MASTER'S THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: RLWE-based distributed key generation and threshold decryption

Author: Ferran Alborch Escobar

Advisors: Ramiro Martínez Pinilla and Paz Morillo Bosch

Department: Applied Mathematics

Academic year: 2020-2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master in Advanced Mathematics and Mathematical Engineering

Master's thesis

RLWE-based distributed key generation and threshold decryption

Ferran Alborch Escobar

Supervised by Ramiro Martínez Pinilla and Paz Morillo Bosch

25th June 2021

Thanks to my supervisors Paz Morillo and Ramiro Martínez for introducing me to the fascinating world of latticed-based cryptography, for their invaluable help and advice and for their help in starting in the academic world.

Thanks to Diana, Diana and Ana for reading through the thesis and helping improving it with their comments and advice.

Thanks to my family for their unconditional support.

Abstract

Ever since the appearance of quantum computers, prime factoring and discrete logarithm based cryptography has been put in question, giving birth to the so called post-quantum cryptography. The most prominent field in post-quantum cryptography is lattice-based cryptography, protocols that are proved to be as difficult to break as certain difficult lattice problems like Learning With Errors (LWE) or Ring Learning With Errors (R-LWE). This Master's Degree Thesis forwards and elevates the work done in the Bachelor's Degree Thesis [AE20] by taking the R-LWE-based protocols specified there and giving more accurate and improved proofs of correctness and security, developing an original proposal for dispute resolution throughout both protocols, making an analysis of the specific hardness of breaking security of the protocol and analyzing a given implementation of both protocols in C.

Keywords

Post-quantum cryptography, Threshold cryptography, Lattices, R-LWE encryption

Contents

1. Introduction

The vast improvement of computers during the information revolution ushered humanity into the information age in the late XXth century. As this age progressed with the coming of the new century it was made clear that through the use of the internet, everything and anything was at an arm's reach. This meant that one had information going through these world wide channels, some of which should not be of easy access to anyone but authorized entities. Here is where we find cryptography.

The need to hide (encrypt) information so only a few target people are able to recover (decrypt) it, has been a well thought out problem throughout history, especially in the military operations arena. Some of the greatest advances in the history of cryptography are strongly tied with wars, one of the main examples being World War II. With the appearance of the computer, it was rapidly seen that cryptography could be of great help, and advancements in both areas have come hand in hand since then. Whenever a more powerful computing technique is found, new cryptographic protocols may be needed, and whenever a new cryptographic protocol is developed more powerful computing techniques may be needed to break it. This circular behaviour meant that the information age has also been a golden era for cryptography.

One of these new computing advancements forwarding cryptography was the idea of quantum computing in [Ben80]. This new model allowed for different ways to execute algorithms, and thus gave a new tool to try and solve problems thought to be hard at that moment. It was not very long until an algorithm was developed by Shor in [Sho99] that allowed for fast resolutions of both the prime factorization and discrete logarithm problems. This meant that the encryption schemes most used at the moment (and up to current days), based on the RSA problem proposed in [RSA78], the hardness of which depended on hardness of prime factoring, or on ElGamal proposed in [ElG85] or elliptic curves problems, the hardness of which depended on the discrete logarithm, are insecure against quantum adversaries. Therefore new quantum resistant protocols were needed.

There are two main ways to face this problem: quantum cryptography and post-quantum cryptography. Quantum cryptography relies on quantum algorithms that cannot be broken by quantum adversaries, while post-quantum deals with classical (non-quantum) algorithms that cannot be broken by quantum adversaries. Although both are interesting in their own right, given that widespread usage of moderately powerful quantum computers seems unachievable in the short run, we focus in post-quantum cryptography. In this realm the area that has had more recent advancements is lattice-based cryptography, especially cryptography based in the Learning With Errors (LWE) problem and its variants, like Ring Learning With Errors (R-LWE), the variant which our proposals are built around. This is backed by the fact that in the Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process [AASA+20] most third-round finalists are lattice-based schemes.

