicative protocol with $\kappa_M = (q^2+3q-2)/2q^2$. As we also know the knowledge error of our specific protocols enjoys strong parallel repetition and decreases as κ^δ with δ parallel repetitions, in order to target a security level λ we just need to ensure that the respective $\kappa^\delta \leq 2^{-\lambda}$.

- $$\left(\frac{q+1}{2q}\right)^{\delta_{OL}} \leq 2^{-\lambda}. \tag{4.1}$$
- $$\left(\frac{q^2+3q-2}{2q^2}\right)^{\delta_M} \leq 2^{-\lambda}. \tag{4.2}$$

However, we can not obtain these unconditional hard bounds on the success probability of an adversary when dealing with NIZKPoKs that come from applying the Fiat-Shamir transform to an interactive ZKPoK. The intuition behind the transform tells us that the challenge obtained via a hash function, modeled as a random oracle, is as unpredictable as the challenge an honest verifier would send in the interactive version. The formal security proofs, using variants of the Forking Lemma [98], rewind the adversary and reprogram the oracle to obtain several proofs from which the witness can be extracted. Nevertheless, it is not equivalent to the interactive version, because a malicious prover failing during the interaction is immediately discovered, but a prover failing to produce a valid non-interactive proof can just try again to increase its success probability as many times as they want. For that matter it is only possible to obtain a similar result bounding the success probability of adversaries with a function that depends on the number of oracle queries Q. It is what we obtain from the formal proofs (the extractor has to guess which query to reprogram), and takes into account the computational power of the attacker because new (or more expensive) attempts increase the success probability but require new oracle queries.

The security of the Fiat-Shamir transform and the soundness of the derived non-interactive proof systems have been extensively studied for Σ-protocols (recall, only 3 rounds and negligible soundness error with a single iteration). The case we are interested in, with multiple moves and many parallel repetitions is much more delicate.

Furthermore, the same way we are considering post-quantum assumptions to build schemes that are secure against quantum capable adversaries we might also work with the QROM, allowing the adversary to submit oracle queries in quantum superposition.

In any case the existing security proofs for multiple rounds usually consider for simplicity that all challenges belong to the same space, and therefore there is only a single oracle. That is not the case with our protocols, as the first challenges are integer and the second ones are bits. However, provided that in the non-interactive version the challenges are computed using a pseudorandom generator seeded with the output of a XOF that takes as input the previous elements we can also interpret each NIZKPoK as the transform of an alternative interactive version of the protocols where the verifier always challenges with just a random seed from $\mathcal{S} \subseteq \{0,1\}^*$. The challenges from the original version are now computed by the prover on its own using the pseudorandom generator with these seeds. We call this a single challenge set variant, as opposed to the original multiple challenge sets version. There exist detailed proofs of the soundness preservation of Fiat-Shamir for the single challenge set scenario while the interactive soundness is easier to analyze in the multiple challenge sets case. We explain this more deeply in Section 4.3.1 and formally show that the soundness is preserved too for this alternative variant (a result we believe is of independent interest), provided that the seed space $\mathcal{S}$ is large enough.

- $\left|\mathcal{S}\right| \geq 2^{\lambda} q^{\delta}.$ (4.3)

After these preliminaries we devote Section 4.3.2 to ensure that the Fiat-Shamir transform allows us to satisfy the standard notion of soundness if δ is large enough. The conditions imposed by the security reduction imply a large overhead. Fiat-Shamir ensures soundness and knowledge-soundness are preserved but with a security loss of $\mathcal{O}(Q^{2\mu})$ or $\mathcal{O}(Q^{\mu})$ depending on whether we consider it in the QROM or in the plain ROM respectively, where Q is the number of oracle queries made by the adversary and $2\mu + 1$ is the number of moves. To get provable security, one should increase δ_{OL} and δ_M up to the order of 3λ or 5λ so that $\kappa^{\delta} \approx 2^{-3\lambda}$ or $\kappa^{\delta} \approx 2^{-5\lambda}$ to compensate for the respective ROM or QROM security loss.

This upper bound on the security loss is so large that it is usually considered that, while tight in general (as seen with some pathologic examples [13]), it is an overkill for practical protocols. For that reason, some proposals assume it is actually milder (and sometimes it is completely disregarded) for practical protocols [39]. On the one hand, we believe that is a risky move from a practical point of view, as someone implementing such proposals could significantly overestimate its security. On the other hand, even if the proposal is only a theoretical study and no unsecure instantiation is going to be used in the wild we still think the study would be considerably incomplete without addressing the actual security loss, as comparisons with other proposals would then turn spurious because estimated sizes and running times might be largely underestimated without the overhead needed to compensate for such security loss.

In order to take care of these issues, besides providing the theoretical analysis, we also follow a heuristic approach and study the existing attack strategies. We explicitly show how our scheme would be vulnerable to a certain extent to one of them, and provide the additional condition on the number of repetitions necessary to compensate it and still ensure that the success probability of the attack is below the targeted $2^{-\lambda}$ threshold in Section 4.3.3. This new condition turns out to be much milder than the overhead necessary for provable security.

- $$(2\lambda - 1)\log\left(\frac{2\lambda - 1}{\delta(1 - q^{-1})}\right) + (\delta - 2\lambda + 1)\log\left(\frac{\delta - 2\lambda + 1}{\delta q^{-1}}\right) \geq 2\lambda. \tag{4.4}$$

Finally, in Sections 4.3.4 and 4.3.5 we briefly discuss zero-knowledge and completeness of the non-interactive proof systems.

4.3.1 Multiple vs. Single Challenge Set

Notice that our notion of a public-coin interactive protocol assumes that in each move the challenge is uniformly drawn from a possibly different challenge set C_i. Some articles proving security of the Fiat-Shamir transform, as [13] or [45], assume for convenience that all challenges are drawn from a single universal challenge set C. In [13] the authors mention that the proof of their main result (which unfortunately does not apply to our protocols) could be rewritten to admit arbitrary challenge sets. We think that a detailed general proof of why this assumption of having a single challenge set can be taken without any loss of generality in the ROM is of independent interest, and it allows us to use any transformation regardless of their challenge sets convention without any extra work.

Theorem 4.1 (Single vs. Multiple Challenge sets). *Any public-coin $(2\mu + 1)$-move Honest Verifier Zero-Knowledge Proof of Knowledge between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where challenge c_i in the ith round is uniformly sampled by $\mathcal{V}$ from a challenge set C_i, can be transformed (in the ROM) into another public-coin $(2\mu + 1)$-move Honest Verifier Zero-Knowledge Proof of Knowledge between another prover $\mathcal{P}'$ and another verifier $\mathcal{V}'$ where every challenge s_i is uniformly sampled from a sufficiently large single challenge set $\mathcal{S}$ preserving completeness, zero-knowledge, soundness and knowledge-soundness.*

That new protocol is defined in the following way. $\mathcal{P}'$ computes its initial commitment a as $\mathcal{P}$ would do and sends it to $\mathcal{V}'$. In each round i the verifier $\mathcal{V}'$ samples a seed uniformly at random from the challenge set $s_i \leftarrow_R \mathcal{S}$, and the prover $\mathcal{P}'$ then calls a random oracle O_i that outputs uniformly random elements c_i from C_i if the seed had not been called before or the previous response otherwise. Then $\mathcal{P}'$ answers b_i in the same way $\mathcal{P}$ would do if challenged with that c_i. Finally, $\mathcal{V}'$ checks the verifying equations as $\mathcal{V}$ would do with the challenges $c_i \leftarrow O_i(s_i)$.

Proof. It is clear that completeness is directly inherited from the original protocol. Honest verifier zero-knowledge is also preserved, as one can compute the c_i calling the oracles with the s_i and then use the original simulator. Provided that the oracles output uniformly random c_i and after that point everything is computed in the same manner, the simulated conversations would again follow the same distributions as the ones between honest $\mathcal{P}'$ and $\mathcal{V}'$.

The only properties that require a dedicated meticulous analysis are soundness and knowledge-soundness. Let $\mathcal{P}^\circ$ be a malicious prover against the single challenge set protocol with access to μ oracles $\{O_i\}_{i=1}^{\mu}$ and a success probability ϵ. We can also construct a prover $\mathcal{P}^*$ against the multiple challenge protocol, interacting with $\mathcal{P}^\circ$ providing its challenges and simulating their oracle queries. We will refer to s_i as the inner challenges and the corresponding $c_i \leftarrow O_i(s_i)$ as the outer challenges. It is possible following the next procedure.

$\mathcal{P}^*$ starts running $\mathcal{P}^\circ$ to produce the first commitment a. Whenever $\mathcal{P}^\circ$ calls oracle O_i with an $s \in \mathcal{S}$ then $\mathcal{P}^*$ samples $c \leftarrow_R C_i$ uniformly at random if s was never queried before or returns the previously sampled or programmed element. To compute the $s_i \in \mathcal{S}$ inner challenge seeds prover $\mathcal{P}^*$ gets challenges c_i from the verifier and outputs with some probability $p_i^{(c_i)}$ a uniformly random seed from the ones that have already been queried to the oracle O_i and had been answered with the outer challenge c_i or otherwise $\mathcal{P}^*$ chooses a uniformly random seed from the ones that have not been asked by $\mathcal{P}^\circ$ to the oracle O_i and program its simulated oracle to subsequently answer c_i to that seed s_i.

We are going to define some disjoint partitions of $\mathcal{S}$, useful to define these probabilities and prove the desired properties. Let $\mathcal{S}_i^c \subseteq \mathcal{S}$ be the subset of seeds that have been oracle called by $\mathcal{P}^\circ$ to O_i and have received c as answer. Let $\mathcal{S}_i^{\neg c} \subseteq \mathcal{S}$ be the subset of seeds that have been oracle called by $\mathcal{P}^\circ$ and have received an answer different from c. And finally, let $\mathcal{S}_i^{\varnothing} \subseteq \mathcal{S}$ be the subset of seeds that have not been queried yet. That way at any time we have $\mathcal{S} = \mathcal{S}_i^c \sqcup \mathcal{S}_i^{\neg c} \sqcup \mathcal{S}_i^{\varnothing}$ for every i and c.

If we fix $p_i^{(c_i)} := (\left|\mathcal{S}_i^{c_i}\right| \cdot |C_i|)/|\mathcal{S}|$ then the success probability of $\mathcal{P}^*$ would be that of $\mathcal{P}^\circ$. Intuitively the expected number of seeds that would be mapped to c_i would be $|\mathcal{S}|/|C_i|$, and we already have $\left|\mathcal{S}_i^c\right|$ many, so this seems the right guess for that probability, and we are going to confirm that it indeed works. Observe this is always a viable strategy if $|S|$ is large enough. On the one hand if $\mathcal{S}_i^c = \varnothing$ then $p_i^{(c)} = 0$, and we never have to sample from an empty set. On the other hand we can ensure that $p_i^{(c)} \leq 1$ if $|\mathcal{S}| \geq Q_i |C_i|$ where Q_i is the number of oracle queries to oracle $\mathcal{O}_i$ and therefore is an upper bound of $|\mathcal{S}_i^c|$. This also guarantees that $\mathcal{S}_i^\varnothing \neq \varnothing$ either.

Reduction protocol 4.7 Multiple from single challenge set adversary.*

$$\begin{array}{lcccr}
\mathcal{P}^\circ & & \mathcal{P}^* & & \mathcal{V} \\ \hline
 & \xleftrightarrow{\mathcal{O}(\cdot)} & & & \\
 & \xrightarrow{a} & & & \\
 & & & \xrightarrow{a} & \\
 & & & & c_1 \leftarrow_{\mathrm{R}} C_1 \\
 & & & \xleftarrow{c_1} & \\
 & & \begin{cases} s_1 \leftarrow_{\mathrm{R}} \mathcal{S}_1^{c_1} & \text{wp } p_1^{(c_1)} \\ s_1 \leftarrow_{\mathrm{R}} \mathcal{S}_1^{\varnothing} & \\ \text{let } \mathcal{O}_1(c_1) = s_1 & \text{o/w} \end{cases} & & \\
 & \xleftarrow{s_1} & & & \\
 & \xleftrightarrow{\mathcal{O}(\cdot)} & & & \\
 & \xrightarrow{b_1} & & & \\
 & & & \xrightarrow{b_1} & \\
 & & \vdots & & \\
 & \xrightarrow{b_\mu} & & & \\
 & & & \xrightarrow{b_\mu} & \\
 & & & & c_\mu \leftarrow_{\mathrm{R}} C_\mu \\
 & & & \xleftarrow{c_\mu} & \\
 & & \begin{cases} s_\mu \leftarrow_{\mathrm{R}} \mathcal{S}_\mu^{c_\mu} & \text{wp } p_\mu^{(c_\mu)} \\ s_\mu \leftarrow_{\mathrm{R}} \mathcal{S}_\mu^{\varnothing} & \\ \text{let } \mathcal{O}_\mu(c_\mu) = s_\mu & \text{o/w} \end{cases} & & \\
 & \xleftarrow{s_\mu} & & & \\
 & \xleftrightarrow{\mathcal{O}(\cdot)} & & & \\
 & \xrightarrow{b_\mu} & & & \\
 & & & \xrightarrow{b_\mu} &
\end{array}$$

To prove that the success probability is the same, we need to check that the challenges s_i and the oracle answers simulated by $\mathcal{P}^*$ follow the same distribution as the ones $\mathcal{P}^\circ$ would receive interacting with an honest verifier and truly random oracles.

Let us analyze the probability of providing $\mathcal{P}^\circ$ with a given s as the ith inner challenge. Either this s has been submitted before to O_i by $\mathcal{P}^\circ$ or not. In the first case, let us denote by c the answer $\mathcal{P}^*$ gave it. The probability of choosing such s is the probability of the ith outer challenge being c multiplied by the probability $p_i^{(c)}$ of choosing s among the $\mathcal{S}_i^c$ already queried seeds that output c multiplied again by the probability of getting that particular seed when sampling uniformly from that set.

$$\begin{aligned}\Pr\left[s_i = s \mid s \in \mathcal{S}_i^c\right] &= \Pr\left[c_i = c\right] \cdot p_i^{(c)} \cdot \Pr\left[s_i = s \mid s_i \leftarrow_{\mathrm{R}} \mathcal{S}_i^c\right] \\ &= \frac{1}{|C_i|} \cdot \frac{\left|\mathcal{S}_i^c\right| \cdot |C_i|}{|\mathcal{S}|} \cdot \frac{1}{\left|\mathcal{S}_i^c\right|} = \frac{1}{|\mathcal{S}|}\end{aligned}$$

On the other hand, if s was not submitted before, i.e. $s \in \mathcal{S}_i^\varnothing$, for each possible challenge c its probability is now $1 - p_i^{(c)}$ multiplied by the probability of choosing that s from $\mathcal{S}_i^\varnothing$.

$$\begin{aligned}\Pr\left[s_i = s \mid s \in \mathcal{S}_i^\varnothing\right] &= \sum_{c \in C_i} \Pr\left[c_i = c\right] \cdot \left(1 - p_i^{(c)}\right) \cdot \Pr\left[s_i = s \mid s_i \leftarrow_{\mathrm{R}} \mathcal{S}_i^\varnothing\right] \\ &= \sum_{c \in C_i} \frac{1}{|C_i|} \cdot \frac{|\mathcal{S}| - \left|\mathcal{S}_i^c\right| \cdot |C_i|}{|\mathcal{S}|} \cdot \frac{1}{\left|\mathcal{S}_i^\varnothing\right|} \\ &= \frac{1}{|\mathcal{S}|} \cdot \frac{|\mathcal{S}| - \sum_{c \in C_i} \left|\mathcal{S}_i^c\right|}{\left|\mathcal{S}_i^\varnothing\right|} = \frac{1}{|\mathcal{S}|} \cdot \frac{\left|\mathcal{S}_i^\varnothing\right|}{\left|\mathcal{S}_i^\varnothing\right|} = \frac{1}{|\mathcal{S}|}\end{aligned}$$

Then, as every s has the same probability, $\mathcal{P}^\circ$ receives challenges uniformly distributed in $\mathcal{S}$ the same way they would have if they were interacting with a real honest verifier.

Regarding its calls to the oracle, every query with an s that has not been set as inner challenge is answered as usual, sampling a uniform c. Every s_i that they have received as inner challenge has already a defined answered c_i. Nevertheless, provided that this c_i has been sent by an honest verifier it also follows a uniformly random distribution as an oracle call would do.

As we have granted that the interaction $\mathcal{P}^*$ has with $\mathcal{P}^\circ$ follows the same distribution as the interaction with a verifier and μ random oracles then the success

*The usual abbreviations *wp* and *o/w* are used for "with probability" and "otherwise".

probability is exactly the same, as we wanted to prove. That implies that the soundness error of the new single-oracle protocol is at most equal to the one from the multi-challenge protocol, because any success probability of the former can also be realized with the latter.

Knowledge-soundness preservation comes from the same transformation, as we can again use the constructed $\mathcal{P}^*$ with the original extractor to obtain a witness with the same probability. □

We can use this proof to ensure the security of a version with a single challenge set, which allows us to continue the proof using the literature that builds their theorems under this assumption. Although the reduction might seem artificial it is not far from the real world, as our own implementation internally uses some seeds to generate the pseudorandom challenges, precisely as described (and therefore can be interpreted as having different challenge sets or a single one just depending on whether we consider as challenge the seed or the corresponding pseudorandom output). This general proof is also easier to apply than redoing the proof of the Fiat-Shamir transform security or redefining $(k_1, \ldots, k_\mu)$-special soundness.

4.3.2 The Fiat-Shamir Transform

With the last modification, we can see that our protocols satisfy the definition of a *Public-Coin Interactive Proof* (PCIP) system as defined in [45], essentially our definition of an interactive proof of knowledge assuming every challenge is sampled from the same set. Now we know this can be assumed without loss of generality we can formally introduce the Fiat-Shamir transform.

Definition 4.2 (Fiat-Shamir transform for general PCIP adapted from [45]). The Fiat-Shamir transform for general PCIP, given by $\langle \mathcal{P}, \mathcal{V} \rangle$, defines a non-interactive proof system where the proof Π for a statement x is generated by a prover $\mathcal{P}_{FS}^H$ by computing the prover messages a and $\{b_i\}_{i=1}^{\mu}$ of the transcript of an interactive protocol with challenges deterministically obtained from the previous elements, $c_1 := H(0, x, a)$ and $c_i := H(i-1, x, c_{i-1}, b_{i-1})$ for $i > 1$, where H is a hash function with the appropriate domain. The verifier $\mathcal{V}_{FS}^H$ accepts $\Pi := (x, a, \{b_i\}_{i=1}^{\mu})$ if the transcript obtained when computing the challenges is accepting (we denote by $\mathcal{V}_{FS}^H(x, \Pi)$ the output of verifier $\mathcal{V}_{FS}^H$ when presented with a proof Π for a statement x).

Remark 4.3. Their original definition does not include the statement x in the c_i challenges with $i > 1$, as it is already implicit because each hash takes as input the previous challenge and the first does have x as part of its input. Nevertheless, as they mention, "any additional strings can be included in the argument when computing c_i using H, without influencing the security properties of the non-interactive proof system in a detrimental way". We add the statement so that this transform, once we include x, also corresponds to the alternative definition proposed in [12].

The model assumes that H behaves as a (Quantum) Random Oracle, and our goal is to prove soundness in that setting. For that reason, in the following definitions and proofs we use $(\mathcal{V}^{O}_{FS}, \mathcal{V}^{O}_{FS})$ even if we implement $(\mathcal{V}^{H}_{FS}, \mathcal{V}^{H}_{FS})$. We first recall that the soundness of non-interactive proof systems can not be directly addressed as before. The success probability of an adversary might unavoidably grow by just trying again if they fail at the first attempt. For that reason, we can only bound its probability when restricting the number of oracle queries Q, ensuring it only grows polynomially with it.

Definition 4.4 (Soundness of a non-interactive proof system $(\mathcal{P}^{H}_{FS}, \mathcal{V}^{H}_{FS})$, adapted from [46])**.** We say that a non-interactive proof system $(\mathcal{P}^{H}_{FS}, \mathcal{V}^{H}_{FS})$ is *sound* if there exists a negligible function $\eta(\lambda)$ and a constant e such that for any (quantum) adversary $\mathcal{A}$ making at most Q queries to a uniformly random O and any $\lambda \in \mathbb{N}$:

$$\Pr_{O}\left[\mathcal{V}^{O}_{FS}(x,\Pi) = \text{accept} \wedge x \notin \Re \;\middle|\; (x,\Pi) \leftarrow_{\text{R}} \mathcal{A}^{O}\right] \leq Q^{e}\eta(\lambda).$$

Notice the dependency on λ is implicit. It works as the security parameter that defines the specifications of the proof system $(\mathcal{P}^{O}_{FS}, \mathcal{V}^{O}_{FS})$, in particular in our case it has an important impact on the number of repetitions.

We can prove our protocols are sound in the QROM using the recent result from [45] that ensures the existence of a quantum algorithm $\mathcal{B}$ such that

$$\begin{aligned}\Pr\Big[x = x^{\circ} \wedge v = \text{accept} \;\Big|\; (x,v) \leftarrow_{\text{R}} \langle \mathcal{B}^{\mathcal{A}}, \mathcal{V}\rangle\Big] &\geq \\ \geq \frac{\mu!}{(2Q+\mu+1)^{2\mu}} \Pr_{O}\left[x = x^{\circ} \wedge \mathcal{V}^{O}_{FS}(x,\Pi) = \text{accept} \;\middle|\; (x,\Pi) \leftarrow_{\text{R}} \mathcal{A}^{O}\right] - \epsilon_{x^{\circ}},\end{aligned}$$

where the additive error term $\epsilon_{x^{\circ}}$ is equal to $\mu!/|C|$ when summed over all x°, and can be made arbitrarily small. Following the notation from [45], $(x,v) \leftarrow_{\text{R}} \langle \mathcal{B}^{\mathcal{A}}, \mathcal{V}\rangle$ means that $\mathcal{B}^{\mathcal{A}}$ first outputs a statement x and then interacts with $\mathcal{V}$ so that $v \leftarrow_{\text{R}} \langle \mathcal{B}^{\mathcal{A}}, \mathcal{V}\rangle(x)$.