There are many applications of post-quantum cryptography, but the one we are involved with and hoping our contributions can be applied to, is electronic voting. However in electronic voting we have an added difficulty, and that is the lack of trust. Given that the lack of trust in other entities is what initially spawned the concept of cryptography, going further in this direction is the next logical step to follow. Therefore, what we want is to "spread" that trust, so that one single corrupt player can no longer mess up with the protocol. Distributed cryptography is this idea of spreading the tasks between several players so that only certain subsets of them can perform the crpytographic protocol. And this finally brings us to the main subject of our thesis: R-LWE-based distributed key generation and threshold decryption.

1.1 State of the Art and Contributions

Despite the usefulness of and interest in efficient post-quantum threshold public key encryption cryptography, there are not many proposals, and even less that focus on the R-LWE problem. Most current proposals revolve around the LWE problem (for example [BD10], [BGG+18], [PE17] and [SRB16]) which has the potential problem of keys and ciphertexts growing with $O(n^2)$ instead of with $O(n)$ like the R-LWE variant (with n the dimension of the lattice), thus having a high possibility to need a greater amount of operations and therefore computation time. In the world of threshold encryption based on R-LWE as far as we know there is only one proposal given in [ZXJ+14], which is based on the homomorphic properties of their Fully Homomorphic Encryption scheme. However, this proposal does not come with a distributed key generation protocol (they rely on a Trusted Third Party (TTP) for that) and as with all the other proposals, to the best of our knowledge there are no given implementations to truly analyze computation times.

Our objectives for this work (which we expect to turn into contributions) are on the vein of improving the current state of the art. To do so we want to improve and expand on the original protocols we gave on [AE20], that as far as we are concerned are the first R-LWE based threshold protocols including both decryption and key generation. The protocols are based on the LWE proposal given by Bendlin and Damgård in [BD10], their ideas transported into the R-LWE setting. In this Master's Degree Thesis we have four objectives: improve and in some cases completely remake the proofs given in [AE20] to make them more accurate, more legible and to conform better with the standard proofs in cryptography; produce an original dispute resolution protocol to work against an active adversary; give an accurate analysis, not only asymptotic, for when our instance of R-LWE is hard to solve to decide ranges for the parameters; and to give an implementation of both protocols in C to analyze the computation time and storage needed for the protocols and which are the limiting parameters for the performance of the protocols.

1.2 Organization of the Thesis

This Master's Degree Thesis is structured into the following chapters:

- **Preliminaries:** where notation and the needed background knowledge about cryptography and lattices is given.
- **Discrete Gaussian Measures and Correctness:** where we discuss two different discrete Gaussian distributions, and we use some results on one of them to prove the correctness of our encryption scheme.
- **Definitions and Proofs of Security:** where we explain several concepts about proofs of security in cryptography and use these definitions and properties to prove the security of our encryption scheme and the non-leakage of information in the distribution.
- **Implementation:** where we give our original dispute resolution protocol and discuss the protocols implementation we give, how we have done it and the results we have extracted from it.
- **Conclusions:** where we sum up the achievements of this thesis and lay out future work that may stem from here.

2. Preliminaries

2.1 Notation

Elements in $\mathbb{R}$, $\mathbb{Z}$ or $\mathbb{Z}_q$ will be indicated as lower case letters $(a, b, ...)$, while elements in $\mathbb{R}^n$, $\mathbb{Z}^n$ or $\mathbb{Z}_q^n$ will be indicated as bold lower case letters $(\mathbf{a}, \mathbf{b}, ...)$. $X \longleftarrow \chi$ means X is sampled from a random variable following the distribution χ, $Y \longleftarrow \chi^n$ means Y is a vector such that every coordinate independently follows the distribution χ. We denote the inner product of $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ as $\langle \mathbf{a}, \mathbf{b} \rangle := \sum_{i=1}^{n} a_i b_i$ for some $n, q \in \mathbb{Z}_{>0}$. We will also identify any polynomial of degree $n-1$, $\mathbf{f}(x) = a_0 + a_1 x + ... + a_{n-1} x^{n-1} \in \mathbb{Z}_q[x]$ with the vector $\mathbf{f} = (a_0, a_1, ..., a_{n-1}) \in \mathbb{Z}_q^n$. A function g is said to be negligible over n ($g := \text{neg}(n)$) if $\forall k \in \mathbb{Z}_{>0}$, $\exists n_0 \in \mathbb{Z}_{>0}$ such that $\forall n \geq n_0$, $|g(n)| < \frac{1}{n^k}$. Finally, for any set $\mathcal{J}$, we denote as $j \xleftarrow{\$} \mathcal{J}$ the action of choosing j uniformly at random from $\mathcal{J}$.