As [45] mentions, this implies soundness preservation as long as the challenge space C has size superpolynomial in the security parameter, since μ is constant and Q is polynomial in the security parameter. That makes $\epsilon_{x^{\circ}}$ negligible, and we can also aim to make the bound on the success probability of the interactive version κ^{δ} negligible in λ by choosing large enough δ. There we have the negligible function required by the definition. Then, provided that μ is constant, $(2Q+\mu+1)^{2\mu}/(\mu!)$ can be bounded by some power of Q.

In order to target a provable security level one would just need to choose $|\,C\,|$ and δ large enough so that the $\kappa^{\delta}+\mu!/|\,C\,|$ compensates the $(2Q+\mu+1)^{2\mu}/(\mu!)$ security loss. One could achieve a soundness error smaller than $2^{-\lambda}$ choosing both $\delta \approx \log(|\,C\,|) \approx 5\lambda$, as we can always assume $Q < 2^{\lambda}$ and get $(\kappa^{\delta}+\mu!/|\,C\,|)(2Q+\mu+1)^{2\mu}/(\mu!) \lesssim 2^{-\lambda}$, increasing the proofs by a factor of 5.

This $O(Q^{2\mu})$ security loss is tight in general, as discussed in [45]. The 2 in the exponent comes from the fact that we are considering the QROM, the quantum version of the ROM, and have to account for Grover-search attacks.

Similar proofs in the plain ROM exist with a security loss of just $(Q+1)^{\mu}$, as informally claimed in [13]. In this important work they show that for many interactive proofs, mainly those with $(k_1,\dots,k_{\mu})$-special soundness, one can prove a better bound on the security loss of only $O(\mu Q)$, linear instead of exponential in the number of rounds. However, that is not the case of the considered schemes, as the parallel repetition of a $(k_1,\dots,k_{\mu})$-special sound protocol is not $(k_1,\dots,k_{\mu})$-special sound itself.

Besides soundness, one can similarly prove knowledge-soundness because it is also preserved by Fiat-Shamir.

Definition 4.5 (Knowledge-Soundness of a non-interactive proof system as in [46])**.** The non-interactive proof system $(\mathcal{P}^{O}_{FS}, \mathcal{V}^{O}_{FS})$ is a computational proof of knowledge if there exists a polynomial-time algorithm $\mathcal{E}$, a polynomial $p(\lambda)$, constants $d, e \geq 0$ and a negligible function $\eta(\lambda)$, such that for any (quantum) polynomial-time algorithm $\mathcal{A}$ making at most Q oracle queries, any $\lambda \in \mathbb{N}$ and any statement x we have

$$\Pr\left[(x,w)\in\Re \,\middle|\, w \leftarrow_{\mathrm{R}} \mathcal{E}^{\mathcal{A}}(x)\right] \geq \frac{1}{Q^{e}p(\lambda)} \Pr_{H}\left[\mathcal{V}^{H}_{FS}(x,\Pi) \,\middle|\, \Pi \leftarrow_{\mathrm{R}} \mathcal{A}^{H}\right]^{d} - \eta(\lambda).$$

It can be similarly defined for adaptive adversaries allowing $\mathcal{A}$ to choose x

As [45] mentions in their Corollary 15, the proof of knowledge property is preserved (at the expanse of the $Q^{-2\mu}$ security loss) in the QROM because any dishonest prover against the Fiat-Shamir version with success probability ϵ can be used to get a dishonest prover against the interactive protocol with a success probability of $\epsilon \cdot (2Q+1)^{-2\mu}$, that can be used to extract a witness.

So far we have not mentioned any difference between the opening, linear or multiplicative proofs. For the opening and the linear proofs we know the interactive protocols $\langle \mathcal{P}, \mathcal{V} \rangle$ are sound with soundness error as small as we want. Soundness is preserved if we modify them to get $\langle \mathcal{P}', \mathcal{V}' \rangle$, single challenge versions. And finally soundness is also preserved with a $O(Q^{-2\mu})$ security loss when we apply the

Fiat-Shamir transform to get the non-interactive proof system $\left(\mathcal{P}'^{H}_{FS}, \mathcal{V}'^{H}_{FS}\right)$.

Diagram 4.1 Soundness diagram of the opening and linear relation NIZKPoKs

$$\begin{tikzcd}
\text{Multiple sets} & \text{Single set} & \text{Non-interactive} \\
\langle \mathcal{P}, \mathcal{V} \rangle \arrow[r] & \langle \mathcal{P}', \mathcal{V}' \rangle \arrow[r, "Q^{-4}"] & \left(\mathcal{P}'^{H}_{FS}, \mathcal{V}'^{H}_{FS}\right)
\end{tikzcd}$$

The multiplicative relation proof needs more attention. We proved that the interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ defined in Protocol 3.3 was sound because we could interpret it as a 7-move protocol with $(2,3,2)$-special soundness (let us denote it by $\langle \widehat{\mathcal{P}}, \widehat{\mathcal{V}} \rangle$). However, the number of moves is important if we want to turn them into the single challenge versions, because now $\langle \mathcal{P}', \mathcal{V}' \rangle$ is no longer the same as $\langle \widehat{\mathcal{P}}', \widehat{\mathcal{V}}' \rangle$ (there would be a different number of oracles). And these differences would matter because the security loss of a 7-move protocol would be even bigger than the one of a 5-move protocol.

For these reasons even if we start proving soundness from $\langle \widehat{\mathcal{P}}, \widehat{\mathcal{V}} \rangle$, the proof defined by Algorithms 4.4 and 4.6 is $\left(\mathcal{P}'^{H}_{FS}, \mathcal{V}'^{H}_{FS}\right)$, and it receives its soundness via the 5-moves versions.

Diagram 4.2 Soundness diagram of the multiplicative relation NIZKPoK

$$\begin{tikzcd}
\text{Multiple sets} & \text{Single set} & \text{Non-interactive} \\
\underset{\text{5-move}}{\langle \mathcal{P}, \mathcal{V} \rangle} \arrow[r] \arrow[d, leftrightarrow] & \underset{\text{5-move}}{\langle \mathcal{P}', \mathcal{V}' \rangle} \arrow[r, "Q^{-4}"] \arrow[d, phantom, "\nparallel"] & \underset{\text{from 5-move}}{\left(\mathcal{P}'^{H}_{FS}, \mathcal{V}'^{H}_{FS}\right)} \arrow[d, phantom, "\nparallel"] \\
\underset{\text{7-move}}{\langle \widehat{\mathcal{P}}, \widehat{\mathcal{V}} \rangle} \arrow[r] & \underset{\text{7-move}}{\langle \widehat{\mathcal{P}}', \widehat{\mathcal{V}}' \rangle} \arrow[r, "Q^{-6}"] & \underset{\text{from 7-move}}{\left(\widehat{\mathcal{P}}'^{H}_{FS}, \widehat{\mathcal{V}}'^{H}_{FS}\right)}
\end{tikzcd}$$

The same applies to knowledge-soundness because we have seen that these sequence of transformations also preserves it.

4.3.3 Known Attacks

We have seen so far how the schemes can be instantiated achieving provable security. The large number of parallel repetitions needed to make the interactive protocol have a knowledge error so small that it compensates the security loss of the Fiat-Shamir transform in the QROM (or even in the ROM) would make it far from practical. Notwithstanding that overhead is the only option we are aware of for obtaining provable security, we also want this work to be comparable with existing proposals.

And it is usually the case that the proposed parameters do not consider entirely the Fiat-Shamir security reduction.

Quoting from [13]:

> *If one wants to rely on the proven security reduction, one needs to choose a large security parameter for Π, in order to compensate for the order Q^{μ} security loss, effecting its efficiency; alternatively, one has to give up on proven security and simply assume that the security loss is much milder than what the general bound suggests. Often, the security loss is simply ignored.*

Disregarding additional security losses is called the ϵ^{δ}-heuristic, as one directly assumes the soundness error exponentially decreases with parallel repetitions without considering any possible attacks [65].

However, even if assuming the security loss is milder than the worst case scenario, we should still consider which is the best known attack. For example, the post-quantum signature scheme [39], which has a structure very similar to the one considered through this dissertation as it is based too on a $q2$-identification scheme (a $(2,2)$-special sound protocol with $|C_1| = q$ and $|C_2| = 2$), was successfully attacked in [65]. We remark that the existence of these attacks only implies that larger parameters should be chosen. We can ensure that the Fiat-Shamir transform is not inherently broken because if we sufficiently increase these parameters (number of repetitions and challenge space size) we could obtain provable security. Regarding the studied protocols, as far as we are concerned, the best known attack is the general strategy presented in [12] against the parallel repetition of $(k_1,\dots,k_\mu)$-special sound protocols that satisfy an additional property they call $(l_1,\dots,l_\mu)$-special unsoundness (sometimes $(l_1,\dots,l_\mu)$-out-of-$(|C_1|,\dots,|C_\mu|)$-special unsoundness) with N responses per round. Let us formally introduce that concept, show that the studied protocols satisfy the definition and reanalyze accordingly the security of the non-interactive proofs.

Definition 4.6 ($(l_1,\dots,l_\mu)$-special unsoundness with N potential responses per round as in [13])**.** We say that a public-coin interactive proof $\langle \mathcal{P},\mathcal{V}\rangle$ has $(l_1,\dots,l_\mu)$-*special unsoundness* if there exists a dishonest prover $\mathcal{A}$ such that when interacting with $\mathcal{V}$ on input x the following holds:

- $\mathcal{A}$ starts in *active mode*, meaning that at every round after $\mathcal{A}$'s message there exists a subset $\Gamma_i \subseteq C_i$ of size l_i such that if the challenge $c_i \in \Gamma_i$ then $\mathcal{A}$ switches to *passive mode*.
- If $\mathcal{A}$ switches to *passive mode* then it remains in *passive mode* and the final conversation is accepting.

Besides that, we say that it has $(l_1, \dots, l_\mu)$*-special unsoundness with N potential responses per round* if during the active mode $\mathcal{A}$ can efficiently produce N distinct messages with the previously mentioned property.

As it is said in [13], many $(k_1, \dots, k_\mu)$-special sound protocols are also $(k_1 - 1, \dots, k_\mu - 1)$-special unsound. In our case it is indeed possible to design an adversary $\mathcal{A}$ against the interactive version of the opening protocol that shows it is $(1, 1)$-special unsound.

In the first round $\mathcal{A}$ just needs to guess $\Gamma_1 := \{\widetilde{\alpha}\} \subseteq \mathbb{Z}_q$ and prepare both commitments so that they would be able to pass every verification if the challenge α turns out to be $\widetilde{\alpha}$. $\mathcal{A}$ can do so making up $\{\boldsymbol{e}'_j\}_j \leftarrow_{\text{R}} \mathfrak{B}_{nk}$ (so that the $\pi_{\tau_j}(\boldsymbol{e}'_j) \overset{?}{\in} \mathfrak{B}_{nk}$ verifications checks out), and also making up $\boldsymbol{z} \leftarrow_{\text{R}} \mathcal{L}(\boldsymbol{a})$ and preparing the first commitment choosing $\boldsymbol{y} := \boldsymbol{z} - (\widetilde{\alpha}(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j(\boldsymbol{f}_j + \widetilde{\alpha}\boldsymbol{e}'_j)))$ with a random $s \in R_q$ so that everything checks if α turns out to be $\widetilde{\alpha}$ (as then $\widetilde{\boldsymbol{z}}$ would be $\boldsymbol{z}$, and therefore it would belong to $\mathcal{L}(\boldsymbol{a})$). If the guess is right they can switch to passive mode and satisfactorily answer with the made up s or $\{\boldsymbol{e}'_j\}_j$. Observe they have many options for $\boldsymbol{z}$, s or $\{\boldsymbol{e}'_j\}_j$, so this attack has the many responses per round property.

If the guess was wrong, the adversary could still try to guess the second challenge. If they guess that $\Gamma_2 := \{1\}$ then they only need to answer with the usual $\boldsymbol{g}_j = \pi_{\tau_j}(\boldsymbol{f}_j + \alpha\boldsymbol{e}'_j)$, and then they would be able to satisfactorily answer if the challenge is indeed $b = 1$ with the made up $\boldsymbol{e}'_j$. However, in this case they would only have one possibility. Therefore, to be able to have many responses per round the adversary should guess $\Gamma_2 := \{0\}$ and prepare the $\boldsymbol{g}_j$ so that the $b = 0$ checks are satisfied. They can just randomly sample $\boldsymbol{g}_j$ for $j \geq 1$ and then solve for $\boldsymbol{g}_0$ so that $\boldsymbol{z} = \boldsymbol{y} + \alpha(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j {\pi_{\tau_j}}^{-1}(\boldsymbol{g}_j))$, and they can satisfactorily answer if the challenge is $b = 0$. The freedom of choosing $\boldsymbol{g}_j$ for $j \geq 1$ now guarantees the many responses per round property.

A complete description of the active and passive modes of such an adversary is given in Appendix 4.A. The opening protocol is therefore $(1, 1)$-special unsound with $\Omega(q^{nk})$ responses per round. Then, we can see that there exists an attack exploiting the parallel repetition structure of our non-interactive proof.

Theorem 4.7 (δ-fold parallel repetition attack, Theorem 5 from [12]). *Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be a $(2\mu + 1)$-move public-coin interactive proof with challenge spaces $C_1, \dots, C_\mu$. Suppose $\langle \mathcal{P}, \mathcal{V} \rangle$ has $(l_1, \dots, l_\mu)$-special unsoundness with N responses per round. Let $\langle \mathcal{P}^\delta, \mathcal{V}^\delta \rangle$ be the δ-fold parallel repetition of $\langle \mathcal{P}, \mathcal{V} \rangle$. Let $m_1, \dots, m_\mu \in \mathbb{N}$ such that $\sum_{i=1}^{\mu} m_i = \delta$. Let $Q = \mu Q'$ for $Q' \in \mathbb{N}$ with $Q' \sum_{i=1}^{\mu} \left({}^{l_i}/_{|C_i|}\right)^{m_i} < {}^1/_4$ and $Q' \leq N$. There is a Q-query dishonest prover $\mathcal{P}^*$ against the Fiat-Shamir transform of $\langle \mathcal{P}^\delta, \mathcal{V}^\delta \rangle$ so that for every* *statement x*

$$\Pr\left[\mathcal{V}_{FS}^{\delta\,\mathcal{O}}(x, \mathcal{P}^{*\mathcal{O}}) = \text{accept}\right] \geq \frac{1}{2}\left(\frac{Q}{\mu}\right)^{\mu} \prod_{i=1}^{\mu} \left({}^{l_i}/_{|C_i|}\right)^{m_i}.$$

The theorem from [12] provides a lower bound on the attacker success probability because they want to emphasize that the security loss is $\Omega(Q^{\mu})$. However, in order to choose parameters so that the scheme is secure against this attack what we want is an upper bound. Let us describe the attack strategy and find that upper bound.

The idea is the following. Choosing $m_1, \ldots, m_\mu \in \mathbb{N}$ such that $\sum_{i=1}^{\mu} m_i = \delta$, the attacker tries to guess the challenges, because if $c_i \in \Gamma_i$ for some parallel executions then they would be able to change into passive mode for these repetitions and continue answering till the end. Their goal is to get at least $\sum_{j=1}^{i} m_j$ parallel executions in passive mode before moving to the next round, so that they end with a complete fake proof provided that this strategy ensures all δ parallel executions are in passive mode before the final answer. If not sufficiently many new guesses are correct in a given round to move to the next they just resample new messages to get fresh challenges. To get into account the fact that the adversary is computationally limited and only queries the oracle Q times it is designed to abort after ${}^{Q}/_{\mu}$ unsuccessful attempts per round. See [12] for a full description.

We need to upper bound the success probability of the attack and select the number of repetitions in a way that ensures this upper bound is smaller than $2^{-\lambda}$. Let us first analyze the involved probability distributions and recap some useful probability propositions.

Definition 4.8 (Binomial Distribution)**.** The *binomial distribution* of parameters n and p, denoted $Bin(n, p)$, is the discrete probability distribution that counts the number of successes in n independent experiments, each of them having two possible outcomes, *success* with probability p or *failure* with probability $1 - p$.

Considering the attack, at the ith round there are $\delta - m'_{i-1}$ parallel protocols in active mode (we denote by $m'_{i-1} \geq \sum_{j<i} m_j$, with $m'_0 = 0$, the number of parallel repetitions already in passive mode after the $(i-1)$th round), and each of them turns into passive mode with probability ${}^{l_i}/_{|C_i|}$ (the probability of $c \in \Gamma_i$ if $c \leftarrow_R C_i$). The number of protocol repetitions that turn into passive mode follows a binomial distribution of parameters $\delta - m'_{i-1}$ and ${}^{l_i}/_{|C_i|}$.

Definition 4.9 (Geometric Distribution)**.** The *geometric distribution* of parameter p, denoted $Geo(p)$, is the discrete probability distribution that counts the number of trials until the first success (including the successful one) when sequentially conducting independent experiments, each of them having two possible outcomes, *success* with probability p or *failure* with probability $1-p$.

The attacker advances a round of the protocol if the number of parallel repetitions in passive mode is greater or equal than $\sum_{j\leq i} m_j$, meaning that we require at least $(\sum_{j\leq i} m_j) - m'_{i-1}$ new parallel repetitions turning into passive mode. Let us say that happens with probability p_i (it is the probability of a certain binomial distribution surpassing a certain threshold). If that is not the case then they compute a different message and tries again. If no limit was imposed, the number of trials until it continues to the next round would follow a geometric distribution of parameter p_i.

Particularizing that to our current protocol, let

$$X \sim Bin\big(\delta, q^{-1}\big)$$

be the probability distribution that models the number of parallel repetitions in which the adversary would be able to correctly answer in passive mode after the first challenge. They continue if that number is at least m_1 (recall that $m_1, m_2 \in \mathbb{Z}_{\geq 0}$ are any two non-negative integers such that $m_1 + m_2 = \delta$). The number of attempts until that would happen would follow a distribution

$$Y \sim Geo\big(\Pr\left[X \geq m_1\right]\big),$$

and therefore the adversary that aborts if they cannot continue after Q/μ tries advances to the second round with probability $\Pr\left[Y \leq Q/2\right]$.

Provided that m'_1 repetitions are already in passive mode, which happens with probability $\Pr\left[X = m'_1 \mid X \geq m_1\right]$, the adversary can only fake the proof if it guesses the remaining $\delta - m'_1$ challenges.

For that reason, we analogously define

$$W_m \sim Bin\big(\delta - m, 1/2\big) \text{ and } Z_m \sim Geo\big(\Pr\left[W_m = \delta - m\right]\big),$$

as we might need to consider the case $m'_1 = m$ for any $m_1 \leq m \leq \delta$. These two distributions model the number of correctly guessed challenges and the number of trials until all remaining challenges are guessed. The adversary successfully fakes a proof if $Z_m \leq Q/\mu$.

In conclusion, the adversary success probability is

$$\epsilon = \Pr\left[Y \leq Q/2\right] \left(\sum_{m=m_1}^{\delta} \Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right] \right).$$

Before we start, let us recall some useful probability propositions. We might indicate that we are using them placing a reference in a parenthesis next to where the property has been used.

Proposition 4.10 (Bernoulli's inequality)**.**
For every integer $r \geq 1$ and every real number $x > -1$ we have $(1+x)^r \geq 1 + rx$.

Corollary 4.11. *Let $A \sim Geo(p)$, then $\Pr\left[A \leq k\right] \leq kp$.*

Proof. That is the case because $\Pr\left[A \leq k\right] = 1-(1-p)^k \overset{\text{(Prop. 4.10)}}{\leq} 1-(1-kp) = kp$. □

Proposition 4.12 (Chernoff bound for the binomial distribution)**.**
Let $A \sim Bin(n,p)$, then

$$\Pr\left[A \geq k\right] \leq \exp\left(-n\left(\frac{n-k}{n}\ln\left(\frac{n-k}{n(1-p)}\right) + \frac{k}{n}\ln\left(\frac{k}{np}\right)\right)\right).$$

Proof. Theorem 1 from [9]. □

We can use these tools to bound the attack success probability. Observe that the adversary can choose m_1 and m_2, as long as $m_1 + m_2 = \delta$, to obtain different strategies. However, if m_1 is too large, obtaining enough challenges in Γ_1 might be too difficult, and they would not be able to continue to the second round except with too small probability. We therefore study three separated cases.

Case $\Pr\left[X \geq m_1\right] \leq 2^{-(2\lambda-1)}$.
If m_1 is so large that $\Pr\left[X \geq m_1\right] \leq 2^{-(2\lambda-1)}$ then

$$\epsilon \leq \Pr\left[Y \leq Q/2\right] \overset{\text{(Cor. 4.11)}}{\leq} Q/2 \Pr\left[X \geq m_1\right] \leq Q 2^{-2\lambda} \overset{(Q<2^\lambda)}{<} 2^{-\lambda}.$$

Case $2^{-(2\lambda-1)} < \Pr\left[X \geq m_1\right] \leq 2^{-(\lambda-1)}$.
If m_1 is not that large but still implies that $2^{-(2\lambda-1)} < \Pr\left[X \geq m_1\right] \leq 2^{-(\lambda-1)}$ we can nevertheless use Corollary 4.11 with $\Pr\left[Y \leq Q/2\right]$ to get a non-trivial bound $\Pr\left[Y \leq Q/2\right] \leq Q/2 \Pr\left[X \geq m_1\right]$, but we still have to consider what happens in the second round.

Observe that the probability of a success in the second round, once they have already passed the first round, is the sum of $\Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right]$. We can guarantee the first probability is small if m is large enough, and we can only guarantee the second is small if m is small enough. For that matter, we have to split the summation.

On the one hand,

$$\begin{aligned}
&\Pr\left[Y \le Q/2\right] \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \ge m_1\right] \Pr\left[Z_m \le Q/2\right] \le \\
&\le Q/2 \Pr\left[X \ge m_1\right] \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \ge m_1\right] Q/2 \Pr\left[W_m = \delta - m\right] && \text{(Cor. 4.11)} \\
&= Q^2 2^{-2} \Pr\left[X \ge m_1\right] \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \ge m_1\right] 2^{-(\delta-m)} \\
&\le Q^2 2^{-2(\lambda+1)} \Pr\left[X \ge m_1\right] \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \ge m_1\right] && (2^{-(\delta-m)} \le 2^{-2\lambda}) \\
&= Q^2 2^{-2(\lambda+1)} \Pr\left[X \ge m_1\right] \Pr\left[X \le \delta - 2\lambda \mid X \ge m_1\right] \\
&\le Q^2 2^{-2(\lambda+1)} \Pr\left[X \ge m_1\right] \\
&\le Q^2 2^{-2(\lambda+1)} 2^{-(\lambda-1)} \\
&< 2^{-(\lambda+1)}. && (Q < 2^\lambda)
\end{aligned}$$

On the other hand,

$$\begin{aligned}
&\Pr\left[Y \le Q/2\right] \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \ge m_1\right] \Pr\left[Z_m \le Q/2\right] \le \\
&\le Q/2 \Pr\left[X \ge m_1\right] \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \ge m_1\right] \Pr\left[Z_m \le Q/2\right] && \text{(Cor. 4.11)} \\
&\le Q/2 \Pr\left[X \ge m_1\right] \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \ge m_1\right] \\
&= Q/2 \Pr\left[X \ge m_1\right] \frac{\Pr\left[X \ge \delta - 2\lambda + 1\right]}{\Pr\left[X \ge m_1\right]} \\
&< 2^{\lambda-1} \Pr\left[X \ge \delta - 2\lambda + 1\right] && (Q < 2^\lambda) \\
&\le 2^{-(\lambda+1)}. && \text{(provided } \delta \text{ is large enough)}
\end{aligned}$$

What we need is to choose δ so that $\Pr\left[X \ge \delta - 2\lambda + 1\right] \le 2^{-2\lambda}$. It is always possible if δ is large enough, as it represents the probability of succeeding in all but a constant number of experiments. We would later find a sufficient condition using Proposition 4.12.

Notice we are splitting the summation considering $m_1 \le \delta - 2\lambda$. If that is not the case the first would be an empty sum, and we could even remove some terms from the second sum. In any case, it is clear that, provided δ satisfies the requirement, $\epsilon < 2^{-(\lambda+1)} + 2^{-(\lambda+1)} = 2^{-\lambda}$.

We can finally analyze the case where we cannot take advantage of the adversary aborting with some non-negligible probability without concluding the first round and therefore not even attacking the second round.

Case $2^{-(\lambda-1)} < \Pr\left[X \geq m_1\right]$.

We just have to follow a similar approach, splitting again the summation. We start bounding the first part,

$$\begin{aligned}
&\Pr\left[Y \leq Q/2\right] \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right] \leq \\
&\leq \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right] \\
&\leq \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \geq m_1\right] Q/2 \Pr\left[W_m = \delta - m\right] && \text{(Cor. 4.11)} \\
&= \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \geq m_1\right] Q 2^{-(\delta-m+1)} \\
&\leq Q 2^{-(2\lambda+1)} \sum_{m=m_1}^{\delta-2\lambda} \Pr\left[X = m \mid X \geq m_1\right] && (2^{-(\delta-m+1)} \leq 2^{-(2\lambda+1)}) \\
&= Q 2^{-(2\lambda+1)} \Pr\left[X \leq \delta - 2\lambda \mid X \geq m_1\right] \\
&\leq Q 2^{-(2\lambda+1)} \\
&< 2^{-(\lambda+1)}. && (Q < 2^{\lambda})
\end{aligned}$$

And bound again the second part,

$$\begin{aligned}
&\Pr\left[Y \leq Q/2\right] \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right] \leq \\
&\leq \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \geq m_1\right] \Pr\left[Z_m \leq Q/2\right] \\
&\leq \sum_{m=\delta-2\lambda+1}^{\delta} \Pr\left[X = m \mid X \geq m_1\right] \\
&= \frac{\Pr\left[X \geq \delta - 2\lambda + 1\right]}{\Pr\left[X \geq m_1\right]} \\
&< \frac{\Pr\left[X \geq \delta - 2\lambda + 1\right]}{2^{-(\lambda-1)}} \\
&\leq 2^{-(\lambda+1)}. && \text{(provided } \delta \text{ is large enough)}
\end{aligned}$$

Once again the only condition is that we wave to choose δ large enough so that $\Pr\left[X \geq \delta - 2\lambda + 1\right] \leq 2^{-2\lambda}$ and therefore $\epsilon < 2^{-(\lambda+1)} + 2^{-(\lambda+1)} = 2^{-\lambda}$.

It only remains to obtain a sufficient condition for $\Pr\left[X \geq \delta - 2\lambda + 1\right] \leq 2^{-2\lambda}$, using Proposition 4.12.

$$\begin{aligned}&\Pr\left[X \geq \delta - 2\lambda + 1\right] \leq \\ &\leq \exp\left(-\delta\left(\frac{2\lambda-1}{\delta}\ln\left(\frac{2\lambda-1}{\delta(1-q^{-1})}\right) + \frac{\delta-2\lambda+1}{\delta}\ln\left(\frac{\delta-2\lambda+1}{\delta q^{-1}}\right)\right)\right).\end{aligned}$$

It would be sufficient to ensure that this expression can be upper bounded by $2^{-2\lambda}$. Taking (base 2) logarithms (and replacing natural logarithms with binary logarithms) the final condition would be

$$(2\lambda-1)\log\left(\frac{2\lambda-1}{\delta(1-q^{-1})}\right) + (\delta-2\lambda+1)\log\left(\frac{\delta-2\lambda+1}{\delta q^{-1}}\right) \geq 2\lambda. \qquad \text{(eq. (4.4))}$$

Provided that it is not a difficult computation, we can just try increasing δ, starting from the minimum value determined by eq. (4.1), until we find the first one that satisfies the condition.

Observe that the success probability of the attack could be slightly improved by tweaking some design decisions (for example, the fact that the Q maximum number of queries is equally split among both rounds), but that would not have any meaningful impact. On the other hand, some assumptions are ostensibly conservative. When considering the hardness of the RLWE problems, we wanted to ensure that known attacks need more than 2^{λ} operations to be minimally successful. Here, however, we are bounding by 2^{λ} just the number of oracle calls, not the number of total operations, that would be much greater. By analyzing the expected running time, one could then bound it by 2^{λ} and get stricter upper bounds for Q.

One can see that an equivalent analysis applies to the linear relation protocol. It is also $(1,1)$-special unsound, and an adversary can be constructed by repeating the same strategy three times in parallel, just defining $z_3 := \lambda_1 z_1 + \lambda_2 z_2$ (it is also completely detailed in Appendix 4.A). Therefore, the same kind of attack applies with the same success, and the conditions for the parameters also apply to the linear case.

It is more difficult to assess the security of the multiplicative relation protocol. Recall that the $(2,3,2)$-special soundness is that of a 7-move protocol. However, although in the interactive version the protocol it is equivalent whether the verifier sends α and β sequentially or at the same time, we have already discussed that is not the case in the non-interactive version after applying Fiat-Shamir, as both challenges come from the same hash in the 5-move version and would require two different hashes if transforming the 7-move version.

It is possible to prove that the multiplicative interactive proof is $(1,1)$-out-of-

$(q^2, 2)$-unsound, following a similar strategy than the one used with the linear protocol. Nonetheless, the success probability would be lower, as the first challenge set is now $\mathbb{Z}_q^2$ and guessing pairs of integers is harder than guessing a single integer. However, there is also a partial attack with the same success probability as the one against the opening and linear protocols because a malicious adversary $\mathcal{A}$ knowing valid openings to two commitments $\boldsymbol{c}_1 = \boldsymbol{a}m_1 + \boldsymbol{b}r_1 + \boldsymbol{e}_1$ and $\boldsymbol{c}_2 = \boldsymbol{a}m_2 + \boldsymbol{b}r_2 + \boldsymbol{e}_2$ could fake a multiplicative proof with an arbitrary $\boldsymbol{c}_3 \in R_q^k$ just guessing β. They would choose $\Gamma_1 := \{(\alpha, \widetilde{\beta}) \mid \alpha \in \mathbb{Z}_q\}$ and proceed as an honest prover with regard to $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ and as the adversary against the opening protocol with regard to $\boldsymbol{c}_3$. To make everything satisfy the checks if the challenge β turns out to be $\widetilde{\beta}$ they just need to choose $z_3 = \boldsymbol{a}(\mu_3 + \widetilde{\beta}m_1m_2)$ (with minor additional changes on how the other commitments are computed if the guess turns out to be wrong).

In the end $(q, 1)$-out-of-$(q^2, 2)$-special unsoundness is by all means equivalent to the $(1, 1)$-out-of-$(q, 2)$-special unsoundness (the analysis on the success probability only depends on $l_1/|C_1|$ and $l_2/|C_2|$, and these two are equal for both scenarios), and the same condition on the parameters given by eq. (4.4) would apply for the multiplicative NIZKPoK.

4.3.4 Zero-Knowledge

The interactive version of the protocol is honest-verifier zero-knowledge because there exists a simulator that taking the challenges as input outputs conversations computationally indistinguishable from real conversations between an honest prover and an honest verifier.

Provided that the conversation itself is then indistinguishable from something that can be computed by an algorithm that does not know the witness, we can ensure no relevant information is leaked, because anything that can be computed from the conversation could also be efficiently computed using the simulator.

Finally, in the ROM, as we assume challenges in the Fiat-Shamir transformed version follow a uniform distribution, the non-interactive proof would be in that sense equivalent to the interactive conversation, and we can ensure again that no information is leaked. We use CSPRNGs, so the zero-knowledge property is only computational.

4.3.5 Completeness

As it was the case for the ZKPoKs from the previous chapter all three NIZKPoKs are complete by design.

We have mentioned that Fiat-Shamir with aborts based proofs do not enjoy standard completeness with one execution because there is a significant probability of aborting before sending an accepting final response. In the non-interactive scenario completeness would be obtained starting again from the beginning until a valid proof is produced.

The NIZKPoK version of our protocols uses rejection sampling for sampling uniformly random integers in $\mathbb{Z}_q$. But, unlike the Fiat-Shamir with aborts strategy, the rejection probability does not depend on any secret information, so we can just sample again without restarting the whole procedure or compromising completeness (more details in Section 4.4).

4.4 Implementation

In this section we explain in detail the choices that we have made in our implementation. What we present here is merely a prototype, only intended for academic purposes and not ready for production. There is no user interface to define a message, output or read a commitment, an opening or a proof (everything is kept in memory).

What we do include in the GitHub repository [90] are the core algorithms. The functions that output the public keys, generate a commitment with its opening and verify such opening. Furthermore, we include the functions that generate the opening, linear and multiplicative proofs and the ones that verify them. Together with these algorithms, we also include some tests that produce random commitments and proof statements in order to check that every opening generated by the commitment algorithm is validated by the opening verification, and that every proof is also valid. Besides that, these tests also benchmark all the algorithms, logging the running time in milliseconds and the number of processor cycles.

We have not taken into account memory management or possible side-channel attacks (for example we have not tried to make our implementation constant time), because that would be out of the scope of this research. Commitment public keys are generated from a uniformly random distribution (conditioned to the properties satisfied by binding keys), but depending on the trust model of a particular application $(\boldsymbol{a}, \boldsymbol{b})$ might need to be generated using multi-party computation or deterministically computed using a pseudorandom procedure that convinces the participants that nobody has a trapdoor.

Nevertheless, we do have implemented our constructions taking in mind the sizes of actually secure parameters (not only toy examples). We also have decided to always use CSPRNGs with sufficiently large seeds. We do so because we want to properly measure the performance of the algorithms, and these functions turn out to consume a significant amount of the total running time. A cryptographic scheme is as weak as the weakest of its parts, and transitioning from theoretically uniform distributions to pseudorandom ones is a delicate step worth of being carried out in detail.

To illustrate the importance of this last step we borrow a real example from a different field of cryptography. The wallets in the Ethereum blockchain are protected from brute force attacks because their secret keys are large enough. However, many users and projects have been recently exploited (loosing crypto assets valued hundreds of millions of USD) because in order to get an address with some custom starting characters (a so-called *vanity address*) they used an open-source software called *profanity* that internally generated the private keys deterministically from a random seed obtained from the C++ built in random generator of only 32 bits, allowing very efficient brute force attacks [104]. We believe examples like this strongly support the need of rigorous analysis as we do in this work.

4.4.1 Language and Libraries of the Implementation

All the main algorithms are implemented in C. Only the script used to generate the parameters (as described in the next Section 4.5) is implemented using `sagemath`, because the Lattice Estimator we use is written in this language, and the script that sequentially runs the tests is written in python for its flexibility. We have chosen C as our principal language mainly because of its efficiency. It also allows to improve the efficiency of some computations by working on a reasonably low level. Furthermore, there are plenty of well established and robust libraries, from which we take advantage of the following.

The goal of this implementation is to study the performance of the proposed techniques on a different variety of scenarios. For that reason, even if our restraints do not require a specially large modulus q we have chosen to work with arbitrary large integers anyway, so we can study how the scheme scales with larger q and allows us to compare it with alternatives that do have such constraints. To do so we use the GNU Multiple Precision arithmetic library (GMP v6.2.1).

We also make heavy use of the Fast Library for Number Theory (FLINT v2.9.0) to work with polynomials in a quotient ring such as R_q. Our implementation uses it for any polynomial operation except for products, where we have implemented the faster algorithms described in Chapter 2.

As we have been discussing, random sources and pseudorandom functions are critical for the security of the implementation, and we use the standard OpenSSL library (v3.0.2) for random sampling, hashing (SHA-256) and XOF (SHAKE-128).

One of the main drawbacks of Stern's approach is that the non-negligible soundness error requires multiple repetitions to obtain soundness. Nevertheless, we can take advantage of the parallel repetition structure to parallelize the execution and speed up the protocols. We use OpenMP for that, and the parallelization level has a considerable impact on the final performance.

4.4.2 Main Computational Tasks

The two main computational tasks that we require to implement our schemes are sampling (from a discrete Gaussian distribution, uniformly random integers in $\mathbb{Z}_q$ and uniformly random permutations) and multiplying in a truncated polynomial ring. In the following subsections, we will give an overview of the approach that we have followed to instantiate such tasks in our implementation.

Discrete Gaussian Sampling

We want our schemes to be as flexible as possible, to study the different possible trade-offs. For this reason, we have chosen to instantiate discrete Gaussian sampling using the *Discrete Ziggurat* (DZ) algorithm [32]. We have implemented the procedure from scratch, and it can work with multi-precision arithmetic. This algorithm works covering the probability density function with rectangles of similar size. Then, to sample an element, one of the rectangles is selected uniformly at random, and afterwards a rejection sampling procedure is used to either obtain one of the values inside the rectangle or start again, so that the final probability distribution is that of a truncated discrete Gaussian. The number of rectangles used allows us to have a trade-off between computational speed and memory usage, as a more fine-graded rectangle partition implies faster sampling but needs more memory for allocating the rectangle coordinates. When we have no restriction on memory consumption, DZ achieves a performance close to the fastest known algorithm for this task [69]. Using DZ has another advantage, it allows us to directly sample from a bounded discrete Gaussian distribution. The details of DZ are out of scope of this work, and we refer the interested reader to the original paper.

Uniform Sampling

Even if computationally simpler than Gaussian sampling, the implemented algorithms make heavy use of uniform sampling of integers in $\mathbb{Z}_q$, and this task ends up being more computationally expensive.

Our implementation makes use of the RAND_priv_bytes function from OpenSSL as a source of randomness. To sample from $\mathbb{Z}_q$ we have to do rejection sampling. We can sample $\lceil \log_{2^8}(q) \rceil$ bytes to get an integer uniformly distributed in $[0, 2^{8\lceil \log_{2^8}(q) \rceil} - 1]$, and then reduce it modulo $2^{\lceil \log(q) \rceil}$ to get an integer s uniformly distributed in $[0, 2^{\lceil \log(q) \rceil} - 1]$. We keep s if $s < q$, and otherwise we discard it and start again.

The intermediate reduction modulo a power of 2 guarantees a reasonable rejection rate smaller than $1/2$, which is really worthy as reducing modulo a power of two can be done at a low level and is computationally cheap, so this approach turns out to be a faster option than similar alternatives which require reductions modulo q.

If we have to sample multiple integers, for example to sample vectors of polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1 \rangle$, in order to amortize the cost of calling the RAND_priv_bytes function we first populate a buffer of pseudorandom bytes and use them sequentially, only refreshing the buffer with new randomness when there are less than $\lceil \log_{2^8}(q) \rceil$ bytes left.

To deterministically sample integers from $\mathbb{Z}_q$ using a seed we use the same approach, replacing RAND_priv_bytes with an XOF as SHAKE-128, initialized with both the seed and a counter that keeps track of how many times we have already refreshed the buffer. The previously defined PRN, the function that takes as input a given seed and outputs the desired number of pseudorandom integers modulo q, is implemented following this procedure.

We also need to uniformly sample multiple permutations from the set of permutations of $2nk$ elements. A permutation is encoded as a product of $2nk$ transpositions $(1, i_1)(2, i_2) \dots (2nk - 1, i_{2nk-1})(2nk, i_{2nk})$ where (j, i_j) is such that $i_j \geq j$. That way, sampling $2nk$ indexes uniformly in the respective intervals, $i_j \leftarrow_{\text{R}} [j, 2nk]$, determines a permutation uniformly distributed in the set of permutations of $2nk$ elements $\mathfrak{S}_{2nk}$ (i.e. following the Fisher-Yates algorithm).

We can deterministically compute such pseudorandom indexes from a seed similarly than before. Sampling from an interval $[j, 2nk]$ is equivalent to sampling from $[0, 2nk - j]$ and then adding j. However, for this case, where indexes to be sampled are much smaller, we alternate the order of the rejection and the modular reduction to reduce the rejection rate. To sample an integer uniformly from $[0, 2nk - j]$ we can also sample an integer uniformly from $[0, m(2nk - j + 1) - 1]$ for any $m \in \mathbb{Z}_{>0}$ and then reduce it modulo $2nk - j + 1$. The greater multiple of $(2nk - j + 1)$ smaller or equal than $2^{8\lceil \log_{2^8}(2nk-j+1) \rceil}$ is $2^{8\lceil \log_{2^8}(2nk-j+1) \rceil} - (2^{8\lceil \log_{2^8}(2nk-j+1) \rceil} \text{ rem } (2nk - j + 1))$. We start sampling $s \leftarrow_{\text{R}} [0, 2^{8\lceil \log_{2^8}(2nk-j+1) \rceil} - 1]$. Then if $s < 2^{8\lceil \log_{2^8}(2nk-j+1) \rceil} - (2^{8\lceil \log_{2^8}(2nk-j+1) \rceil} \text{ rem } (2nk - j + 1))$ we reduce s modulo $2nk - j + 1$ and add j, or reject it and repeat otherwise.

We denote by π_τ the permutation obtained following this procedure from a seed $\tau \in \{0, 1\}^{8\lceil \lambda/8 \rceil}$.

This approach greatly reduces the rejection rate and improves the efficiency of the algorithm. It was not the case for the $\mathbb{Z}_q$ sampler because in that case the modular reduction over q would cost more than the modular reduction over a power of two, and the benefit obtained from the reduced rejection rate would not compensate that. Now we are in a different scenario, provided that $2nk$ is much smaller than q and modular reductions are much more efficient when we do not have to deal with arbitrary large integers.

Multiplication in a Truncated Polynomial Ring

The dominant computational task we do with the elements of the scheme is computing the product of polynomials, as it is much more costly than additions and multiplications by a scalar. For that reason we have devoted the entirety of Chapter 2 to explain when and how to efficiently compute such products using partial FFT.

Most of the other computations we have to do, as sampling uniformly random polynomials, adding them and verifying equalities, can also be done in the transformed domain, so in our implementation almost every polynomial is directly represented in the transformed domain and never anti-transformed back. From a mathematical point of view it is completely equivalent, as the polynomial in the transformed domain still represents the same mathematical object. From a computational point of view directly working in the transformed domain, where products are cheaper, saves a lot of computations.

The (α, ω) roots of -1 and 1 that define an (α, ω)-set as described in Definition 2.19 are computed by the testing program when q, n and d are known, and the main algorithms consider them as part of the public parameters pp.

There is a full research area devoted to efficient low level implementations of this kind of lattice operations involving partial FFT [18]. We provide a much simpler implementation without considering such level of detail, and compute the last step using the implementation of Karatsuba's algorithm provided by FLINT.

4.5 Instantiation

In this section we explain how to obtain secure sets of parameters to instantiate our commitment scheme. Defining the security theoretically, i.e. with asymptotic proofs, requires a different analysis than the one necessary for claiming a specific level of security for a particular set of parameters. Besides that, finding the best set of parameters that fulfills these particular conditions is not direct either.

Analyzing the security for concrete parameters, we have established a set of restrictions, eqs. (3.1) to (3.5) and (4.1) to (4.4), that ensure that the commitment scheme and the companion NIZKPoKs would enjoy a certain security under established hypothesis.

However, finding a set of parameters that fulfills all these conditions might not be direct. It is even harder to find the *best* set of parameters. On the one hand it depends on what is the final purpose, as we might prefer a smaller commitment size, smaller NIZKPoKs sizes, faster implementations, lower overhead between the commitment and the message sizes. . . and we might have additional restrictions, again on the size of certain parameters or in any other characteristic.

One has to consider that decreasing one parameter that reduces the commitment size might reduce the available space for other parameters, as reducing one parameter might end up making another condition more restrictive, forcing us to increase other parameters. In the end that could (possibly) yield to a worse global outcome regarding commitment (or proofs) size (or running time).