2.2 Cryptographic Primitives

We will start by giving some cryptographic primitives, well-known definitions, protocols or techniques used in cryptography upon which we will build our encryption scheme and protocols. First we will properly define what an encryption scheme is.

Definition 2.1 ([MVOV18])**.** An *encryption scheme* is a tuple $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ such that:

- $\mathcal{M}$ is a set called *plaintext space*.
- $\mathcal{C}$ is a set called *ciphertext space*.
- $\mathcal{K}$ is a set called *key space*. Generally a key generation is also specified to generate $k \in \mathcal{K}$.
- $\mathcal{E} = \{\mathsf{E}_k : k \in \mathcal{K}\}$ is a set of functions $\mathsf{E}_k : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ called *encryption functions*. $\mathcal{R}$ is a *randomness space* because some encryption (and decryption) protocols use random values.
- $\mathcal{D} = \{\mathsf{D}_k : k \in \mathcal{K}\}$ is a set of functions $\mathsf{D}_k : \mathcal{C} \times \mathcal{R} \to \mathcal{M}$ called *decryption functions*.

Note that if we have $\mathcal{D}$ and $\mathcal{E}$ use the same key, then we call it *symmetric encryption*, otherwise we call it *asymmetric* or *public key encryption*. Also, in public key encryption $\mathcal{K}$ can be divided in two different sets, $\mathcal{K}_s$ the *secret key space* and $\mathcal{K}_p$ the *public key space*. The public key (known by all entities) is used to encrypt messages and the secret key (known only to the entity decrypting) is used to decrypt.

Once we have defined an encryption scheme we need to prove some properties about it, otherwise it would not be useful, the most important of which are correctness, to ensure that the protocols give a correct output, and security, an adversary cannot recover the message without the key. In section 2.5 we will give the encryption scheme in which we are basing our protocol and given that we want to delve deep into the definitions of correctness and security and ways to prove them, they are dedicated an entire chapter each, chapter 3 for correctness and chapter 4 for security.

Having defined encryption schemes, we reinstate again, as in the introduction, that what we are really focusing in is not the classical cryptography scheme with one sender and one receiver, but rather what is called threshold or distributed cryptography. The situation we are interested in is the requirement of the work of several servers to decrypt a single message. The concept was born originally as a way to share information (or secrets) in multiparty computation, and was then extended to encryption schemes.

Definition 2.2 ([Sha79])**.** A *threshold secret sharing scheme* of threshold t and u players is a scheme such that given some data D it divides it into u pieces $D_1 \dots, D_u$ such that:

- Knowledge of $t+1$ or more pieces D_i makes D easily computable.
- Knowledge of t or less pieces D_i leaves D completely undetermined (i.e. all its possible values are equally likely).

Remark 2.3. Note that threshold secret sharing is merely one of the ways a secret can be distributed between u players, and allows only for a fairly rigid way to recover it. However, one can be even more general and define what is called an *access structure* Γ, which denotes all subsets of $\mathcal{P} = (P_1, \dots, P_n)$ allowed to retrieve the information. In threshold secret sharing we have:

$$\Gamma_t := \{A \subseteq \mathcal{P} \mid |A| > t\} \text{ with } t < u.$$

We will focus on threshold access structures since they are the ones we need.

Definition 2.4. We will call a *threshold encryption scheme* a secret sharing scheme where what we try to recover is a plaintext from a ciphertext.

Therefore, in a threshold encryption scheme the encryption works as in general encryption schemes (only one person or entity encrypts the message), but it is necessary the collaboration of $t+1$ players to decrypt the message.

One of the first secret sharing schemes and one of the most used still due to its simplicity to compute and understand, is Shamir Secret Sharing. We will use it profusely throughout our work.