To obtain our results, we have chosen to carefully study the selection of parameters and define a specific procedure that ends up in an optimal outcome for this particular scheme. Our strategy consists on finding an order such that when the previous parameters are fixed we can find an optimum value for the next one, where optimum means that, conditioned on the previously fixed parameters, it is the value that minimizes size and number of operations of both the commitments and proofs while still satisfying the relations and, most importantly, it makes the conditions for the remaining parameters as loose as possible (i.e. it does not further restrict the available search space more than it is strictly necessary).

For this particular commitment scheme λ, n and q are design parameters that we can freely select depending on the level of security (λ), message space (n,q), or additional properties that we desire. The power of 2 denoted by d that determines the number of irreducible factors $x^n + 1$ splits into when considered modulo q is directly determined by q as the only d that satisfies eq. (3.1). In fact, one should first choose d in order to find a suitable q, so we include it in the set of four parameters characterizing a commitment instantiation set of parameters (λ, n, q, d). Given this tuple of four parameters we can see that the rest can then be defined in an optimal way.

In order to ease the explanation, we are going to present the procedure from the last to the first parameter.

If we had already fixed $(\lambda, n, q, d, \delta_{OL}, \delta_M, B, k)$ satisfying eqs. (3.1) to (3.3) and (4.1) to (4.4) then σ would be upper bounded by the condition from eq. (3.5).

$$\sigma \leq (B-1)\sqrt{\frac{\log(e)}{2(\lambda + \log(nk) + 1)}}. \qquad \text{(from eq. (3.5))}$$

Observe the right-hand side corresponds to $\mathsf{sigmaFromB}(\lambda + \log(nk), B-1)$ as defined in Section 1.4. We proved in Proposition 1.66 that it corresponds to the largest value of σ such that we can ensure $|e| \leq B-1$ except with a probability smaller than $2^{-(\lambda+\log(nk))}$ if $e \leftarrow_{\mathrm{R}} D_\sigma$. Furthermore, $\lambda + \log(nk) = \mathsf{vecBoundedPrToBoundedPr}(\lambda, nk)$, and we precisely know from Proposition 1.65 that $2^{-(\lambda+\log(nk))}$ is the greatest probability we can allow for an event with a single sample if we want to ensure it only happens with probability at most $2^{-\lambda}$ when we consider nk samples. That is, it is the maximum bound on the probability of $|e| > B-1$ with $e \leftarrow_{\mathrm{R}} D_\sigma$ that implies $\Pr\left[\boldsymbol{e} \notin [-(B-1), \ldots, B-1]^{nk} \mid \boldsymbol{e} \leftarrow_{\mathrm{R}} D_\sigma^{nk}\right] \leq 2^{-\lambda}$. And that is what we wanted because we need $\|\boldsymbol{e}\|_\infty \leq B$ except with $2^{-\lambda}$, and this bound is the maximum value of σ for which we can guarantee it. The previous inequality is therefore equivalent to

$$\sigma \leq \mathsf{sigmaFromB}(\mathsf{vecBoundedPrToBoundedPr}(\lambda, nk), B-1).$$

Considering all the other elements are already fixed, the size of the commitments and the proofs would already be determined. As long as we keep satisfying eq. (3.5) the only impact different σ values would have would be on the hardness of the underlying RLWE problem. We use Albrecht *et al.*'s Lattice Estimator [8] as a black box to compute an estimation, albeit it is only reasonable to assume that greater errors would yield to a harder problem, that is, bitsec should be increasing in σ. Consequently, our best candidate would be precisely the value that reaches this upper bound.

The only other remaining condition is precisely eq. (3.4), imposing a minimum level of estimated hardness. Then there are two options, either the σ that comes from the bound derived from eq. (3.5) satisfies eq. (3.4), and we choose it, or it does not, and we can conclude that no σ exists so that we can add it to the previous $(\lambda, n, q, d, \delta_{OL}, \delta_M, B, k)$ to get set of parameters we can ensure is secure. The final result is then, if possible, the greater σ satisfying eq. (3.5).

$$\mathsf{bestSigma}(\lambda, n, q, B, k) := \begin{cases} (B-1)\sqrt{\frac{\log(e)}{2(\lambda+\log(nk)+1)}} & \text{if } \mathsf{bitsec}(\mathrm{RLWE}_{n\,q\,k\,D_\sigma^n}) \geq \lambda \\ & \text{with } \sigma := (B-1)\sqrt{\frac{\log(e)}{2(\lambda+\log(nk)+1)}}, \\ \perp & \text{otherwise.} \end{cases}$$

Something similar happens with k if we assume $(\lambda, n, q, d, \delta_{OL}, \delta_M, B)$ fixed satisfying eqs. (3.1), (3.3) and (4.1) to (4.4). The condition defined by eq. (3.2) establishes a lower bound, and it is the best candidate because a smaller k implies smaller commitments and proofs and a larger k would reduce the available search space for σ satisfying the remaining conditions defined by eqs. (3.4) and (3.5) (now it is clear the hardness is decreasing in k, and the upper bound on σ is also decreasing in k). Then we can safely choose this lower bound as our candidate for k.

$$\mathsf{bestK}(\lambda, n, q, d, B) := \left\lceil \frac{\lambda + 2n\log(q)}{n\left(\log(q)/d - \log(4B-1)\right)} \right\rceil .$$

Regarding B, fixing $(\lambda, n, q, d, \delta_{OL}, \delta_M)$ satisfying eqs. (3.1) and (4.1) to (4.4), the size (and running time) of the proofs would be smaller the smaller its size. Regarding k a larger B would make the lower bound for k derived from eq. (3.2) greater, effectively reducing the available search space, so smaller B would be preferable for this subject. However, the condition imposed by eq. (3.5) is stricter with smaller B, therefore if B is too small it might be the case that, even with the less restrictive k defined by bestK there is no large enough σ satisfying the remaining conditions eqs. (3.4) and (3.5) that makes the underlying RLWE problem sufficiently hard. Parameter k has an impact on the size and running time of both the commitment and the proof algorithms, while more options for σ would only provide more security. Notwithstanding a harder underlying problem is always desirable we define our procedure prioritizing performance (in this case parameter k), because we want to obtain the best instantiation for a target security level and if what we preferred was a harder problem we could have just chosen a greater λ from the beginning. For that matter we simply ensure B is the minimum power of 2 so that the scheme can be securely instantiated. In this case we cannot explicitate a formula for B, and we can just implicitly define it.

$$\mathsf{bestB}(\lambda, n, q, d) := \begin{cases} \min\left\{2^e \;\middle|\; e \in \mathbb{Z}_{>0},\ \exists k, \sigma \text{ s.t. eqs. (3.2) to (3.5) hold}\right\} & \text{if defined,} \\ \perp & \text{otherwise.} \end{cases}$$

In order to compute it, we just need to test increasing powers of 2 verifying if with $k_e := \mathsf{bestK}(\lambda, n, q, d, 2^e)$ we have $\sigma_e := \mathsf{bestSigma}(\lambda, n, q, 2^e, k_e)$ different from $\perp$. Provided that eq. (3.3) imposes an upper bound to B, we can use it to stop the search because if we reach the bound then we know that no suitable parameter set exists with such $(\lambda, n, q, d, \delta_{OL}, \delta_M)$.

Once we have fixed a specific tuple of (λ, n, q, d) we notice that, provided δ_{OL} and δ_M only appear in eqs. (4.1) to (4.4), respectively, and these equations only involve λ and q, choosing them would not restrict the search space for the rest of parameters. Regarding δ_{OL} and δ_M specifically, as they denote the number of repetitions to achieve soundness, the minimum value that satisfies eqs. (4.1), (4.2) and (4.4) will produce the shorter proofs. Then we can directly start with the minimum values satisfying eqs. (4.1) and (4.2),

$$\begin{aligned}\text{deltaOLinitialCandidate}(\lambda, q) &:= \left\lceil \frac{\lambda}{\log(2q) - \log(q+1)} \right\rceil,\\ \text{deltaMinitialCandidate}(\lambda, q) &:= \left\lceil \frac{\lambda}{\log(2q^2) - \log(q^2+3q-2)} \right\rceil,\end{aligned}$$

and keep increasing them until δ_{OL} and δ_M satisfy eq. (4.4).

The size of the seeds used to compute the challenges can be directly computed as $8\lceil(\lambda + \delta \log(q))/8\rceil$ to satisfy eq. (4.3).

Notice these two parameters are not entangled with the others and could be chosen at any other moment, but for the rest we have a clear order defined by $(\lambda, n, q, d) \to B \to k \to \sigma$.

For the first initial elements (λ, n, q, d) we just need to check that n is a power of 2, that $d \leq n$ is another power of 2 and that q is a prime satisfying eq. (3.1).

We use the Miller-Rabin primality test [100] as implemented by Zhiming Wang in [123]. Of course a non-probabilistic primality test could also be used.

See Appendix 4.B for the complete *script* in `sagemath` implementing this procedure and the GitHub repository [90] for the command line executable version in `generate_params.sage`.

4.6 Results

In this final section we present the sizes of the commitment, the NIZKPoKs of a valid opening, a linear and a multiplicative relation, and the running time for the keygen, commit, verifier, proveropening, verifieropening, proverlinear, verifierlinear, provermultiplicative and verifiermultiplicative implemented algorithms for some sets of parameters of interest.

The performance values presented in this section and in Appendix 4.C correspond to the average of 100 executions of each algorithm, compiled with gcc (version 11.3.0 and -O3 -march=native flags) using an Intel® Core™ i7-8700 CPU (6 cores and 12 threads) running at 3.20 GHz with Ubuntu 22.04.1 LTS and 16 GB of RAM.

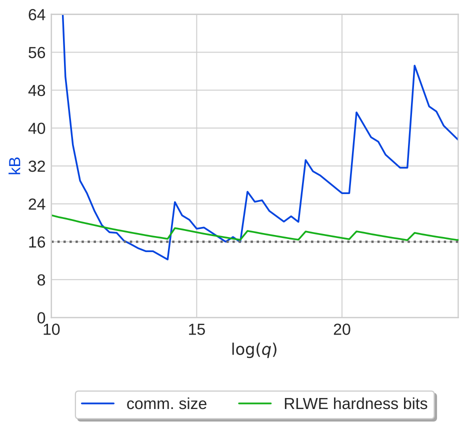
We have chosen to represent sets of parameters with $\lambda = 100$ security bits, lattice dimension $n \in \{512, 1024\}$, and $d = 2$. For each modulus bit size b we have chosen four q modules such that $2^b < q < 2^{b+1}$ (equally spaced in the log scale). There is a huge range of sizes for which secure sets of parameters exist, but not all of them are equally efficient. We are interested in showing the whole picture and not only a few examples, but, as expected, after some size of the modulus q both sizes and running times for the commitment and the NIZKPoKs just increase beyond what can be practically benchmarked. For that reason we truncate the plots when the multiplicative NIZKPoK would take more than 512 MB (it is an arbitrary limit, but it allows us to see how the scheme scales when we surpass the practical sizes while still being able to plot everything in a reasonable scale). Finally, for each of these (λ, n, q, d) the optimal set of parameters has been computed following Section 4.5.

This is not an exhaustive list of parameters, not only because we omit parameter sets with greater q but because even sets with the same $\lceil \log(q) \rceil$ might still behave differently. We believe however that we show enough results to get a comprehensive understanding of the benchmarks. Furthermore, in Appendix 4.C we extend these results to $d = 4$, which is the only other possibility for which secure sets of parameters exist (but we will see that $d = 2$ is, by all means, the best choice). Even if we do not test the running times of parameter sets when the size of the multiplicative proof would exceed 512 MB we also include in Appendix 4.C the size analysis of additional sets of parameters, and explain why there is no possible trade-off that would provide better results.

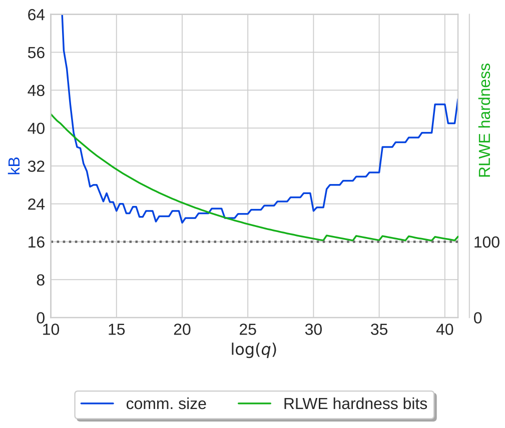
4.6.1 Sizes and Running Times

We start plotting in Figure 4.1 the size of the commitment in kB with the best set of parameters against the size of the modulus, for both $n = 512$ and $n = 1024$, together with the hardness of the underlying RLWE problem.

Figure 4.1 Commitment sizes



(a) $\lambda = 100, \; n = 512, \; d = 2$



(b) $\lambda = 100, \; n = 1024, \; d = 2$

It is interesting to see a zigzag pattern in the $n = 512$ figure that we can easily understand looking at the RLWE hardness data. The discrete nature of the parameters, for example the power of two bound B of the noise terms, means that even if we target 2^{100} as RLWE hardness the problem might not be hard enough for

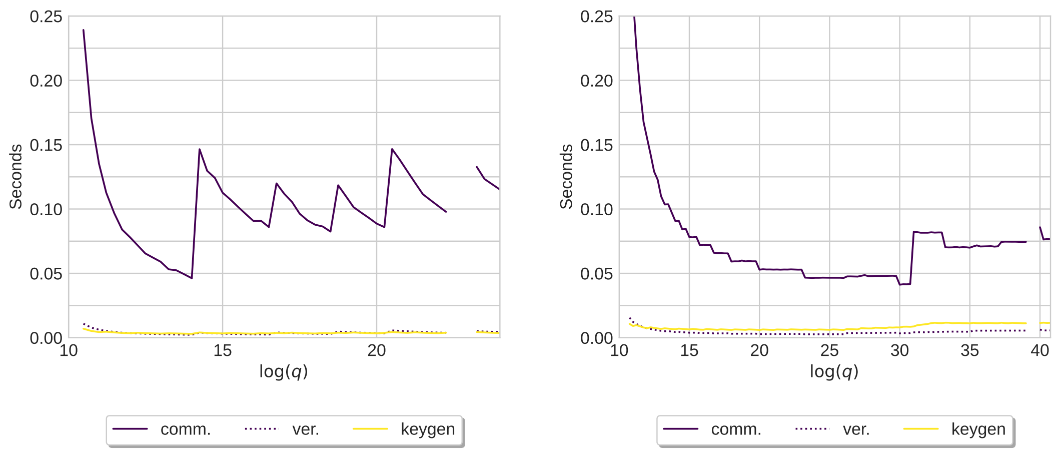
$B/2$ but B already induces a σ that achieves a much greater bitsec. This margin allows increasing q without having to increment B, because, even if the hardness decreases when the ratio of σ/q does, it might still be greater than 2^{100}. The intermingled restrictions from the security conditions imply that, while B can remain constant, albeit choosing a greater q implies using greater integers, the minimum dimension of the vectors k decreases and the commitment sizes can be smaller. This happens up to the point when the security would decrease below the 2^{100} level, which forces us to use the next power of 2 for B and that modifies all the relations ending up with the jumps visible in Figure 4.1.

While the smaller size is achieved with $n = 512$ the commitments with $n = 1024$ start with a much harder RLWE problem, and that grants the existence of a much wider range of still reasonable sizes, given that the jumps start for much greater q and the zigzag pattern is much less prominent.

This non-smooth behavior, a consequence of the numerous non-linear restrictions on the parameters, forces the final user to do a thorough study of the available parameters for this kind of schemes, like the one we are presenting in this section, because even starting with very similar q we could end up defining two instances with very different performance.

We then plot the running times of the key generation, the commitment, and the verification of an opening in Figure 4.2.

Figure 4.2 Commitment times



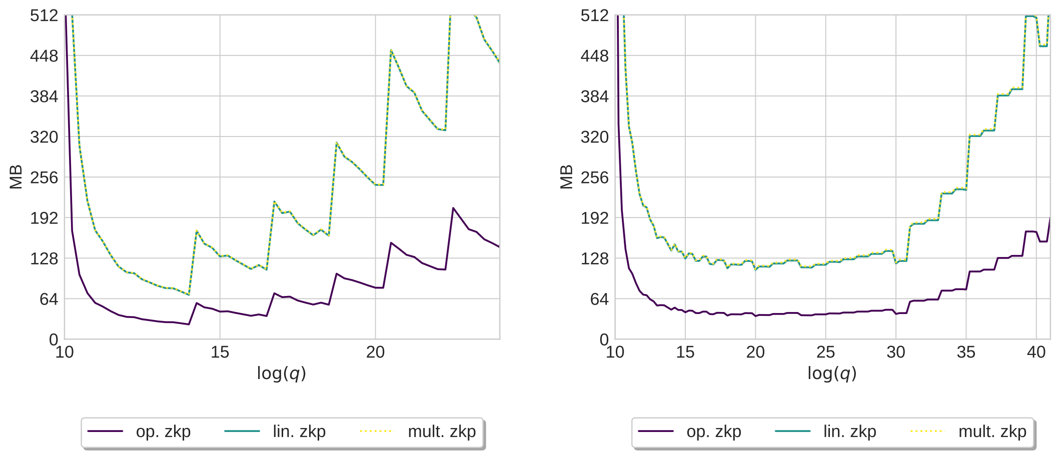
(a) $\lambda = 100, \; n = 512, \; d = 2$ **(b)** $\lambda = 100, \; n = 1024, \; d = 2$

The times are highly correlated with the sizes, and the jumps are even more significant. We see that key generation and verification are extremely fast, and committing can also be done very efficiently.

The NIZKPoKs sizes are plotted in Figure 4.3 and their running times are plotted in Figure 4.4. Proofs of linear and multiplicative relations take almost the same space and have the same running time. In both cases, it is roughly three times the amount required for the proof of knowledge of a valid opening.

Using the proposed NIZKPoK, we can prove knowledge of an opening under a second, and enjoy an even faster verification of the proofs, slightly above half the time required for generating the proof. We see however that, in this case, the performance escalates worse with larger q than the commitment time. That is the case because now an increase in B not only affects the rest of parameters but also the size and running time themselves, as the bit decomposition of the errors implies that both grow with $\log(B)$.

Figure 4.3 NIZKPoKs sizes



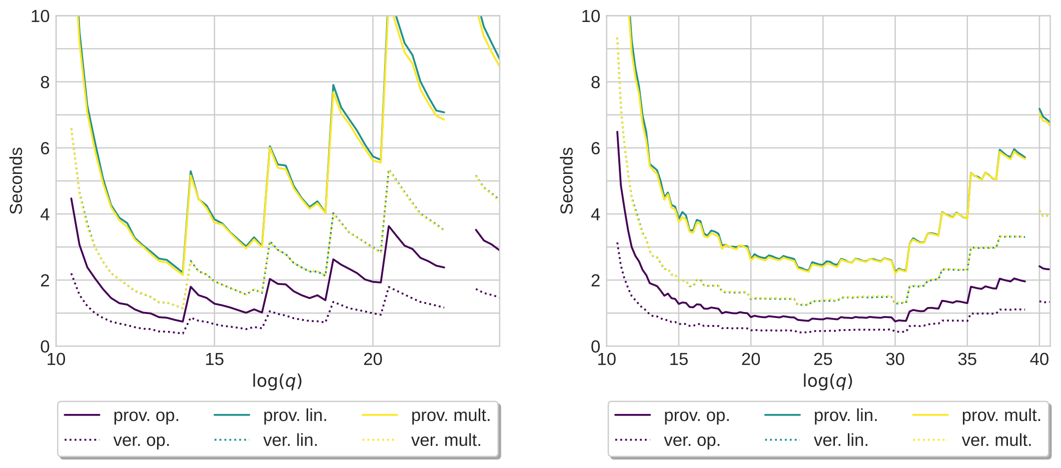
(a) $\lambda = 100, \; n = 512, \; d = 2$ **(b)** $\lambda = 100, \; n = 1024, \; d = 2$

4.6.2 Most Efficient Secure Sets of Parameters

In what follows we present several secure sets of parameters that we choose to highlight because they are optimum regarding size, time or commitment to message ratio. We introduce them in Table 4.1, show their commitment and proof sizes in Table 4.2 and their associated running times for each protocol in Table 4.3. For each of these criteria, we have picked the best with $n = 512$ and the best with $n = 1024$, highlighting in boldface the best value for each n. Some sets are optimal regarding many different aspects, while others only stand out in one table and not in the others.

We want to remark that Table 4.1 contains only one set of parameters with $n = 512$ because this set is the best option regarding the ratio k between the commitment and the message size as well as size and running time of every protocol. It is not the case with $n = 1024$, where different sets are better for different characteristics. We

Figure 4.4 NIZKPoKs times



(a) $\lambda = 100, \ n = 512, \ d = 2$ **(b)** $\lambda = 100, \ n = 1024, \ d = 2$

Table 4.1 Best parameters with $n = 512$ and $n = 1024$

n	q	d	λ	k	σ	B	δ_{OL}	δ_M
512	16381	2	100	**14**	0.55	8	221	221
1024	1048573	2	100	8	0.55	8	213	213
1024	11863253	2	100	7	0.55	8	211	211
1024	16777213	2	100	7	0.55	8	210	210
1024	67108837	2	100	7	0.55	8	209	209
1024	1073741789	2	100	**6**	0.55	8	208	208
1024	1276901389	2	100	**6**	0.55	8	208	208
1024	1518500213	2	100	**6**	0.55	8	208	208
1024	1805811253	2	100	**6**	0.55	8	208	208

present in boldface the k that are optimal for each n.

It is noticeable that all the modulus q are smaller than 2^{32}, so arbitrary-precision arithmetic is not essential for these sets of parameters. While decomposing the errors into bits seems a great overhead, it is not that expensive provided that the bound B can be as small as just 8.

Then in Table 4.2 we show the commitment and proof sizes of the previous sets of parameters, again highlighting in boldface the best cases for each n.

We see that the commitment sizes that we can obtain are completely practical. There are 12 kB commitments when $n = 512$ and around 20 kB commitments when $n = 1024$. Doing our own study of possible secure parameters for the commitment scheme by Benhamouda *et al.* [19] (with which we share the main commitment

Table 4.2 Sizes of the best parameters

n	q	com. size	op. size	lin. size	mult. size
512	16381	**12 kB**	**23.34 MB**	**69.99 MB**	**70.48 MB**
1024	1048573	**20 kB**	**36.47 MB**	**109.38 MB**	**110.70 MB**
1024	11863253	21 kB	37.83 MB	113.45 MB	115.02 MB
1024	16777213	21 kB	37.65 MB	112.91 MB	114.48 MB
1024	67108837	22 kB	40.53 MB	121.56 MB	123.24 MB
1024	1073741789	22 kB	39.85 MB	119.53 MB	121.46 MB
1024	1276901389	23 kB	41.16 MB	123.46 MB	125.45 MB
1024	1518500213	23 kB	41.16 MB	123.46 MB	125.45 MB
1024	1805811253	23 kB	41.16 MB	123.46 MB	125.45 MB

structure) we obtained commitment sizes around 800 kB (to get this value we studied their commitment as is, but some particular decisions they made seem specifically designed to get theoretical asymptotic proofs and not efficient implementations). That is the case because our commitment can be instantiated with $n = 512$ or $n = 1024$ and reasonable modulus size q, while their scheme needs at least $n = 4096$ and $q > 2^{85}$ to get the same security level 2^{100}. Even the improved variant of Benhamouda *et al.* commitment scheme presented in [17], changing both the underlying problem and the structure, still has a greater commitment size of 54.5 kB. Our $n = 512$ version has the same order of magnitude than the commitment scheme by Baum *et al.* presented in [17], that requires 8.1-9 kB, depending on the version.

On the other hand, proof sizes are quite large. The opening proof size with $n = 512$ is of the order of 23 MB, and almost a 60% more when $n = 1024$. The respective opening proof size is tripled for relation proofs. This is the main drawback of Stern-based proofs, as the commitments from [19, 17] have proofs with a size comparable to that of the commitments.

We finally present all protocol times, in milliseconds, in Table 4.3. Again, we highlight with boldface the optimum results.

Execution time is definitely the main asset of this scheme, as we already obtain very fast committing and verifying time and almost practical proving and proof verifying times regarding that we have only developed a prototype implementation that could benefit from higher parallelization and further optimizations. It is interesting to notice that regarding running time, there is not much difference between $n = 512$ and $n = 1024$.

The running time performance and size efficiency might vary a lot between different approaches. For example, the post-quantum SNARK introduced in [30]

Table 4.3 Running time of the best parameters (in milliseconds)

n	q	com.	ver.	key.	$\mathcal{P}_{op}$	$\mathcal{V}_{op}$	$\mathcal{P}_{lin}$	$\mathcal{V}_{lin}$	$\mathcal{P}_{mult}$	$\mathcal{V}_{mult}$
512	16381	**46**	**2**	**3**	**742**	**380**	**2228**	**1146**	**2154**	**1149**
1024	1048573	53	**3**	**6**	877	478	2627	1426	2595	1453
1024	11863253	46	**3**	**6**	783	416	2365	1251	2315	1266
1024	16777213	47	**3**	**6**	765	**413**	2286	**1243**	2250	**1260**
1024	67108837	46	**3**	**6**	810	462	2462	1371	2394	1383
1024	1073741789	**41**	**3**	8	**748**	437	**2244**	1275	**2202**	1298
1024	1276901389	42	4	9	781	434	2348	1301	2318	1326
1024	1518500213	**41**	4	9	770	435	2302	1307	2286	1326
1024	1805811253	42	4	8	764	434	2290	1306	2255	1326

that uses the Aurora framework, based on entirely different assumptions that we are going to omit here, produces proofs of knowledge of a RLWE sample of only 70 kB. but takes around 40 s.

It is important to notice that while we can obtain the smallest commitment size for secure instantiations with $n = 512$ the ratio between the size of the commitment and the size of the committed message can be much smaller, $k = 6$, if we choose $n = 1024$, and even if the sizes increase the running times are still reasonable.

This alone proves the value of the approach from [87]. Even if the commitment itself has the same structure as the one from [19] the latter imposes

$$k > \frac{18\lfloor \log(q)/\log(n) \rfloor}{3\lfloor \log(q)/\log(n) \rfloor - 16},$$

with the additional restriction of $\lfloor \log(q)/\log(n) \rfloor > 6$. While asymptotically their bound is $k > 6$ for the minimal possible $q \sim 2^{70}$ when $n = 1024$ one would start with $k \geq 26$, and even increasing $q \sim 2^{100}$ one would still have $k \geq 13$.

In the previous graphs and tables the verifier times measure the time required to verify the total δ_{OL} or δ_M parallel verifications from which, on average, half of them correspond to $b_i = 0$ challenges and half of them to $b_i = 1$ challenges. It is relevant to point out that the second case is much faster than the first. We provide a general implementation where these challenges b_i are chosen pseudorandomly approximating a uniform distribution over $\{0, 1\}$. However, a specific implementation for a particular set of parameters where the specific running-time ratio between the two options can be computed could be optimized drawing the challenges from a non-uniform distribution, choosing more frequently the faster option $b = 1$. That would require an increase in the number of rounds to preserve the security level, but it could produce a significant net benefit (the details of this strategy, called *thrifty zero-knowledge*, are well described in [41]).

4.7 Conclusions

The main conclusion we would like to remark is that in order to assess the actual efficiency of a lattice-based commitment scheme one needs to conduct a thorough study and address every detail, because these small issues have a great impact on the final performance. Only now that we have a clear understanding of the benefits and the drawbacks of the studied strategies and know where are the main bottlenecks we can propose future research paths to further improve these schemes.

The first version of the commitment scheme and the companion ZKPoKs published in [87] already proved that the $\log(q)$ overhead of the existing proofs could be avoided (because we do not need to decompose in bits the committed messages m), and reduced the soundness error from $2/3$ to $1/2$.

However, we need to carefully analyze the constants to see if the second improvement is worthy. We already improve the soundness proofs from [87] in Chapter 3 so that we can explicitly analyze these constants. If we target $\lambda = 100$ as a security level a 3-move protocol with soundness error of $2/3$ would require $\delta > 200/(\log(3)-1) \approx 342$ repetitions, so that once we compensate the $Q < 2^{100}$ security loss of the Fiat-Shamir transform we still have a bound on the cheating probability of $(2/3)^{\delta} \cdot Q < 2^{-100}$.

In the interactive version the cost of going from a 3-move protocol to a 5-move protocol was the increased interaction implying a higher communication time. This problem vanishes with the non-interactive version, but we have to take into account the greater security loss with the Fiat-Shamir transform. If we need provable security then the $\delta \approx 300$ we have seen in Section 4.3.2 is not so far from the one we would obtain with a $2/3$ soundness error (and we should carefully study the exact size of the conversation to see if it compensates). Choosing to consider only the best attacks as we have done in Section 4.3.3 we have later seen in Section 4.6 that the number of repetitions we obtain is as low as $\delta = 208$. On the one hand we see that, for the specific case of the studied schemes, the price we pay for increasing the number of moves from 3 to 5 is just an additional 8 repetitions. We can say that reducing the soundness error at the expanse of increasing the rounds is really a significant improvement. On the other hand, we have to remark that the security loss of the Fiat-Shamir transform, even if only considered heuristically, has a great impact. As we have mentioned, assuming the number of oracle queries $Q < 2^{100}$ is a conservative estimate, but getting a lower upper bound would require a deeper analysis on the complexity of the attacks.

We also have to mention that good estimations on the hardness of the underlying RLWE problem are something that have a great impact too. There is still an ongoing effort improving attacks to the RLWE problem that we need to take into account to renovate the constants. The Lattice Estimator from Albrecht *et al.* is actively maintained with updates that refine the estimations, and while instantiating our protocols we have had to update our sets of parameters several times because the newer versions deemed the previous sets insecure. Each increase in the constant B has implied a greater size of the proofs.

We however believe this is the only honest way of ensuring a certain level of security. There are many proposed schemes that choose to draw only binary errors (or ternary $\{-1, 0, 1\}$ errors) from non-Gaussian distributions, but then do not enjoy the same reductions to problems already believed to be hard (as discussed in Section 1.3). We think having both the theoretical reductions and up-to-date practical estimations are both important. Regarding the second, we consider it is of particular interest the fact that we have uploaded to the GitHub repository [90] the complete script we have used to obtain the secure parameters with the Lattice Estimator as a module, so anyone could easily run it again in the future with an updated version of the estimator to get up-to-date secure sets.

Combining these two factors, a rigorous analysis of the security of the Fiat-Shamir transform and a rigorous analysis of the hardness of the underlying problem we get that the minimum sizes we obtain for the NIZKPoKs are still not practical, in spite of the significant improvements we have achieved with respect to existing Stern-based proofs.

Once we have seen that the overhead needed to obtain secure NIZKPoKs makes the sizes too big for practical use we could consider again the applicability to interactive cases. While non-interactive proofs seem preferable because a single proof is universally verifiable there are some scenarios where we are just interested into convincing a single verifier, and we might even want that the verifier is not able to convince a third person, as might be the case with e-voting. For non-critical infrastructure using identifications schemes, where multiple failed attempts can be easily detected, it might even be feasible to settle for looser soundness errors as 2^{-80}, taking into account that this bound is statistical, still targeting $\lambda = 100$ for the computational properties as the hardness of the RLWE problem.

Throughout this chapter, we have been balancing two clashing outcomes regarding the error size. The hiding property of the commitment scheme benefits from larger errors that increase the hardness of the underlying RLWE problem, but we had to bound them in order to be able to define the correctness of the commitment and the soundness of the ZKPoKs, getting greater sizes the higher the bound has to be.

In order to enforce hiding, we have used that the probability of the tails is small enough so that the statistical distance between the two distributions (truncated and not) is again small. Besides that, there is an alternative similar strategy that allows us to relate the probability of an undesired event (breaking the hiding property) when we replace a distribution using the Rényi divergence of the two distributions instead of their statistical distance. It would require a specific analysis, more involved than the one with the statistical distance, but it would lead to bounds that might allow increasing the σ parameter without increasing the bound B (or to reduce the bound B without decreasing σ) and still preserve the hiding property. We propose it as possible future research path.

Albeit the running times of the implemented algorithms are already competitive (at least the key generation, committing and opening verifying algorithms), this is the aspect with more room for improvement. Given the academic nature of our work we choose to work with arbitrarily large integers to be able to have a broader picture as possible, but a practical implementation could just be tailored for modulus with less than 32 bit size (or the native integer size of the given machine), which would lead to a significant speed up. Hardware specific implementations could also introduce a higher level of parallelization, and we would expect that to lead to huge performance gains as well (as we have obtained with the modest parallelization already implemented).

We also want to remark that, provided that additions and multiplications are compatible with the partial FFT, one can choose to define as message spaces the quotients of R_q over each of the factors $p_1(x), p_2(x), \dots, p_d(x)$ of $x^n + 1$, and compute commitments and proofs to d polynomials of degree n/d (directly from the transformed domain) in parallel. Furthermore, it would be even more interesting to explore how the binding proof would improve if we just restrict the message space to $R_q/\langle p_1(x)\rangle$ and encode a message $m \in R_q/\langle p_1(x)\rangle$ as the polynomial $m' \in R_q$ such that $m \equiv m' \mod p_i(x)$ for i from 1 to d, that way ensuring that the difference of encodings of different messages is different modulo each $p_i(x)$. Our current proof specifically uses that the difference of two different messages is different modulo at least one of the $p_i(x)$, but the binding property could be enforced more easily if this difference is not 0 modulo each of the factors of $x^n + 1$.

This strategy might provide an interesting trade-off between the message space size and the commitments and proofs sizes and efficiency. It is out of the scope of this work as it greatly deviates from the scheme we have proposed, but should be considered for applications where the message space can be smaller than R_q.

Finally, an additional research path would be to redesign these ZKPoKs to base their security in the MLWE problem instead of the RLWE problem, provided that the most succinct lattice-based commitments and ZKPoKs to date are based on the first, and it would be interesting to analyze how practical are Stern-based techniques when dealing with the MLWE problem the same way we have done in this dissertation with the RLWE case.

We would have liked to compare in detail the performance of our scheme with the existing alternative proposals, such as the commitments schemes from Benhamouda *et al.* [19] and Baum *et al.* [17], or the many ZKPoKs for proving knowledge of the secret in a RLWE sample we have mentioned in Section 3.1.1. However, none of these proposals provide analysis as detailed as the ones we have presented through both Chapters 3 and 4, and the fact that either their analysis is only asymptotic (as it is the case of [19]), the underlying assumption is different (as it is the case of [17]), or proofs are of a different nature (as it is the case of several of the more efficient proofs mentioned in Section 3.1.1 that simply allow proving knowledge of a RLWE sample with noise restricted to binary or ternary distributions) prevents us from directly obtaining fair comparisons.

In any case we have provided an efficient and flexible implementation of both a lattice-based commitment scheme and NIZKPoKs of valid opening, linear and multiplicative relations. We have extensively discussed its tight security proofs so that secure sets of parameters could be selected. Even if the performance is not superior to existing alternatives we consider the analysis is valuable on its own as we have shown that Stern-based techniques had great room for improvement but still are at disadvantage with Fiat-Shamir with aborts alternatives. This conclusion could not have been obtained without the significant work done to bridge the gap between theoretical asymptotic proofs and tight conditions with explicit constants that really allow such comparisons once we are able to obtain quantitative results, as the ones presented in this chapter.

Appendices 4

4.A Special Unsoundness Adversaries

In Section 4.3.3 we mentioned that $(k_1, \ldots, k_\mu)$-special sound protocols are frequently vulnerable to one kind of attack that decreases the security of the Fiat-Shamir transform and forces us to do more parallel repetitions to get the desired security.

We have already hinted how such an attack would work, but for the sake of completeness we now fully specify in this appendix the unsoundness attacks against the opening and the multiplicative relation protocols (the linear one directly follows from the attack on the opening). From the description of the attacks it can then be seen that there seems no direct way to further prove greater unsoundness following this strategy exists, so it is reasonable to select parameters under the assumption that this is the most efficient attack.

Whenever the adversary is in active mode we specify which is its current guess following the notation from [12] and then, after receiving the challenge, we describe how they would continue if the guess was right and can switch to passive mode or how they could try to guess the second challenge to have an additional chance.

As [12] points out many $(k_1, \ldots, k_\mu)$-special sound protocols are also $(k_1 - 1, \ldots, k_\mu - 1)$-special unsound. The following adversary $\mathcal{A}$, described through Protocols 4.A.1 to 4.A.4, proves a single repetition of the opening protocol is $(1, 1)$-special unsound, as we claimed before.

The reader can check that, whenever the adversary enters in passive mode, the resulting conversation would always be accepting.

Protocol 4.A.1 Unsoundness Adversary ($\Gamma_1 := \{\widetilde{\alpha}\}$)

```
    $\mathcal{A}$                                                            $\mathcal{V}$
1:  $\widetilde{\alpha} \leftarrow_{R} \mathbb{Z}_q$
2:  $\tau \leftarrow \{0,1\}^{8\lceil\lambda/8\rceil}$, $\{\tau_j\}_j \leftarrow \mathsf{XOF}(\tau)$
3:  $\{\boldsymbol{f}_j\}_j \leftarrow_{R} \mathbb{Z}_q^{2nk}$
4:  $o_1, o_2 \leftarrow_{R} \{0,1\}^{8\lceil\lambda/8\rceil}$
5:  $\{\boldsymbol{e}'_j\}_j \leftarrow_{R} \mathfrak{B}_{nk}$
6:  $\boldsymbol{z} \leftarrow_{R} \mathcal{L}(\boldsymbol{a})$
7:  $s \leftarrow_{R} R_q$
8:  $c_1 \leftarrow \mathsf{Hash}(\tau \| \boldsymbol{z} - (\widetilde{\alpha}(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j(\boldsymbol{f}_j + \widetilde{\alpha}\boldsymbol{e}'_j))) \| o_1)$
9:  $c_2 \leftarrow \mathsf{Hash}(\{\pi_{\tau_j}(\boldsymbol{f}_j)\}_j \| \{\pi_{\tau_j}(\boldsymbol{e}'_j)\}_j \| o_2)$
10:                                $\xrightarrow{c_1, c_2}$
11:                                                                          $\alpha \leftarrow_{R} \mathbb{Z}_q$
```

If $\alpha = \widetilde{\alpha}$ then $\mathcal{A}$ can switch to passive mode and produce an accepting conversation following the next strategy.

Protocol 4.A.2 Unsoundness Adversary (passive mode if $\alpha = \widetilde{\alpha}$)

```
12:                                $\xleftarrow{\alpha = \widetilde{\alpha}}$
13: for $j \in 0, \ldots, \log(B)$ do
14:     $\boldsymbol{g}_j := \pi_{\tau_j}(\boldsymbol{f}_j + \alpha\boldsymbol{e}'_j)$
15:                                $\xrightarrow{\{\boldsymbol{g}_j\}_j}$
16:                                                                          $b \leftarrow_{R} \{0,1\}$
17:                                $\xleftarrow{b}$
18: if $b = 0$ then
19:     $\boldsymbol{y} := \boldsymbol{z} - (\widetilde{\alpha}(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j(\boldsymbol{f}_j + \widetilde{\alpha}\boldsymbol{e}'_j)))$
20:     $ans := (\tau, \boldsymbol{y}, s, o_1)$
21: else if $b = 1$ then
22:     for $j \in 0, \ldots, \log(B)$ do
23:         $\widetilde{\boldsymbol{e}}'_j := \pi_{\tau_j}(\boldsymbol{e}'_j)$
24:     $ans := (\{\widetilde{\boldsymbol{e}}'_j\}_j, o_2)$
25:                                $\xrightarrow{ans}$
26:                                                          if $b = 0$ then
27:                                                              $\{\tau_j\}_j \leftarrow \mathsf{XOF}(\tau)$
28:                                                              $c_1 \overset{?}{=} \mathsf{Hash}(\tau \| \boldsymbol{y} \| o_1)$
29:                                                              $\widetilde{\boldsymbol{z}} := \boldsymbol{y} + \alpha(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j \pi_{\tau_j}^{-1}(\boldsymbol{g}_j))$
30:                                                              $\widetilde{\boldsymbol{z}} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
31:                                                          else if $b = 1$ then
32:                                                              $c_2 \overset{?}{=} \mathsf{Hash}\left(\{\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j\}_j \| \{\widetilde{\boldsymbol{e}}'_j\}_j \| o_2\right)$
33:                                                              for $j \in 0, \ldots, \log(B)$ do
34:                                                                  $\widetilde{\boldsymbol{e}}'_j \overset{?}{\in} \mathfrak{B}_{nk}$
```

If $\alpha \neq \widetilde{\alpha}$ the adversary can continue in active mode and still try to guess the second challenge. Consider the guess $b = 0$.

Protocol 4.A.3 Unsoundness Adversary ($\alpha \notin \Gamma_1$ and $\Gamma_2 := \{0\}$)

```
12:                                   $\xleftarrow{\alpha\neq\widetilde{\alpha}}$
13: for $j \in 1, \dots, \log(B)$ do
14:     $\boldsymbol{g}_j \leftarrow_R \mathbb{Z}_q^{2nk}$
15: $\boldsymbol{g}'_0 \leftarrow_R \mathbb{Z}_q^{nk}$
16: $\boldsymbol{g}_0 := \pi_{\tau_0}((\alpha - \widetilde{\alpha})(\boldsymbol{c} + \boldsymbol{B}) + \phi(\boldsymbol{f}_0 + \widetilde{\alpha}\boldsymbol{e}'_0 + \sum_{j=1}^{\log(B)} 2^j(\boldsymbol{f}_j + \widetilde{\alpha}\boldsymbol{e}'_j - \pi_{\tau_j}{}^{-1}(\boldsymbol{g}_j)))\|\boldsymbol{g}'_0)$
17:                                   $\xrightarrow{\{\boldsymbol{g}_j\}_j}$
18:                                                              $b \leftarrow_R \{0, 1\}$
```

If the guess was right it can again switch to passive mode and produce an accepting conversation.

```
19:                                   $\xleftarrow{b=0}$
20: $\boldsymbol{y} := \boldsymbol{z} - (\widetilde{\alpha}(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j(\boldsymbol{f}_j + \widetilde{\alpha}\boldsymbol{e}'_j)))$
21: $ans := (\tau, \boldsymbol{y}, s, o_1)$
22:                                   $\xrightarrow{ans}$
23:                                                              $\{\tau_j\}_j \leftarrow \mathsf{XOF}(\tau)$
24:                                                              $c_1 \overset{?}{=} \mathsf{Hash}(\tau\|\boldsymbol{y}\|o_1)$
25:                    $\widetilde{\boldsymbol{z}} := \boldsymbol{y} + \alpha(\boldsymbol{c} + \mathbf{B}) - \boldsymbol{b}s - \phi(\sum_j 2^j \pi_{\tau_j}{}^{-1}(\boldsymbol{g}_j))$
26:                                                              $\widetilde{\boldsymbol{z}} \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$
```

Alternatively they could also guess $b = 1$.