Technique 2.5 ([Sha79])**.** *Shamir Secret Sharing over a field* $\mathbb{F}$ of a secret $s \in \mathbb{F}$ of threshold t works as follows:

- Choose t elements $b_i \in \mathbb{F}$ and define the polynomial $f(z) := s + \sum_{i=1}^{t} b_i z^i$ (i.e. choose a random polynomial $f(z) \in \mathbb{F}[x]$ such that $f(0) = s$).
- For every player P_j, their share of the secret is $f(i_j)$.
- When $t+1$ players want to recover the secret they use Lagrange interpolation to find $f(z)$ and then compute $f(0)$.

Note that the recovery of the secret works because of Lagrange interpolation, which states that given $t+1$ points $(f(i_j),\ j = 1, \dots, t+1)$, there exists a unique polynomial of degree at most t that passes through these points, and it is given by:

$$f(z) = \sum_{j=1}^{t+1} f(i_j) \prod_{k \neq j} \frac{z - i_k}{i_j - i_k}.$$

Furthermore, Shamir Secret Sharing is especially useful since it is linear, therefore it satisfies the two following useful properties (we will prove them modulo q to ease notation but they hold for any field).

Lemma 2.6. *The linear combination (with elements of $\mathbb{Z}_q$) of Shamir shares of different secrets is a Shamir share of the same linear combination of the secrets.*

Proof. Let $(a^1, \dots, a^u) \in \mathbb{Z}_q^u$ Shamir shares of $a \in \mathbb{Z}_q$ with threshold $t < u$, and $(b^1, \dots, b^u) \in \mathbb{Z}_q^u$ Shamir shares of $b \in \mathbb{Z}_q$ with threshold t. This means that for any $t+1$ shares of a, if $f(z) := \sum_{j=1}^{t+1} f(i_j) \prod_{k \neq j} \frac{z-i_k}{i_j-i_k} \pmod q$ with $f(i_j) = a^j$ then $f(0) = a$ and for any $t+1$ shares of b, if $g(z) := \sum_{j=1}^{t+1} g(i_j) \prod_{k \neq j} \frac{z-i_k}{i_j-i_k} \pmod q$ with $g(i_j) = b^j$ then $g(0) = b$.

We want to see that $\lambda(a^1, \dots, a^u) + \mu(b^1, \dots, b^u)$, $\lambda, \mu \in \mathbb{Z}_q$ are Shamir shares of $\lambda a + \mu b \pmod q$, i.e that given any $t+1$ shares, if $h(z) := \sum_{j=1}^{t+1} h(i_j) \prod_{k \neq j} \frac{z-i_k}{i_j-i_k} \pmod q$ with $h(i_j) = \lambda a^j + \mu b^j \pmod q$ then $h(0) = \lambda a + \mu b \pmod q$. Let us see it:

$$h(z) \equiv \sum_{j=1}^{t+1} (\lambda a^j + \mu b^j) \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} \equiv \lambda \left(\sum_{j=1}^{t+1} a^j \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} \right) + \mu \left(\sum_{j=1}^{t+1} b^j \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} \right) \equiv$$

$$\equiv \lambda f(z) + \mu g(z) \pmod q.$$

And therefore $h(0) \equiv \lambda f(0) + \mu g(0) \equiv \lambda a + \mu b \pmod q$. □

Lemma 2.7. *Let $(a^1, \dots, a^u) \in \mathbb{Z}_q^u$ be Shamir share of $a \in \mathbb{Z}_q$ with threshold $t < u$ and $b \in \mathbb{Z}_q$, then $(b + a^1, \dots, b + a^u) \pmod q$ is a Shamir share of $b + a \pmod q$ with threshold t.*

Proof. We know that for any $t+1$ shares of a if $f(z) := \sum_{j=1}^{t+1} f(i_j) \prod_{k \neq j} \frac{z-i_k}{i_j-i_k} \pmod q$ with $f(i_j) = a^j$ then $f(0) = a$. Let us see that for any $t+1$ shares in $(b + a^1, \dots, b + a^n)$, if $g(z) := \sum_{j=1}^{t+1} g(i_j) \prod_{k \neq j} \frac{z-i_k}{i_j-i_k} \pmod q$ with $g(i_j) = b + a^j \pmod q$ then $g(0) = b + a \pmod q$:

$$g(z) :\equiv \sum_{j=1}^{t+1} (b + a^j) \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} \equiv \sum_{j=1}^{t+1} b \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} + \sum_{j=1}^{t+1} a^j \prod_{k \neq j} \frac{z - i_k}{i_j - i_k} :\equiv h(z) + f(z) \pmod q.$$

Where $h(z)$ is a polynomial of degree at most t that has $t+1$ points fixed at b, which implies that $h(z) = b$. Therefore we have $g(0) \equiv h(0) + f(0) \equiv b + a \pmod q$. □

Other cryptographic tools we will use will be both the Pseudo-Random Secret Sharing (PRSS) and the Non-Interactive Verifiable Secret Sharing (NIVSS) techniques. These tools will be primordial in our proposal, since the security of our protocols is based on being able to mask the relevant information with noise in such a way that the adversary cannot retrieve it. To generate this noise we will use these two protocols.

Definition 2.8. A *Pseudo-Random Function (PRF)*, $\Phi_\cdot(\cdot)$, is a deterministic function that maps two sets (domain and range) on the basis of a key, which when run multiple times with the same input gives the same output but given an arbitrary input the output seems random, i.e. one cannot distinguish the output of a given input from a random oracle.

Technique 2.9 ([CDI05])**.** *Pseudo-Random Secret Sharing in $\mathbb{Z}_q$ (PRSS)* allows u players to non-interactively share a common random value x with a threshold of t players ($t < u$) given a pseudo-random function $\Phi_\cdot(\cdot)$ that with input a seed and a value λ outputs values in the interval $I = [a, b]$, $a < 0, b > 0$ and whatever group of players of size less or equal than t cannot obtain relevant information on x. The algorithm works as follows:

- For each subset H of t players a TTP defines a key $K_H \in \mathbb{Z}_q$ uniformly at random.
- Each player P_j is given K_H, $\forall H$ such that $P_j \notin H$.
- The pseudo-random number they are sharing is $x := \sum_H \Phi_{K_H}(\lambda)$, for a value λ. Since there are $\binom{u}{t}$ such subsets H, we know $x \in [\binom{u}{t}a, \binom{u}{t}b]$.

Technique 2.10 ([CDI05]). *Non-Interactive Verifiable Secret Sharing in $\mathbb{Z}_q$ (NIVSS)*, allows a dealer D to share a secret s with u players with threshold t given a value λ and a pseudo-random function $\phi.(\cdot)$ that with input a seed and λ outputs values in the interval $I = [a, b]$, $a < 0, b > 0$. It works very similarly to PRSS. The algorithm works as follows:

1. For each subset H of t players the dealer D chooses a key $K_H \in \mathbb{Z}_q$ uniformly at random.
2. The dealer D gives to player P_j all the K_H such that $P_j \notin H$.
3. The dealer D reconstructs the pseudo-random value the players share $r = \sum_H \phi_{K_H}(\lambda)$, since he has all the keys.
4. D broadcasts the value $s - r$, and now all the players have a share of s by adding their shares on r.

Finally we will need a way to convert additive shares (like the ones in PRSS) into Shamir shares. To do so we will use the following technique.

Technique 2.11 ([Cat05]). *Converting from additive shares to polynomial shares* allows to convert additive shares of a secret a of threshold $u - 1$ (i.e. $a = \sum_{i=1}^{u} a^j$) $(a^1, \dots, a^u)$ to $(a'^1, \dots, a'^u)$ Shamir shares of a for u players with threshold t. The algorithm works as follows:

- Each player P_j chooses t elements $\beta_i \in \mathbb{Z}_q$ and computes $f_j(z) = a^j + \sum_{i=1}^{t} \beta_i z^i$.
- Each player P_j sends to every player i $f_j(i)$.
- The Shamir share of a for each player P_j is $a'^j = f(j) := \sum_{k=1}^{u} f_k(j)$.

Finally, since we will be using distributed methods, one must ensure that the order in which the different players send information does not compromise the security of the scheme, since, broadly speaking, the last player to send information would have an advantage respect the first one due to knowing more information when making its decision.