Protocol 4.A.4 Unsoundness Adversary ($\alpha \notin \Gamma_1$ and $\Gamma_2 := \{1\}$)

```
12:                                   $\xleftarrow{\alpha\neq\widetilde{\alpha}}$
13: for $j \in 0, \dots, \log(B)$ do
14:     $\boldsymbol{g}_j := \pi_{\tau_j}(\boldsymbol{f}_j + \alpha\boldsymbol{e}'_j)$
15:                                   $\xrightarrow{\{\boldsymbol{g}_j\}_j}$
16:                                                              $b \leftarrow_R \{0, 1\}$
```

And once again proceed producing an accepting conversation if the guess was correct.

```
17:                                   $\xleftarrow{b=1}$
18: for $j \in 0, \dots, \log(B)$ do
19:     $\widetilde{\boldsymbol{e}}'_j := \pi_{\tau_j}(\boldsymbol{e}'_j)$
20: $ans := (\{\widetilde{\boldsymbol{e}}'_j\}_j, o_2)$
21:                                   $\xrightarrow{ans}$
22:                                  $c_2 \overset{?}{=} \mathsf{Hash}\left(\{\boldsymbol{g}_j - \alpha\widetilde{\boldsymbol{e}}'_j\}_j \| \{\widetilde{\boldsymbol{e}}'_j\}_j \| o_2\right)$
23:                                                              for $j \in 0, \dots, \log(B)$ do
24:                                                                  $\widetilde{\boldsymbol{e}}'_j \overset{?}{\in} \mathfrak{B}_{nk}$
```

Notice both in situations described in Protocols 4.A.1 and 4.A.3 the adversary can produce many different responses by just resampling the different uniformly random elements, still preserving the properties of Γ_1 and Γ_2 respectively. The opening protocol is therefore $(1,1)$-special unsound with $\Omega(q^{nk})$ responses per round. Regarding Protocol 4.A.4 there are no many responses possible, but the adversary could just always guess $b = 0$ and use Protocol 4.A.3, as it is sufficient to define a successful strategy.

It is more difficult to assess the security of the multiplicative relation protocol. Recall that the $(2,3,2)$-special soundness is that of a 7-move protocol, but we have already discussed that the transformation works better with the 5-move interpretation. Furthermore, in the 7-move second scenario an attack as the one proposed in [12] would not be applicable, because it would not satisfy the *with N responses per round* property. The attack exploits the fact that by choosing different prover responses it is possible to get different challenges for one round preserving the partial success obtained in the previous rounds. As there is no response between β and α the second would be deterministically determined by the first and the hash function. For that matter, we first just prove that the 5-move interactive version is $(1,1)$-special unsound.

Protocol 4.A.5 UNSOUNDNESS ADVERSARY mult. ($\Gamma_1 := \{(\widetilde{\alpha}, \widetilde{\beta})\}$)

```
    $\mathcal{A}$                                                        $\mathcal{V}$
1:  $\widetilde{\alpha}, \widetilde{\beta} \leftarrow_R \mathbb{Z}_q$
2:  $\widetilde{\gamma}_1 := \widetilde{\alpha},\ \widetilde{\gamma}_2 := \widetilde{\alpha},\ \widetilde{\gamma}_3 := \widetilde{\beta}$
3:  $\widetilde{m}_1, \widetilde{m}_2 \leftarrow_R R_q,\ \widetilde{m}_3 := \widetilde{m}_1 \cdot \widetilde{m}_2$
4:  for $h \in 1,2,3$ do
5:      $\tau_h \leftarrow \{0,1\}^{8\lceil\lambda/8\rceil}, \{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
6:      $\{\boldsymbol{f}_{hj}\}_j \leftarrow_R \mathbb{Z}_q^{2nk}$
7:      $\{\boldsymbol{e}'_{hj}\}_j \leftarrow_R \mathfrak{B}_{nk}$
8:      $\mu_h, s_h \leftarrow_R R_q$
9:      $\boldsymbol{z}_h := \boldsymbol{a}(\mu_h + \widetilde{\gamma}_h \widetilde{m}_h)$
10: $\mu_\times, \mu_+ \leftarrow_R R_q$
11: $m_\times := \mu_1\mu_2,\ m_+ := \mu_1\widetilde{m}_2 + \mu_2\widetilde{m}_1$
12: $o_1, o_2, o_3, o_4 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
13: $c_1 \leftarrow \mathsf{Hash}(\{\tau\}_h \| \{\boldsymbol{z}_h - (\widetilde{\gamma}_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j(\boldsymbol{f}_{hj} + \widetilde{\gamma}_h \boldsymbol{e}'_{hj})))\}_h \| o_1)$
14: $c_2 \leftarrow \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
15: $c_3 \leftarrow \mathsf{Hash}(\{\pi_{\tau_{hj}}(\boldsymbol{f}_{hj})\}_{hj} \| \{\pi_{\tau_{hj}}(\boldsymbol{e}'_{hj})\}_{hj} \| o_3)$
16: $c_4 \leftarrow \mathsf{Hash}(\mu_\times + m_\times \| \mu_+ + m_+ \| o_4)$
17:                          $\xrightarrow{c_1, c_2, c_3, c_4}$
18:                                                     $\alpha, \beta \leftarrow_R \mathbb{Z}_q$
```

If the guess was correct then the adversary can turn into passive mode and keep answering till the protocol finishes producing an accepting conversation.

Protocol 4.A.6 Unsoundness Adversary mult. (passive mode if $(\alpha, \beta) = (\widetilde{\alpha}, \widetilde{\beta})$)

```
19:                         $\xleftarrow{(\alpha,\beta)=(\widetilde{\alpha},\widetilde{\beta})}$
20: $o_5 \leftarrow_R \{0,1\}^{8\lceil \lambda/8 \rceil}$
21: $c_5 \leftarrow \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
22: for $h \in 1,2,3$ do
23:     for $j \in 0,\dots,\log(B)$ do
24:         $\boldsymbol{g}_{hj} := \pi_{\tau_{hj}}(\boldsymbol{f}_{hj} + \gamma_h \boldsymbol{e}'_{hj})$
25:                         $\xrightarrow{c_5,\{\boldsymbol{g}_{hj}\}_{hj}}$
26:                                                     $b \leftarrow_R \{0,1\}$
27:                         $\xleftarrow{b}$
28: if $b = 0$ then
29:     $t_\times := \mu_\times + m_\times,\ t_+ := \mu_+ + m_+$
30:     for $h \in 1,2,3$ do
31:         $\boldsymbol{y}_h := z_h - (\widetilde{\gamma}_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j(\boldsymbol{f}_{hj} + \widetilde{\gamma}_h \boldsymbol{e}'_{hj})))$
32:     $ans := (\{\tau_h\}_h, \{\boldsymbol{y}_h\}_h, t_\times, t_+, \{s_h\}_h, o_1, o_4, o_5)$
33: else if $b = 1$ then
34:     for $h \in 1,2,3$ do
35:         for $j \in 0,\dots,\log(B)$ do
36:             $\widetilde{\boldsymbol{e}}'_{hj} := \pi_{\tau_j}(\boldsymbol{e}'_{hj})$
37:     $ans := (\{\widetilde{\boldsymbol{e}}'_{hj}\}_{hj}, \mu_3, \mu_\times, \mu_+, o_2, o_3, o_5)$
38:                         $\xrightarrow{ans}$
39:                                     $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
40:                                     if $b = 0$ then
41:                                         for $h \in 1,2,3$ do
42:                                             $\{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
43:                     $\widetilde{z}_h := \boldsymbol{y}_h + \gamma_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j \pi_{\tau_{hj}}{}^{-1}(\boldsymbol{g}_{hj}))$
44:                     $\widetilde{z}_h \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$: Let $t_h \in R_q$ s.t. $\widetilde{z}_h = \boldsymbol{a}t_h$
45:                     $c_1 \overset{?}{=} \mathsf{Hash}(\{\tau_h\}_h \| \{\boldsymbol{y}_h\}_h \| o_1)$
46:                     $c_4 \overset{?}{=} \mathsf{Hash}(t_\times \| t_+ \| o_4)$
47:                     $c_5 \overset{?}{=} \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2 t_3 - \beta t_1 t_2 \| o_5)$
48:                                     else if $b = 1$ then
49:                     $c_2 \overset{?}{=} \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
50:                     $c_3 \overset{?}{=} \mathsf{Hash}(\{\boldsymbol{g}_{hj} - \gamma_h \widetilde{\boldsymbol{e}}'_{hj}\}_{hj} \| \{\widetilde{\boldsymbol{e}}'_{hj}\}_{hj} \| o_3)$
51:                     $c_5 \overset{?}{=} \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
52:                                         for $h \in 1,2,3$ do
53:                                             for $j \in 0,\dots,\log(B)$ do
54:                                                 $\widetilde{\boldsymbol{e}}'_{hj} \overset{?}{\in} \mathfrak{B}_{nk}$
```

If that is not the case then the prover can continue in active mode trying to guess the second challenge.

Protocol 4.A.7 Unsoundness Adversary mult. ($(\alpha,\beta) \notin \Gamma_1$ and $\Gamma_2 := \{0\}$)

```
19:                                   $\xleftarrow{(\alpha,\beta)\neq(\widetilde{\alpha},\widetilde{\beta})}$
20: $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
21: $t_\times := \mu_\times + m_\times,\ t_+ := \mu_+ + m_+$
22: $o_5 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
23: $c_5 \leftarrow \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2(\mu_3 + \widetilde{\beta}\widetilde{m}_1\widetilde{m}_2) - \beta(\mu_1 + \widetilde{\alpha}\widetilde{m}_1)(\mu_2 + \widetilde{\alpha}\widetilde{m}_2)\|o_5)$
24: for $h \in 1,2,3$ do
25:     for $j \in 1,\dots,\log(B)$ do
26:         $\boldsymbol{g}_{hj} \leftarrow_R \mathbb{Z}_q^{2nk}$
27:     $\boldsymbol{g}'_{h0} \leftarrow_R \mathbb{Z}_q^{nk}$
28:     $\boldsymbol{g}_{h0} := \pi_{\tau_{h0}}((\gamma_h - \widetilde{\gamma}_h)(\boldsymbol{c}_h + \boldsymbol{B}) + \phi(\boldsymbol{f}_{h0} + \widetilde{\gamma}_h\boldsymbol{e}'_{h0} + \sum_{j=1}^{\log(B)} 2^j(\boldsymbol{f}_{hj} + \widetilde{\gamma}_h\boldsymbol{e}'_{hj} - \pi^{-1}_{\tau_{hj}}(\boldsymbol{g}_{hj})))\|\boldsymbol{g}'_{h0})$
29:                                   $\xrightarrow{c_5,\{\boldsymbol{g}_{hj}\}_{hj}}$
30:                                                                          $b \leftarrow_R \{0,1\}$
```

If the guess was right it can again switch to passive mode and produce an accepting conversation.

```
31:                                   $\xleftarrow{b=0}$
32: for $h \in 1,2,3$ do
33:     $\boldsymbol{y}_h := \boldsymbol{z}_h - (\widetilde{\gamma}_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j(\boldsymbol{f}_{hj} + \widetilde{\gamma}_h\boldsymbol{e}'_{hj})))$
34: $ans := (\{\tau_h\}_h, \{\boldsymbol{y}_h\}_h, t_\times, t_+, \{s_h\}_h, o_1, o_4, o_5)$
35:                                   $\xrightarrow{ans}$
36:                                                                          for $h \in 1,2,3$ do
37:                                                                          $\{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
38:                                      $\widetilde{\boldsymbol{z}}_h := \boldsymbol{y}_h + \gamma_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j\pi_{\tau_{hj}}{}^{-1}(\boldsymbol{g}_{hj}))$
39:                                                    $\widetilde{\boldsymbol{z}}_h \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$: Let $t_h \in R_q$ s.t. $\widetilde{\boldsymbol{z}}_h = \boldsymbol{a}t_h$
40:                                                                $c_1 \overset{?}{=} \mathsf{Hash}(\{\tau_h\}_h\|\{\boldsymbol{y}_h\}_h\|o_1)$
41:                                                                     $c_4 \overset{?}{=} \mathsf{Hash}(t_\times\|t_+\|o_4)$
42:                                                $c_5 \overset{?}{=} \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2 t_3 - \beta t_1 t_2\|o_5)$
```

Observe now it is possible to produce different answers ($O(2^\lambda)$ many) just by resampling the opening randomness of the commitment (it was not possible with the attack to the opening or linear protocols). Then choosing $\Gamma_2 := \{0\}$ we do have sufficiently many responses per round.

Alternatively they could also guess $b = 1$.

Protocol 4.A.8 Unsoundness Adversary mult. ($(\alpha,\beta) \notin \Gamma_1$ and $\Gamma_2 := \{1\}$)

```
19:                                                  $\xleftarrow{(\alpha,\beta)\neq(\widetilde{\alpha},\widetilde{\beta})}$
20: $o_5 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
21: $c_5 \leftarrow \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
22: $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
23: for $h \in 1,2,3$ do
24:     for $j \in 0,\dots,\log(B)$ do
25:         $g_{hj} := \pi_{\tau_{hj}}(f_{hj} + \gamma_h e'_{hj})$
26:                                                  $\xrightarrow{c_5,\{g_{hj}\}_{hj}}$
27:                                                                          $b \leftarrow_R \{0,1\}$
```

And once again proceed producing an accepting conversation if the guess was correct.

```
28:                                                  $\xleftarrow{b=1}$
29: for $h \in 1,2,3$ do
30:     for $j \in 0,\dots,\log(B)$ do
31:         $\widetilde{e}'_{hj} := \pi_{\tau_{hj}}(e'_{hj})$
32: $ans := (\{\widetilde{e}'_{hj}\}_{hj}, \mu_3, \mu_\times, \mu_+, o_2, o_3, o_5)$
33:                                                  $\xrightarrow{ans}$
34:                                                  $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
35:                                                  $c_2 \stackrel{?}{=} \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
36:                                                  $c_3 \stackrel{?}{=} \mathsf{Hash}(\{g_{hj} - \gamma_h \widetilde{e}'_{hj}\}_{hj} \| \{\widetilde{e}'_{hj}\}_{hj} \| o_3)$
37:                                                  $c_5 \stackrel{?}{=} \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
38:                                                  for $h \in 1,2,3$ do
39:                                                      for $j \in 0,\dots,\log(B)$ do
40:                                                          $\widetilde{e}'_{hj} \stackrel{?}{\in} \mathfrak{B}_{nk}$
```

So far the proposed attacks work for arbitrary statements and the adversary needs no additional secret information. The restriction obtained for the multiplicative version would be looser than the previous one because $^{l_1}/_{|C_1|} = q^{-2}$, and therefore the adversary success probability would be smaller. Nevertheless, we can find yet another attack with the previous success probability if the adversary only fakes part of the proof.

Notice that the multiplicative protocol aims to prove two different facts, knowledge of three valid openings to m_1, m_2, m_3 and, additionally, a special relation between the committed messages as $m_3 = m_1 \cdot m_2$. If the adversary does indeed know a valid opening to the initial pair of commitments they then can only fake the part related to the third commitment with a higher success probability, that is again the probability of successfully attacking the opening protocol. That is because for this particular scenario the protocol has $(q,1)$-out-of-$(q^2,2)$-special unsoundness (which is by all means equivalent to the $(1,1)$-out-of-$(q,2)$-special unsoundness).

Protocol 4.A.9 UNSOUNDNESS ADV. mult. special case ($\Gamma_1 := \{(\alpha, \widetilde{\beta}) | \alpha \in \mathbb{Z}_q\}$)

```
Let $m_1, m_2 \in R_q$ and $\boldsymbol{c}_3 \in R_q^k$
$r_1, r_2 \leftarrow_R R_q,\ \boldsymbol{e}_1, \boldsymbol{e}_2 \leftarrow_R \left(D^n_{\sigma,B}\right)^k$
$\boldsymbol{c}_1 := \boldsymbol{a}m_1 + \boldsymbol{b}r_1 + \boldsymbol{e}_1$
$\boldsymbol{c}_2 := \boldsymbol{a}m_2 + \boldsymbol{b}r_2 + \boldsymbol{e}_2$

    $\mathcal{A}(\{\boldsymbol{c}_h\}_h, m_1, m_2, r_1, r_2, \boldsymbol{e}_1, \boldsymbol{e}_2)$                    $\mathcal{V}(\{\boldsymbol{c}_h\}_h)$
1:  $\widetilde{\beta} \leftarrow_R \mathbb{Z}_q$
2:  $\{\boldsymbol{e}'_{1j}\}_j \leftarrow \mathsf{expand}(\boldsymbol{e}_1)$
3:  $\{\boldsymbol{e}'_{2j}\}_j \leftarrow \mathsf{expand}(\boldsymbol{e}_2)$
4:  $\{\boldsymbol{e}'_{3j}\}_j \leftarrow_R \mathfrak{B}_{nk}$
5:  for $h \in 1, 2, 3$ do
6:      $\tau_h \leftarrow \{0,1\}^{8\lceil\lambda/8\rceil}, \{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
7:      $\{\boldsymbol{f}_{hj}\}_j \leftarrow_R \mathbb{Z}_q^{2nk}$
8:      $\mu_h \leftarrow_R R_q$
9:  $\rho_1, \rho_2 \leftarrow_R R_q$
10: $s_3 \leftarrow_R R_q$
11: $\boldsymbol{z}_3 := \boldsymbol{a}(\mu_3 + \widetilde{\beta} m_1 m_2)$
12: $\boldsymbol{y}_1 := \boldsymbol{a}\mu_1 + \boldsymbol{b}\rho_1 + \phi(\sum_j 2^j \boldsymbol{f}_{1j})$
13: $\boldsymbol{y}_2 := \boldsymbol{a}\mu_2 + \boldsymbol{b}\rho_2 + \phi(\sum_j 2^j \boldsymbol{f}_{2j})$
14: $\boldsymbol{y}_3 := \boldsymbol{z}_3 - (\widetilde{\beta}(\boldsymbol{c}_3 + \mathbf{B}) - \boldsymbol{b}s_3 - \phi(\sum_j 2^j(\boldsymbol{f}_{3j} + \widetilde{\beta}\boldsymbol{e}'_{3j})))$
15: $\mu_\times, \mu_+ \leftarrow_R R_q$
16: $m_\times := \mu_1\mu_2,\ m_+ := \mu_1 m_2 + \mu_2 m_1$
17: $o_1, o_2, o_3, o_4 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
18: $c_1 \leftarrow \mathsf{Hash}(\{\tau\}_h \| \{\boldsymbol{y}_h\}_h \| o_1)$
19: $c_2 \leftarrow \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
20: $c_3 \leftarrow \mathsf{Hash}(\{\pi_{\tau_{hj}}(\boldsymbol{f}_{hj})\}_{hj} \| \{\pi_{\tau_{hj}}(\boldsymbol{e}'_{hj})\}_{hj} \| o_3)$
21: $c_4 \leftarrow \mathsf{Hash}(\mu_\times + m_\times \| \mu_+ + m_+ \| o_4)$
22:                          $\xrightarrow{c_1, c_2, c_3, c_4}$
23:                                                            $\alpha, \beta \leftarrow_R \mathbb{Z}_q$
```

If the $\widetilde{\beta}$ guess was lucky then the adversary can turn to passive mode and correctly respond till the end of the protocol.

Protocol 4.A.10 Unsoundness Adv. mult. special case (passive mode if $\beta = \widetilde{\beta}$)

```
                                   $\xleftarrow{(\alpha,\beta=\widetilde{\beta})}$
$o_5 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
$c_5 \leftarrow \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
$\gamma_1 := \alpha, \quad \gamma_2 := \alpha, \quad \gamma_3 := \beta$
for $h \in 1,2,3$ do
    for $j \in 0,\dots,\log(B)$ do
        $\boldsymbol{g}_{hj} := \pi_{\tau_{hj}}(\boldsymbol{f}_{hj} + \gamma_h \boldsymbol{e}'_{hj})$
                                   $\xrightarrow{c_5,\{\boldsymbol{g}_{hj}\}_{hj}}$
                                                                  $b \leftarrow_R \{0,1\}$
                                   $\xleftarrow{b}$
if $b = 0$ then
    $t_\times := \mu_\times + m_\times,\ t_+ := \mu_+ + m_+$
    $s_1 := \rho_1 + \alpha r_1$
    $s_2 := \rho_2 + \alpha r_2$
    $ans := (\{\tau_h\}_h, \{\boldsymbol{y}_h\}_h, t_\times, t_+, \{s_h\}_h, o_1, o_4, o_5)$
else if $b = 1$ then
    for $h \in 1,2,3$ do
        for $j \in 0,\dots,\log(B)$ do
            $\widetilde{\boldsymbol{e}}'_{hj} := \pi_{\tau_j}(\boldsymbol{e}'_{hj})$
    $ans := (\{\widetilde{\boldsymbol{e}}'_{hj}\}_{hj}, \mu_3, \mu_\times, \mu_+, o_2, o_3, o_5)$
                                   $\xrightarrow{ans}$
                                                  $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
                                                  if $b = 0$ then
                                                      for $h \in 1,2,3$ do
                                                          $\{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
                                                          $\widetilde{\boldsymbol{z}}_h := \boldsymbol{y}_h + \gamma_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j \pi_{\tau_{hj}}{}^{-1}(\boldsymbol{g}_{hj}))$
                                                          $\widetilde{\boldsymbol{z}}_h \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$: Let $t_h \in R_q$ s.t. $\widetilde{\boldsymbol{z}}_h = \boldsymbol{a}t_h$
                                                      $c_1 \overset{?}{=} \mathsf{Hash}(\{\tau_h\}_h \| \{\boldsymbol{y}_h\}_h \| o_1)$
                                                      $c_4 \overset{?}{=} \mathsf{Hash}(t_\times \| t_+ \| o_4)$
                                                      $c_5 \overset{?}{=} \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2 t_3 - \beta t_1 t_2 \| o_5)$
                                                  else if $b = 1$ then
                                                      $c_2 \overset{?}{=} \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
                                                      $c_3 \overset{?}{=} \mathsf{Hash}(\{\boldsymbol{g}_{hj} - \gamma_h \widetilde{\boldsymbol{e}}'_{hj}\}_{hj} \| \{\widetilde{\boldsymbol{e}}'_{hj}\}_{hj} \| o_3)$
                                                      $c_5 \overset{?}{=} \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
                                                      for $h \in 1,2,3$ do
                                                          for $j \in 0,\dots,\log(B)$ do
                                                              $\widetilde{\boldsymbol{e}}'_{hj} \overset{?}{\in} \mathfrak{B}_{nk}$
```

If the guess was unlucky the adversary can still try to guess the next challenge.