To solve this problem it is standard to use commitment schemes.

Definition 2.12 (Definition 8.8 [BS20]). Given a message space $\mathcal{M}$, a *commitment scheme* is a pair of efficient algorithms $\mathcal{C} = (C, V)$ where C is an algorithm that given $m \in \mathcal{M}$ outputs a commitment c and an opening string o and V is a deterministic protocol that given (m, c, o) outputs *accept* or *reject*; and such that it satisfies the following properties:

- **Correctness:** For all $m \in \mathcal{M}$, if $C(m) = (c, o)$ then $\Pr(V(m, c, o) = \text{'accept'}) = 1$.
- **Binding:** This property is the notion that once a commitment c is generated, it should only commit for one message in $\mathcal{M}$. In particular, for every efficient adversary $\mathcal{A}$ that outputs (c, m_1, o_1, m_2, o_2) we must have that

$$\Pr(m_1 \neq m_2 \text{ and } V(m_1, c, o_1) = V(m_2, c, o_2) = \text{'accept'}) = \text{neg}(\lambda)$$

where λ is the security parameter.

- **Hiding:** This property is the notion that the commitment c alone should not reveal any information about the message m. To properly define this we use a semantic security attack game (see Attack Game 4.3) where instead of encrypting the messages we compute its commitment. What we ask is, if W_b denotes the event that the adversary outputs 1 in experiment b, then

$$|\Pr(W_0) - \Pr(W_1)| = \text{neg}(\lambda).$$

In other words, the commitment binds a unique value in the commit phase (algorithm C) that must be hidden until the information to reveal it is given in the reveal phase (algorithm V). This type of scheme is used to ensure that in every round of a distributed protocol all players choose their actions with the same information regardless of the order in which the round is played.

2.3 Lattices

From all the different types of post-quantum cryptography, we focus on lattice-based cryptography. Thus, to ease the understanding of the hard lattice-based problem our encryption scheme is based we will give some basic definitions and notions about lattices.

Definition 2.13. A *lattice* $\mathcal{L}$ is a set of points in an n-dimensional space (usually $\mathbb{R}^n$), $n \in \mathbb{Z}_{>0}$, such that:

- $\mathcal{L}$ is an additive subgroup: $0 \in \mathcal{L}$ and $\mathbf{x}, \mathbf{y} \in \mathcal{L} \Rightarrow -\mathbf{x}, \mathbf{x} + \mathbf{y} \in \mathcal{L}$.
- $\mathcal{L}$ is discrete: $\forall \mathbf{x} \in \mathcal{L},\ \exists U \ni \mathbf{x}$ such that $U \cap \mathcal{L} = \{\mathbf{x}\}$, with U neighbourhood of $\mathbf{x}$.

This definition therefore gives lattices a periodic structure constrained by addition and opposites, so any lattice can be defined by k vectors in the n-dimensional space, thus giving us the base of the lattice.

Definition 2.14. Given $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{R}^n$ k linearly independent vectors, the *generated lattice* given by this set of vectors is:

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

Note that we have called $\mathbf{B}$ the matrix formed by $\mathbf{b}_1, \dots, \mathbf{b}_k$ as columns, we call $\mathbf{b}_1, \dots, \mathbf{b}_k$ a *basis* of the lattice $\mathcal{L}$. Also note that many different basis may define the same lattice.

Since several different matrices can define one same lattice we would wish to have a measure of how "good" a base is as its usefulness to solve hard lattice problems, and this measure turns out to be the orthogonality of the basis. We have that the hardness of most lattice-based problems depends on how orthogonal the base of the lattice is, so it is the next logical step to find a more orthogonal basis given a lattice $\mathcal{L}$. There are several algorithms: the Lenstra-Lenstra-Lovász (LLL) algorithm presented in [LLL82], that performs gaussian elimination on the elements of the basis two by two; the Blockwise Korkine-Zolotarev (BKZ) reduction presented in [SE94], that improves the former using blocks of vectors but has to find the shortest vector, which is a difficult problem; and there are more algorithms based on the shortest vector problem. However, all these algorithms are exponentially slow, or in the case of LLL is polynomial but its result is exponentially far from being optimal.

From all the possible lattices over $\mathbb{R}^n$ we are interested mainly on those that live on $\mathbb{Z}^n$ since the elements, being vectors of integers, are less costly to store than any real value and furthermore we do not need to deal with truncation errors.

Definition 2.15. A lattice $\mathcal{L}$ is said to be *q-ary* if $\mathbb{Z}_q^n \subset \mathcal{L} \subset \mathbb{Z}^n$. There are two usual ways to represent a q-ary lattice given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$:

- The Λ_q form:

$$\Lambda_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^n | \mathbf{y} \equiv \mathbf{A}\mathbf{z} \pmod{q},\ \mathbf{z} \in \mathbb{Z}^m\}.$$

- The orthogonal $\Lambda_q^\perp$ form:

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^n | \mathbf{A}^T\mathbf{y} \equiv 0 \pmod{q}\}.$$

With this we can define one of the core lattice problems the hardness of which many other problems depend, and that is the Shortest Vector Problem.

Definition 2.16. The *minimum* $\lambda_i(\mathbf{B})$ is the radius of the smallest hypersphere centered in the origin that contains at least i linearly independent points of the lattice.

Definition 2.17. Given $\mathbf{B}$ a base of the lattice $\mathcal{L}(\mathbf{B})$, the *$\gamma-$approximated Shortest Vector problem* (γ-*SVP*) is finding a non-zero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$.

A decision version of the SVP is the *GAPSVP* problem, that is, given $\mathbf{B}$ a base of lattice $\mathcal{L}(\mathbf{B})$, to decide whether its shortest vector is shorter than 1 or bigger than a given value β.

The hardness of these problems depends on γ and β. The γ-SVP has been proven to be NP-hard for γ polynomial on some parameter of the lattice [Ajt98], with only exponential algorithms known to solve it [GN08] and it is believed that no probabilistic polynomial time algorithm exists neither classic or quantic. The GAPSVP is also conjectured not to have any solving probabilistic polynomial time algorithm in the range of parameters that are useful for cryptography.

Finally we will define a subset of lattices called ideal lattices, in which our encryption scheme will work. They have a more defined basis structure, and therefore we can use some optimizations while computing elements within them. Furthermore, there is no known algorithm that is able to use this structure to help solving the lattice problems we use significantly faster.

Definition 2.18. Given a vector $\mathbf{f} = (f_1, \dots, f_n) \in \mathbb{Z}^n$ we call the *transformation matrix* $\mathbf{F}$ the following matrix:

$$\mathbf{F} = \left(\begin{array}{ccc|c} 0 & \dots & 0 & -f_1 \\ \ddots & & & -f_2 \\ & Id_{n-1} & & \vdots \\ & & \ddots & -f_n \end{array}\right).$$

Definition 2.19. An *ideal lattice* is a lattice $\mathcal{L}$ which basis is the matrix $\mathbf{A} = [\mathbf{a}, \mathbf{F}\mathbf{a}, \dots, \mathbf{F}^{n-1}\mathbf{a}]$, where $\mathbf{F}$ is the transformation matrix of some $\mathbf{f} \in \mathbb{Z}^n$. We note this ideal lattice as $\mathcal{L}(\mathbf{a})$.

Note that for example if $\mathbf{f} = (1, 0, \dots, 0)$ the basis matrix $\mathbf{A}$ is an antyciclic matrix of vector $\mathbf{a} = (a_1, \dots, a_n)$, therefore it will be of the form

$$\mathbf{A} = \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{pmatrix}.$$

In particular the basis matrix, and therefore the lattice, are defined (given a fixed $\mathbf{f}$) by the vector $\mathbf{a}$, so we can store all the information of the lattice on one (or two) vectors. Note that the concepts of q-ary and ideal lattices are not mutually exclusive, and in fact throughout this work we will be using lattices that are both q-ary and ideal.

2.4 Ring Learning With Errors

The Learning with Errors (LWE) problem was introduced by Regev in 2005 in a previous version of [Reg09] as a generalization of the parity learning problem, and gave both a cryptographic protocol based on it and a reduction of its security to a hard lattice problem (the GAPSVP).