Protocol 4.A.11 Unsoundness Adv. mult. special case ($(\alpha,\beta) \notin \Gamma_1$ and $\Gamma_2 := \{0\}$)

```
24:                                  $\xleftarrow{(\alpha,\beta\neq\widetilde{\beta})}$
25: $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
26: $t_\times := \mu_\times + m_\times,\ t_+ := \mu_+ + m_+$
27: $o_5 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
28: $c_5 \leftarrow \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2(\mu_3 + \widetilde{\beta} m_1 m_2) - \beta(\mu_1 + \alpha m_1)(\mu_2 + \alpha m_2)\|o_5)$
29: for $h \in 1,2$ do
30:     for $j \in 0,\dots,\log(B)$ do
31:         $\boldsymbol{g}_{hj} := \pi_{\tau_{hj}}(\boldsymbol{f}_{hj} + \alpha\boldsymbol{e}'_{hj})$
32: for $j \in 1,\dots,\log(B)$ do
33:     $\boldsymbol{g}_{3j} \leftarrow_R \mathbb{Z}_q^{2nk}$
34: $\boldsymbol{g}'_{30} \leftarrow_R \mathbb{Z}_q^{nk}$
35: $\boldsymbol{g}_{30} := \pi_{\tau_{30}}((\beta - \widetilde{\beta})(\boldsymbol{c}_3 + \boldsymbol{B}) + \phi(\boldsymbol{f}_{30} + \widetilde{\beta}\boldsymbol{e}'_{30} + \sum_{j=1}^{\log(B)} 2^j(\boldsymbol{f}_{3j} + \widetilde{\beta}\boldsymbol{e}'_{3j} - \pi^{-1}_{\tau_{3j}}(\boldsymbol{g}_{3j})))\|\boldsymbol{g}'_{30})$
36:                                  $\xrightarrow{c_5,\{\boldsymbol{g}_{hj}\}_{hj}}$
37:                                                                  $b \leftarrow_R \{0,1\}$
```

If the guess was right it can again switch to passive mode and produce an accepting conversation.

```
38:                                  $\xleftarrow{b=0}$
39: $s_1 := \rho_1 + \alpha r_1$
40: $s_2 := \rho_2 + \alpha r_2$
41: $\boldsymbol{y}_1 := \boldsymbol{a}\mu_1 + \boldsymbol{b}\rho_1 + \phi(\sum_j 2^j \boldsymbol{f}_{1j})$
42: $\boldsymbol{y}_2 := \boldsymbol{a}\mu_2 + \boldsymbol{b}\rho_2 + \phi(\sum_j 2^j \boldsymbol{f}_{2j})$
43: $\boldsymbol{y}_3 := z_3 - (\widetilde{\beta}_3(\boldsymbol{c}_3 + \mathbf{B}) - \boldsymbol{b}s_3 - \phi(\sum_j 2^j(\boldsymbol{f}_{3j} + \widetilde{\beta}\boldsymbol{e}'_{3j})))$
44: $ans := (\{\tau_h\}_h, \{\boldsymbol{y}_h\}_h, t_\times, t_+, \{s_h\}_h, o_1, o_4, o_5)$
45:                                  $\xrightarrow{ans}$
46:                                                                  for $h \in 1,2,3$ do
47:                                                                      $\{\tau_{hj}\}_j \leftarrow \mathsf{XOF}(\tau_h)$
48:                                      $\widetilde{\boldsymbol{z}}_h := \boldsymbol{y}_h + \gamma_h(\boldsymbol{c}_h + \mathbf{B}) - \boldsymbol{b}s_h - \phi(\sum_j 2^j \pi_{\tau_{hj}}{}^{-1}(\boldsymbol{g}_{hj}))$
49:                                      $\widetilde{\boldsymbol{z}}_h \overset{?}{\in} \mathcal{L}(\boldsymbol{a})$: Let $t_h \in R_q$ s.t. $\widetilde{\boldsymbol{z}}_h = \boldsymbol{a}t_h$
50:                                                      $c_1 \overset{?}{=} \mathsf{Hash}(\{\tau_h\}_h\|\{\boldsymbol{y}_h\}_h\|o_1)$
51:                                                      $c_4 \overset{?}{=} \mathsf{Hash}(t_\times\|t_+\|o_4)$
52:                                      $c_5 \overset{?}{=} \mathsf{Hash}(\beta t_\times + \alpha\beta t_+ + \alpha^2 t_3 - \beta t_1 t_2\|o_5)$
```

Protocol 4.A.12 Unsoundness Adv. mult. special case ($(\alpha,\beta) \notin \Gamma_1$ and $\Gamma_2 := \{1\}$)

```
24:                                  $\xleftarrow{(\alpha,\beta)\neq(\widetilde{\alpha},\widetilde{\beta})}$
25: $o_5 \leftarrow_R \{0,1\}^{8\lceil\lambda/8\rceil}$
26: $c_5 \leftarrow \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
27: $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
28: for $h \in 1,2,3$ do
29:     for $j \in 0,\ldots,\log(B)$ do
30:         $g_{hj} := \pi_{\tau_{hj}}(f_{hj} + \gamma_h e'_{hj})$
31:                                  $\xrightarrow{c_5,\{g_{hj}\}_{hj}}$
32:                                                          $b \leftarrow_R \{0,1\}$
```

And once again proceed producing an accepting conversation if the guess was correct.

```
33:                                  $\xleftarrow{b=1}$
34: for $h \in 1,2,3$ do
35:     for $j \in 0,\ldots,\log(B)$ do
36:         $\widetilde{e}'_{hj} := \pi_{\tau_{hj}}(e'_{hj})$
37: $ans := (\{\widetilde{e}'_{hj}\}_{hj}, \mu_3, \mu_\times, \mu_+, o_2, o_3, o_5)$
38:                                  $\xrightarrow{ans}$
39:                                                          $\gamma_1 := \alpha,\ \gamma_2 := \alpha,\ \gamma_3 := \beta$
40:                                                          $c_2 \overset{?}{=} \mathsf{Hash}(\mu_3 \| \mu_\times \| \mu_+ \| o_2)$
41:                                                          $c_3 \overset{?}{=} \mathsf{Hash}(\{g_{hj} - \gamma_h \widetilde{e}'_{hj}\}_{hj} \| \{\widetilde{e}'_{hj}\}_{hj} \| o_3)$
42:                                                          $c_5 \overset{?}{=} \mathsf{Hash}(\beta\mu_\times + \alpha\beta\mu_+ + \alpha^2\mu_3 \| o_5)$
43:                                                          for $h \in 1,2,3$ do
44:                                                              for $j \in 0,\ldots,\log(B)$ do
45:                                                                  $\widetilde{e}'_{hj} \overset{?}{\in} \mathfrak{B}_{nk}$
```

Observe that the analysis on the success probability only depends on $l_1/|C_1|$ and $l_2/|C_2|$, and these two are equal for a $(q,1)$-out-of-$(q^2,2)$-special unsoundness attack as they were for a $(1,1)$-out-of-$(q,2)$-special unsoundness attack, so we get again the same condition for the number of repetitions.

4.B Script for Finding Parameters

The prototype implementation of the commitment scheme and the companion NIZKPoKs publicly available in [90] contains several thousand lines of code. We have developed it, to the best of our knowledge, trying to write code as readable and well-documented as possible. Provided that most of the tasks of the linear and multiplicative protocols are shared with the opening proof we have structured everything following a modular design, so each small computation is enclosed in its own function that can be later reused.

Nevertheless, a detailed description of each of the functions would be overly extensive to include it in this dissertation. On the contrary, the optimal parameter search script is succinct enough to meticulously describe it here.

The following lines of code, Appendix 4.B, are the contents of the `sagemath` script from the `generate_params.sage` file in the GitHub repository [90]. Though lines 1 – 7 the required libraries (as the Lattice Estimator [8]) are imported, and a binary logarithm function with sufficient precision is defined using `sage`'s internal functions.

From lines 10 – 25 the script verifies that the input parameters n, q and d introduced by the user when calling the script satisfy the necessary conditions. That is, n should be a power of 2, q has to be a prime number, d should be another power of 2 smaller or equal than n, and finally we should have $q \equiv 2d + 1 \pmod{4d}$. If any of these conditions fails then the script aborts showing a descriptive error message.

Then, from lines 28 – 40 auxiliary functions as vecBoundedPrToBoundedPr, sigmaFromB or bitsec are defined, following the descriptions from Section 1.4.

Following this, in lines 65 – 101, we just implement the functions bestK, bestSigma and bestB, as defined in Section 4.5. The other two remaining functions bestDeltaOL and bestDeltaM directly compute deltaOLinitialCandidate and deltaMinitialCandidate as defined in Section 4.5 and increases them one by one until they satisfy eq. (4.4).

Finally, following the procedure defined in Section 4.5, a full secure set of parameters $(\lambda, n, q, d, k, \sigma, B, \delta_{OL}, \delta_M)$ is computed. These values are printed to the console and saved to a file.

The only interesting trick that we use is to define several *security levels* from 0 to 2 for the bitsec auxiliary function. In level 0 only the most basic attacks against the LWE problem are considered, we then include estimations with more involved attacks with level 1 and finally use all possible optimizations with level 2. With this strategy we can only trust that the scheme is secure if it achieves λ bits of security when bitsec is called with level 2. However, the lower levels help us to discard what possible parameters are insecure without the need of the computationally heavy tests used by [8] when we want to consider optimized attacks. Only when it has surpassed the test with one level we continue to the following, achieving a great speed-up in the parameter computation process.

Listing 4.1 Optimal parameter search script

```
from latticeEstimator.estimator import * # to estimate the
    RLWE hardness
import csv # to write the parameters into a csv file
import miller_rabin # to check if an integer is prime
from sage.functions.log import logb # to compute binary
    logarithms of large integers

def log2(x):
    return numerical_approx(logb(x,2))


# INPUT PARAMETERS TESTS
if not (n and (not(n & (n - 1))) ):
    print("n␣=␣{0:d}␣is␣not␣a␣power␣of␣2".format(n))
    return

if not miller_rabin.miller_rabin(int(q)):
    print("q␣=␣{0:d}␣is␣not␣a␣prime␣number".format(q))
    return

if not ((d and (not(d & (d - 1)))) and n>=d and d>1):
    print("d␣=␣{0:d}␣is␣not␣a␣power␣of␣2␣such␣that␣n␣>=␣d␣>␣
        1".format(d))
    return

if (q - (2*d+1))%(4*d):
    print("q␣=␣{0:d}␣is␣not␣equivalent␣to␣2d+1␣mod␣4d␣for␣d
        ={1:d}".format(q,d))
    return

# AUXILIARY FUNCTIONS
def vecBoundedPrToBoundedPr(b,d):
    # outputs a such that some event happens at least once
    # in d samples with a probability of at most 2^-b
```

```python
    # if the probability of a single sample is at most 2^-a
    return b + log2(d)

def sigmaFromB(a,B):
    # outputs sigma such that a sample from D_sigma
    # has absolute value greater than B
    # with a probability lower or equal to 2^-a
    return B*math.sqrt(log2(math.e)/(2*(a+1)))

def bitsec(params,estimateLevel=1):
    # estimates the hardness of a RLWE problem
    if estimateLevel == 0:
        try: x = LWE.primal_usvp(params)
        except: x = {'rop': float('inf')}
        try: y = LWE.dual(params)
        except: y =  {'rop': float('inf')}
        if math.isinf(min(x['rop'],y['rop'])):
            return 0
        bits = log2(min(x['rop'],y['rop']))
    elif estimateLevel == 1:
        try: x = LWE.primal_usvp(params)
        except: x = {'rop': float('inf')}
        try: y = LWE.dual_hybrid(params, mitm_optimization=
            True)
        except: y = {'rop': float('inf')}
        try: z = LWE.primal_bdd(params, red_shape_model="gsa
            ")
        except: z = {'rop': float('inf')}
        if math.isinf(min(x['rop'],y['rop'],z['rop'])):
            return 0
        bits = log2(min(x['rop'],y['rop'],z['rop']))
    else:
        try: x = LWE.estimate(params)
        except: x = {'Error':{'rop': float('inf')}}
        if math.isinf(min([x[y]['rop'] for y in x])):
            return 0
        bits = log2(min([x[y]['rop'] for y in x]))
    return bits
```

```

# FUNCTIONS COMPUTING OPTIMAL PARAMETERS
def bestDeltaOL(lamb,q):
    delta = ceil(lamb/(logb(2*q/(q+1),2)))
    while not ((2*lamb - 1)*log2((2*lamb - 1)/(delta*(1-1/q)
        )) + (delta - 2*lamb + 1)*log2(q*(delta-2*lamb+1)/
        delta) >= 2*lamb):
        delta += 1
    return delta

def bestDeltaM(lamb,q):
    delta = ceil(lamb/(logb(2*q*q/(q*q+3*q-2),2)))
    while not ((2*lamb - 1)*log2((2*lamb - 1)/(delta*(1-1/q)
        )) + (delta - 2*lamb + 1)*log2(q*(delta-2*lamb+1)/
        delta) >= 2*lamb):
        delta += 1
    return delta

def bestK(lamb,n,q,d,B):
    return ceil((lamb+2*n*log2(q))/(n*(log2(q)/d - log2(4*B
        -1))))

def bestSigma(lamb,n,q,d,B,k,estimateLevel=1):
    sigma = sigmaFromB(a=vecBoundedPrToBoundedPr(b=lamb,d=k*
        n),B=(B-1))
    # we truncate it to two decimals
    sigma = float(floor(sigma*100)/100)
    params = LWE.Parameters(n=n, q=q, Xs=ND.UniformMod(q),Xe
        =ND.DiscreteGaussian(sigma),m=k*n)
    bits = bitsec(params,estimateLevel)
    if bits >= lamb: return sigma
    else: return None

def bestB(lamb,n,q,d,estimateLevel=1,minB=2):
    B = minB
    sigma = None
    while (sigma is None) and (d < log2(q)/(log2(4*B-1))):
        k = bestK(lamb,n,q,d,B)
        sigma = bestSigma(lamb,n,q,d,B,k,estimateLevel)
        if sigma is None:
            B *= 2
    if not (d < log2(q)/(log2(4*B-1))):
        return None
    else:
        return B

# COMPUTING THE PARAMETERS
B = 2
for level in range(3):
    B = bestB(lamb,n,q,d,estimateLevel=level,minB=B)
    if B is None:
        print("\nNot found")
        break

if B is not None:
    k = bestK(lamb,n,q,d,B)
    sigma = bestSigma(lamb,n,q,d,B,k,estimateLevel=2)
    deltaOL = bestDeltaOL(lamb,q)
    deltaM = bestDeltaM(lamb,q)
    print([lamb,n,q,d,k,sigma,B,deltaOL,deltaM])
    with open(args.output, 'a') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow([lamb,n,q,d,k,sigma,B,deltaOL,
            deltaM])
```

4.C Additional Results

We have chosen to only present in Section 4.6 results from a selected subset of parameters, to keep the analysis concise. The mere existence of *worse* sets of parameters has no special theoretical interest, neither practical consequences. Nevertheless, we can still extract some insights that are relevant enough to devote this last appendix to these additional parameters.

As we have already said, in order to get meaningful plots, we have only benchmarked the algorithms with sets that produce multiplicative NIZKPoKs of at most 512 MB, because after some point the sizes blow up. Albeit that, we have also computed secure sets of parameters for larger q's. Given that the sizes only depend on $\lceil q \rceil$ for these additional sets of parameters we have only tried one $2^b < q < 2^{b+1}$ for each $b \in \mathbb{N}$ (notice some restrictions are related to q itself and not its bit-size, so this is again not an exhaustive search).

We observe that, for a fixed n, we can find secure sets of parameters for many moduli q, but only until some size, from which no secure set of parameters satisfying the desired constraints seems to exist. Provided that we compute the hardness of the problem using the Lattice Estimator as a black box, we can only explain it pointing out that the log of the optimum B seems to grow faster than linearly in the log of q to preserve the hardness of the underlying problem, up until a point where eq. (3.3) cannot be honored. This is important because in theoretical proposals it is usual to say that some condition will be fulfilled for sufficiently large modulus, but it can be the case that, when considering all conditions at the same time, no secure set of parameters with such large modulus exists satisfying all of them.

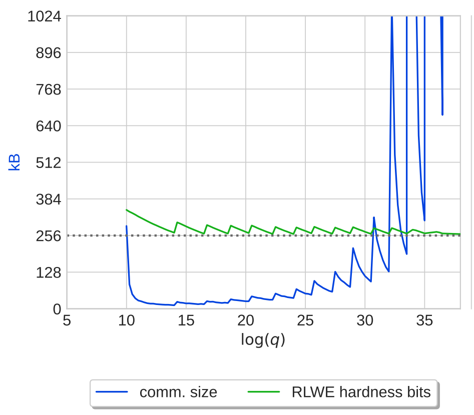
We show the sizes defined by all these additional parameters in Figure 4.C.1, again truncating the y-axis because some sets are only secure with absurdly large k, which would imply much greater sizes. These additional sets are only interesting to see the general picture of the parameter space, not because any of these are of practical interest.

We have not stopped searching for additional parameters at the first q such that no secure set existed, because we cannot ensure that no more secure sets exist for even larger q's. We have, however, discarded the existence of additional secure sets of parameters that would yield to smaller commitment sizes than the ones presented in Table 4.2. From eq. (3.2) we can see that $k \geq 4$, and this allows us to define a lower bound on the sizes of the commitments that we would obtain before explicitly computing them. We have tested different q up to the point when this lower bound (not tight at all) is already greater than the commitments we have obtained in Table 4.2 and stopped there. No additional secure sets have been found. We have also used a similar approach to discard the existence of better sets of parameters with $n = 256$ (for which no secure set has been found either) or $n > 1024$ (for which secure sets can be found but with greater commitment sizes).

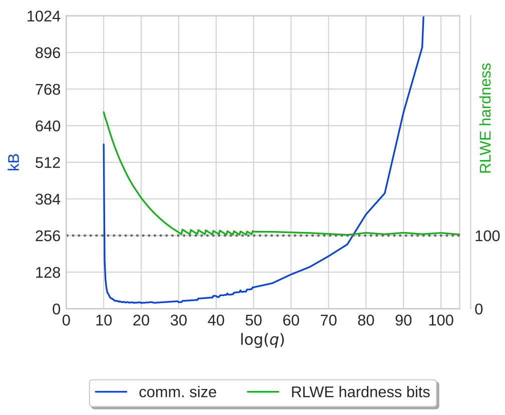
Besides these additional parameters regarding n and q we also explore here the existing trade-off regarding parameter d. The product of polynomials computation using the partial FFT multiplication algorithm is more efficient the higher the d, but we can see that secure sets of parameters only exists with $d = 4$ (besides the already explored $d = 2$). The more restrictive conditions imply a greater k that ends up producing not only even greater sizes but also slower times (see Figure 4.C.2). Therefore, this trade-off is not actually useful for this scheme because of the restrictions implied by the current design of the binding property.

To ease comparisons with other schemes we present in Table 4.C.1 a version of Table 4.3 detailing the performance of the protocols in millions of processor cycles.

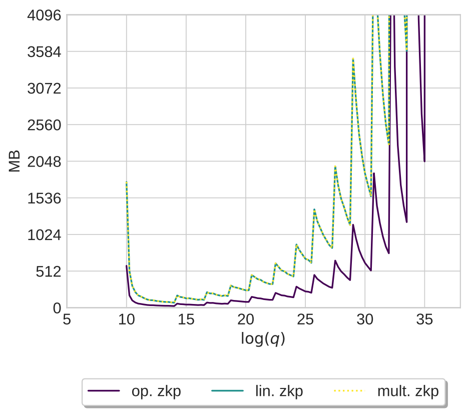
Figure 4.C.1 Additional commitment and NIZKPoKs sizes



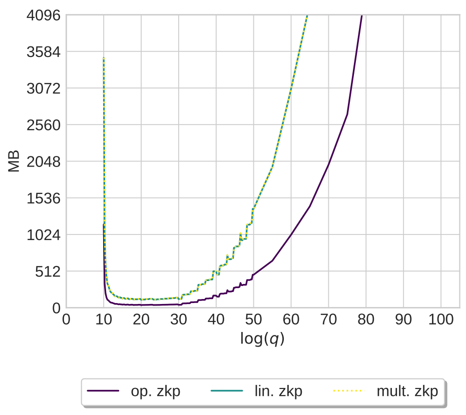
(a) $\lambda = 100, \ n = 512, \ d = 2$



(b) $\lambda = 100, \ n = 1024, \ d = 2$

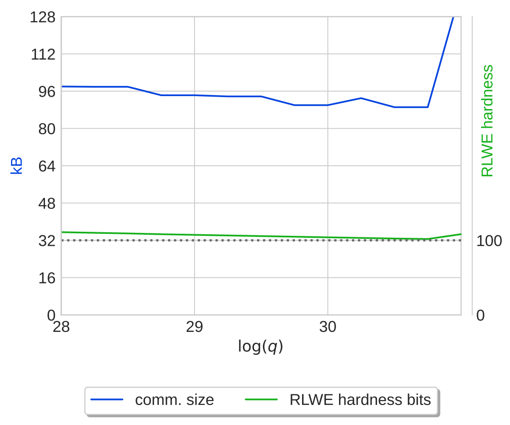


(c) $\lambda = 100, \ n = 512, \ d = 2$

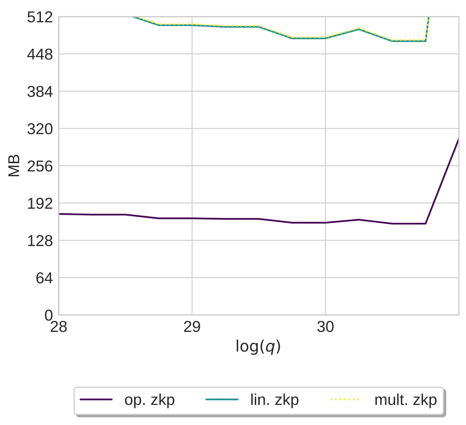


(d) $\lambda = 100, \ n = 1024, \ d = 2$

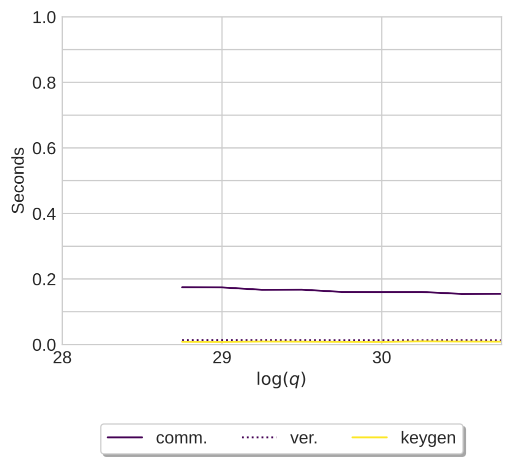
Figure 4.C.2 Commitment and NIZKPoKs sizes and times for $d = 4$



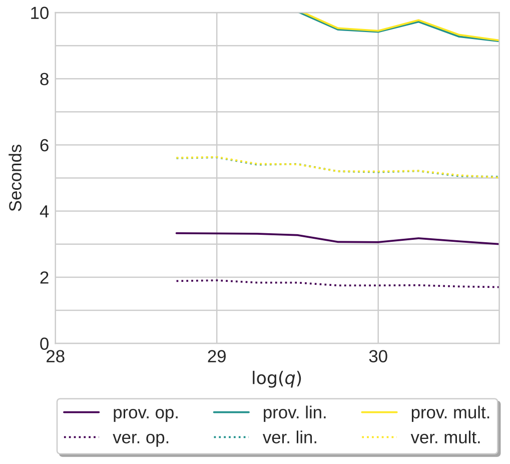
(a) $\lambda = 100, \ n = 1024, \ d = 4$



(b) $\lambda = 100, \ n = 1024, \ d = 4$



(c) $\lambda = 100, \ n = 1024, \ d = 4$



(d) $\lambda = 100, \ n = 1024, \ d = 4$

Table 4.C.1 Running time of the best parameters (in mill. of processor cycles)

n	q	com.	ver.	key.	$\mathcal{P}_{op}$	$\mathcal{V}_{op}$	$\mathcal{P}_{lin}$	$\mathcal{V}_{lin}$	$\mathcal{P}_{mult}$	$\mathcal{V}_{mult}$
512	16381	147.37	6.96	9.47	2368.29	1212.89	2817.98	3658.69	2580.35	3668.70
1024	1048573	168.67	8.86	19.12	2800.07	1525.85	2930.43	428.74	3988.30	342.63
1024	11863253	148.36	8.22	19.89	2499.80	1327.29	3254.21	3992.30	3094.35	4042.38
1024	16777213	148.51	8.25	19.36	2443.48	1318.44	3000.48	3966.21	2886.83	3980.50
1024	67108837	148.04	8.47	19.10	2587.04	1474.27	3564.68	423.43	3347.62	247.97
1024	1073741789	131.37	10.55	25.29	2386.08	1393.49	2868.73	3982.53	2734.25	4098.82
1024	1276901389	132.52	11.38	27.13	2492.12	1386.63	3199.94	3852.37	3102.74	3286.20
1024	1518500213	132.37	11.36	27.24	2457.95	1386.97	3053.65	3870.64	3002.06	3243.41
1024	1805811253	133.07	11.37	26.91	2439.11	1384.01	3016.04	3868.50	2903.86	3374.66

Acronyms

BKZ *Blockwise Korkine-Zolotarev*. 32

CRT *Chinese Remainder Theorem*. 88, 94–98, 101, 102, 104, 106

CSPRNG *Cryptographically Secure Pseudorandom Number Generator*. 21, 159, 184, 185

CVP *Closest Vector Problem*. 26, 29

DDH *Decisional Diffie-Hellman*. 4

DLog *Discrete Logarithm*. 4, 5, 18, 47

DZ *Discrete Ziggurat*. 187

FFT *Fast Fourier Transform*. 71, 73, 77, 78, 83, 92, 94, 100, 101, 106, 122, 155, 189, 202, 221

FHE *Fully Homomorphic Encryption*. 24, 110

ISIS *Inhomogeneous Short Integer Solution*. 37, 47, 50, 113, 115, 117, 128

LLL *Lenstra-Lenstra-Lovasz*. 32

LPN *Learning Parity with Noise*. 116, 151

LWE *Learning With Errors*. 38–42, 44, 58, 112, 116, 215

MLWE *Module Learning With Errors*. 44, 58, 109, 203

NIZKPoK *Non-Interactive Zero-Knowledge Proof of Knowledge*. 20, 21, 72, 158, 159, 161, 166, 167, 184, 185, 189, 190, 193–196, 201, 203, 215, 220

NTT *Number Theoretic Transformation*. 77

PCIP *Public-Coin Interactive Proof*. 172

PPT *Probabilistic Polynomial-Time*. 7, 10, 13, 26, 46, 54, 58

QROM *Quantum Random Oracle Model*. 158, 167, 173–175

Ring-ISIS *Ring Inhomogeneous Short Integer Solution*. 43

Ring-SIS *Ring Short Integer Solution*. 43, 44

RLWE *Ring Learning With Errors*. 44–47, 58, 60, 64, 107–110, 117, 118, 120, 123, 124, 128, 151, 154, 159, 183, 191, 192, 194, 195, 199, 201, 203

ROM *Random Oracle Model*. 22, 112, 158, 167, 168, 174, 175, 184

RSA *Rivest-Shamir-Adelman*. 4, 6, 23

SD *Statistical Distance*. 61–63, 65, 66, 124

SDP *Syndrome Decoding Problem*. 50, 113, 128

SIS *Short Integer Solution*. 37, 113, 115

SVP *Shortest Vector Problem*. 26, 32, 37, 38, 44

XOF *eXtendable Output Function*. 22, 160, 161, 186, 188

ZKPoK *Zero-Knowledge Proof of Knowledge*. 18, 20, 22, 24, 47, 49, 50, 54, 55, 64, 72, 107–110, 116, 118, 126, 127, 129, 132, 138, 152, 153, 157, 166, 184, 200–203

Bibliography

[1] Carlos Aguilar Melchor, Slim Bettaieb, Philippe Gaborit, and Julien Schrek. A code-based undeniable signature scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 99–119, Oxford, UK, December 17–19, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-45239-0_7.

[2] Carlos Aguilar Melchor, Pierre-Louis Cayrel, Philippe Gaborit, and Fabien Laguillaumie. A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory*, 57(7):4833–4842, 2011. doi:10.1109/TIT.2011.2145950.

[3] Carlos Aguilar Melchor, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop*, pages 648–652, 2011. doi:10.1109/ITW.2011.6089577.

[4] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press. doi:10.1145/237814.237838.

[5] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th ACM STOC*, pages 10–19, Dallas, TX, USA, May 23–26, 1998. ACM Press. doi:10.1145/276698.276705.

[6] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A practical group signature scheme based on rank metric. In Sylvain Duquesne and Svetla Petkova-Nikova, editors, *Arithmetic of Finite Fields*, pages 258–275, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-55227-9_18.

[7] Martin R. Albrecht and Amit Deo. Large modulus ring-LWE $\geq$ module-LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*,

volume 10624 of *LNCS*, pages 267–296, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. `doi:10.1007/978-3-319-70694-8_10`.

[8] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. `doi:10.1515/jmc-2015-0016`.

[9] Richard Arratia and Louis Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131, 1989. `doi:10.1007/BF02458840`.

[10] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-84245-1_19`.

[11] Thomas Attema and Serge Fehr. Parallel repetition of $(k_1, \dots, k_\mu)$-special-sound multi-round interactive proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 415–443, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. `doi:10.1007/978-3-031-15802-5_15`.

[12] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. `https://eprint.iacr.org/2021/1377`.

[13] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142, Chicago, IL, USA, November 7–10, 2022. Springer, Heidelberg, Germany. `doi:10.1007/978-3-031-22318-1_5`.

[14] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 470–499, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-56880-1_17`.

[15] Eric Bach and Jeffrey Outlaw Shallit. *Algorithmic number theory*. Foundations of computing. MIT Press, Cambridge, MA, 1996.

[16] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptogra-

phy: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018. doi:10.1007/s00145-017-9265-9.

[17] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-98113-0_20.

[18] Hanno Becker, Vincent Hwang, Matthias J. Kannwischer, Bo-Yin Yang, and Shang-Yi Yang. Neon NTT: Faster dilithium, kyber, and saber on cortex-A72 and apple M1. *IACR TCHES*, 2022(1):221–244, 2022. doi:10.46586/tches.v2022.i1.221-244.

[19] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 305–325, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-24174-6_16.

[20] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. A general method for solving divide-and-conquer recurrences. *SIGACT News*, 12(3):36–44, September 1980. doi:10.1145/1008861.1008865.

[21] Elwyn R. Berlekamp, Robert J. McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, May 1978. doi:10.1109/TIT.1978.1055873.

[22] Daniel J Bernstein. Multidigit multiplication for mathematicians. https://cr.yp.to/papers/m3.pdf.

[23] Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 34–51, Limoges, France, June 4–7, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-38616-9_3.

[24] Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45727-3_7.

[25] Olivier Blazy, Philippe Gaborit, and Dang Truong Mac. A correction to a code-based blind signature scheme. In Antonia Wachter-Zeh, Hannes Bartz, and Gianluigi Liva, editors, *Code-Based Cryptography*, pages 84–94, Cham, 2022. Springer International Publishing. `doi:10.1007/978-3-030-98365-9_5`.

[26] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, Heidelberg, Germany, 1998. Invited paper. `doi:10.1007/BFb0054851`.

[27] Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on RSA given a small fraction of the private key bits. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 25–34, Beijing, China, October 18–22, 1998. Springer, Heidelberg, Germany. `doi:10.1007/3-540-49649-1_3`.

[28] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-56880-1_16`.

[29] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 176–202, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-26948-7_7`.

[30] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum SNARKs for RSIS and RLWE and their applications to privacy. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 247–267, Paris, France, April 15–17, 2020. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-44223-1_14`.

[31] Pedro Branco and Paulo Mateus. A code-based linkable ring signature scheme. In Joonsang Baek, Willy Susilo, and Jongkil Kim, editors, *ProvSec 2018*, volume 11192 of *LNCS*, pages 203–219, Jeju, South Korea, October 25–28, 2018. Springer, Heidelberg, Germany. `doi:10.1007/978-3-030-01446-9_12`.

[32] Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. Discrete ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 402–417,

Burnaby, BC, Canada, August 14–16, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-43414-7_20.

[33] Pierre-Louis Cayrel and Sidi Mohamed El Yousfi Alaoui. Dual construction of stern-based signature scheme. *Zenodo*, March 2010. doi:10.5281/zenodo.1072826.

[34] Pierre-Louis Cayrel, Sidi Mohamed El Yousfi Alaoui, Gerhrad Hoffmann, and Pascal Véron. An improved threshold ring signature scheme based on error correcting codes. In Ferruh Özbudak and Francisco Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields*, pages 45–63, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-31662-3_4.

[35] Pierre-Louis Cayrel, Philippe Gaborit, and Emmanuel Prouff. Secure implementation of the stern authentication and signature schemes for low-resource devices. In Gilles Grimaud and François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications*, pages 191–205, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-85893-5_14.

[36] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. Improved zero-knowledge identification with lattices. In Swee-Huay Heng and Kaoru Kurosawa, editors, *ProvSec 2010*, volume 6402 of *LNCS*, pages 1–17, Malacca, Malaysia, October 13–15, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-16280-0_1.

[37] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 255–272, Puebla, Mexico, August 8–11, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-14712-8_16.

[38] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 171–186, Waterloo, Ontario, Canada, August 12–13, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-19574-7_12.

[39] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*,

volume 10032 of *LNCS*, pages 135–165, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6_5.

[40] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. NTT multiplication for NTT-unfriendly rings. *IACR TCHES*, 2021(2):159–188, 2021. doi:10.46586/tches.v2021.i2.159-188.

[41] Simon Cogliani, Houda Ferradi, Rémi Géraud, and David Naccache. Thrifty zero-knowledge. In Feng Bao, Liqun Chen, Robert H. Deng, and Guojun Wang, editors, *Information Security Practice and Experience*, pages 344–353, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-49151-6_24.

[42] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965. doi:10.2307/2003354.

[43] Don Coppersmith. Finding a small root of a univariate modular equation. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 155–165, Zaragoza, Spain, May 12–16, 1996. Springer, Heidelberg, Germany. doi:10.1007/3-540-68339-9_14.

[44] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[45] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round Fiat-Shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56877-1_21.

[46] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26951-7_13.

[47] Ali El Kaafarani and Shuichi Katsumata. Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 89–119, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-76581-5_4.

[48] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, Rachid El Bansarkhani, and Gerhard Hoffmann. Code-based identification and signature schemes in software. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics*, pages 122–136, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40588-4_9.

[49] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, and Meziani Mohammed. Improved identity-based identification and signature schemes using quasi-dyadic goppa codes. In Tai-hoon Kim, Hojjat Adeli, Rosslin John Robles, and Maricel Balitanas, editors, *Information Security and Assurance*, pages 146–155, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23141-4_14.

[50] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 259–288, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64834-3_9.

[51] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_5.

[52] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. doi:10.1007/3-540-47721-7_12.

[53] Charles M. Fiduccia. Polynomial evaluation via the division algorithm. the fast fourier transform revisited. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, STOC '72, pages 88–93, New York, NY, USA, 1972. Association for Computing Machinery. doi:10.1145/800152.804900.

[54] Ulrich Fincke and Michael Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985. doi:10.2307/2007966.

[55] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *2007 IEEE International Symposium on Information Theory*, pages 191–195, June 2007. doi:10.1109/ISIT.2007.4557225.

[56] Cyril J. Gadd and R. Campbell Thompson. A middle-babylonian chemical text. *IRAQ*, 3(1):87–96, 1936. doi:10.2307/4241587.

[57] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 207–216, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374408.

[58] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_13.

[59] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 623–651, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45374-9_21.

[60] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_5.

[61] Wang Hongbin and Ren Yan. Code-based designated verifier signature scheme. In *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, pages 550–554, 2013. doi:10.1109/EIDWT.2013.99.

[62] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. On zero-knowledge identification based on q-ary syndrome decoding. In *2013 Eighth Asia Joint Conference on Information Security*, pages 12–18, 2013. doi:10.1109/ASIAJCIS.2013.10.

[63] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. Proof of plaintext knowledge for code-based public-key encryption revisited. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13*, pages 535–540, Hangzhou, China, May 8–10, 2013. ACM Press. doi:10.1145/2484313.2484385.

[64] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 663–680, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-34961-4_40.

[65] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22, Vienna, Austria, December 14–16, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-65411-5_1.

[66] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206, Boston, MA, USA, April 25–27, 1983. ACM Press. doi:10.1145/800061.808749.

[67] Anatolii Karatsuba. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596, 1963.

[68] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 372–389, Melbourne, Australia, December 7–11, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-89255-7_23.

[69] Donald E. Knuth and Andrew Chi-Chih Yao. *Algorithms and Complexity: New Directions and Recent Results*, chapter "The complexity of nonuniform random number generation". Academic Press, Orlando, USA, 1976.

[70] Veronika Kuchta, Amin Sakzad, Ron Steinfeld, and Joseph K. Liu. Lattice-based zero-knowledge arguments for additive and multiplicative relations. *Designs, Codes, and Cryptography*, 89(5):925–963, 2021. doi:10.1007/s10623-021-00851-1.

[71] Adarsh Kumar, Carlo Ottaviani, Sukhpal Singh Gill, and Rajkumar Buyya. Securing the future internet of things with post-quantum cryptography. *SECURITY AND PRIVACY*, 5(2):e200, 2022. doi:10.1002/spy2.200.

[72] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_14.

[73] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. `doi:10.1007/BF01457454`.

[74] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 700–732, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. `doi:10.1007/978-3-319-96881-0_24`.

[75] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. *Theoretical Computer Science*, 759:72 – 97, 2019. `doi:10.1016/j.tcs.2019.01.003`.

[76] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 107–124, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-36362-7_8`.

[77] Vadim Lyubashevsky. Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, 2011. `https://eprint.iacr.org/2011/537`.

[78] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-29011-4_43`.

[79] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 54–72, Lausanne, Switzerland, February 10–13, 2008. Springer, Heidelberg, Germany. `doi:10.1007/978-3-540-71039-4_4`.

[80] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. `doi:10.1007/978-3-031-15979-4_3`.

[81] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In Jay Ligatti,

Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1051–1070, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3417894.

[82] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 215–241, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_9.

[83] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_1.

[84] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, November 2013. doi:10.1145/2535925.

[85] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-38348-9_3.

[86] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 204–224, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78381-9_8.

[87] Ramiro Martínez and Paz Morillo. RLWE-based zero-knowledge proofs for linear and multiplicative relations. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 252–277, Oxford, UK, December 16–18, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-35199-1_13.

[88] Ramiro Martínez and Paz Morillo. Revisiting fast fourier multiplication algorithms on quotient rings, 2023. doi:10.48550/arXiv.2304.08860.

[89] Ramiro Martínez, Paz Morillo, and Sergi Rovira. Implementation and performance of a RLWE-based commitment scheme and ZKPoK for its linear and

multiplicative relations. Cryptology ePrint Archive, Report 2023/1026, 2023. https://eprint.iacr.org/2023/1026.

[90] Ramiro Martínez and Sergi Rovira. RLWE-commitment. GitHub repository, 2023. https://github.com/rammmiro/RLWE-Commitment.

[91] Kirill Morozov and Tsuyoshi Takagi. Zero-knowledge protocols for the McEliece encryption. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12*, volume 7372 of *LNCS*, pages 180–193, Wollongong, NSW, Australia, July 9–11, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-31448-3_14.

[92] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008. doi:10.1515/JMC.2008.009.

[93] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. *An Introduction to the theory of numbers*. John Wiley, 5th edition, 1991.

[94] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536461.

[95] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_5.

[96] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016. doi:10.1561/0400000074.

[97] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 145–166, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. doi:10.1007/11681878_8.

[98] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398, Zaragoza, Spain, May 12–16, 1996. Springer, Heidelberg, Germany. doi:10.1007/3-540-68339-9_33.

[99] Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In Alejandro Hevia and Gregory Neven, editors, *LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 139–158, Santiago, Chile, October 7–10, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-33481-8_8.

[100] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980. doi:10.1016/0022-314X(80)90084-0.

[101] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi:10.1145/1060590.1060603.

[102] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.

[103] Eduard Sanou Gozalo. Post-quantum cryptography: lattice-based encryption. Master's thesis, Universitat Politècnica de Catalunya, 2016. http://hdl.handle.net/2117/93068.

[104] Jason Scharfman. *Decentralized Finance (DeFi) Fraud and Hacks: Part 2*, pages 97–110. Springer International Publishing, Cham, 2023. doi:10.1007/978-3-031-23679-2_7.

[105] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_22.

[106] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994. doi:10.1007/BF01581144.

[107] Daniel Shanks. Five number-theoretic algorithms. In R. S. D. Thomas and Hugh C. Williams, editors, *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Winnipeg)*, pages 51–70, 1973.

[108] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.

[109] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press. `doi:10.1109/SFCS.1994.365700`.

[110] Rosemberg Silva, Pierre-Louis Cayrel, and Richard Lindner. A lattice-based batch identification scheme. In *2011 IEEE Information Theory Workshop*, pages 215–219, October 2011. `doi:10.1109/ITW.2011.6089381`.

[111] Rosemberg Silva, Antonio C. de A. Campello, and Ricardo Dahab. Lwe-based identification schemes. In *2011 IEEE Information Theory Workshop*, pages 292–296, 2011. `doi:10.1109/ITW.2011.6089439`.

[112] Unathi Skosana and Mark Tame. Demonstration of Shor's factoring algorithm for $n = 21$ on IBM quantum processors. *Scientific Reports*, 11(1):16599, August 2021. `doi:10.1038/s41598-021-95973-w`.

[113] Bo Song and Yiming Zhao. Provably secure identity-based identification and signature schemes with parallel-PVR. In Kwok-Yan Lam, Chi-Hung Chi, and Sihan Qing, editors, *ICICS 16*, volume 9977 of *LNCS*, pages 227–238, Singapore, November 29 – December 2, 2016. Springer, Heidelberg, Germany. `doi:10.1007/978-3-319-50011-9_18`.

[114] Yongcheng Song, Jiang Zhang, Xinyi Huang, Wei Wu, and Haining Yang. Statistical zero-knowledge and analysis of rank-metric zero-knowledge proofs of knowledge. *Theoretical Computer Science*, 952:113731, 2023. `doi:10.1016/j.tcs.2023.113731`.

[115] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-10366-7_36`.

[116] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany. `doi:10.1007/3-540-48329-2_2`.

[117] Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, Nov 1996. `doi:10.1109/18.556672`.

[118] Gilbert Strang. *Introduction to linear algebra*. Cambridge Press, Wellesley, 5th edition, 2016.

[119] Yang Tao, Xi Wang, and Rui Zhang. Short zero-knowledge proof of knowledge for lattice-based commitment. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 268–283, Paris, France, April 15–17, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-44223-1_15.

[120] Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1):57–69, Jan 1997. doi:10.1007/s002000050053.

[121] Joachim Von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*, volume 53. Cambridge University Press, 3rd edition, 2013. doi:10.1017/CB09781139856065.

[122] Xiaoyun Wang, Mingjie Liu, Chengliang Tian, and Jingguo Bi. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem (keynote talk). In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11*, pages 1–9, Hong Kong, China, March 22–24, 2011. ACM Press. doi:10.1145/1966913.1966915.

[123] Zhiming Wang. Miller-Rabin. GitHub repository, 2019. https://github.com/zmwangx/miller-rabin.

[124] Xiang Xie, Rui Xue, and Minqian Wang. Zero knowledge proofs from ring-LWE. In Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, editors, *CANS 13*, volume 8257 of *LNCS*, pages 57–73, Paraty, Brazil, November 20–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-02937-5_4.

[125] Guomin Yang, Chik How Tan, Yi Mu, Willy Susilo, and Duncan S. Wong. Identity based identification from algebraic coding theory. *Theoretical Computer Science*, 520:51–61, 2014. doi:10.1016/j.tcs.2013.09.008.

[126] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 147–175, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_6.