However, cryptosystems based on the LWE problem have several issues. For example, many of them need to encrypt bit by bit and primarily the public keys required are very costly to store since they are usually (big) matrices of elements in $\mathbb{Z}_q$. Coupling these two together we get that a lot of storage space is usually needed to encrypt small amounts of information.

To solve these problems, the Ring Learning with Errors variant was introduced by Lyubasevsky, Peikert and Regev in [LPR13]. It is essentially a particular case of LWE but in polynomial rings over finite rings. The problem is over the polynomial ring $R_q = \mathbb{Z}_q[x]/\langle \mathbf{f} \rangle$, where $\mathbf{f}$ is a polynomial in $\mathbb{Z}_q[x]$.

Given an element $\mathbf{a}(x) \in R_q$, one can see the principal ideal generated by $\mathbf{a}(x)$

$$\langle \mathbf{a}(x) \rangle = \{\mathbf{c}(x) \in R_q | \mathbf{c}(x) = \mathbf{a}(x) \cdot \mathbf{b}(x),\ \mathbf{b}(x) \in R_q\}$$

as an ideal lattice in $\mathbb{Z}_q^n$. This correspondence is easy to see in the case $\mathbf{f}(x) = x^n + 1$ (the particular R_q we will use given its specific properties) due to the fact that the vector of coefficients of the product of polynomials in R_q can be found through the anticyclic matrix as follows

$$\mathbf{a}(x) \cdot \mathbf{b}(x) \pmod{x^n + 1} \equiv \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

where $\mathbf{a} = (a_1, \dots, a_n)$ and $\mathbf{b} = (b_1, \dots, b_n)$ are the coefficients of $\mathbf{a}(x)$ and $\mathbf{b}(x)$ respectively.

With this out of the way we can finally define the Ring Learning With Errors problem, on which a lot of lattice-based cryptography is based.

Definition 2.20. Let χ be a probability distribution over R_q and $\mathbf{s} \in R_q$. Then the *R-LWE distribution* $A_{\mathbf{s},\chi}$ is the distribution in $R_q \times R_q$ given by $(\mathbf{a},\ \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$, where $\mathbf{a} \in R_q$ is chosen uniformly at random, $\mathbf{e} \longleftarrow \chi$ and all the operations to compute $\mathbf{b}$ are made in R_q.

Definition 2.21. The *decisional R-LWE problem* is to distinguish samples from $A_{\mathbf{s},\chi}$ from the uniform distribution in $R_q \times R_q$ with a probability that is non-negligibly bigger than $\frac{1}{2}$.

Definition 2.22. The *search R-LWE problem* is to find $\mathbf{s}$ given polinomially many samples from $A_{\mathbf{s},\chi}$ with non-negligible probability.

Therefore given what we have seen in subsection 2.3, we can see that a sample of the R-LWE is a point of an ideal lattice that has been offset by a margin set by the distribution χ (which is normally taken such

that the error is small). So the search R-LWE problem could be seen as finding a point in the ideal lattice $\mathcal{L}(\mathbf{a})$ (remember that a vector $\mathbf{a}$ uniquely defines an ideal lattice through its anticyclic matrix) "close" to the sample, and the decision R-LWE could be seen as given an ideal lattice $\mathcal{L}(\mathbf{a})$, decide whether the points given are all "close" to $\mathcal{L}(\mathbf{a})$ or are uniformly distributed.

2.5 Encryption Scheme and Protocols

Having given all the necessary preliminaries, we can finally present our encryption scheme, threshold decryption protocol and distributed key generation protocol, which we will prove correct in chapter 3, secure in chapter 4 and analyze its implementation in chapter 5. The definitions will be based on the ones stated in [AE20], but will be slightly changed to make them clearer and more comprehensible, and to slightly improve some parameters.

Encryption Scheme 2.23 (Definition 5.2 [AE20])**.** Let $q, n \in \mathbb{Z}_{>0}$ and χ be a distribution over R_q. The *encryption scheme* $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and key generation we will be using is the following:

- $\mathcal{M} = \{0,1\}^n \subseteq \mathbb{Z}_q^n \cong R_q$. We will see every $\mathbf{m} \in \mathcal{M}$ as an element in R_q with $\mathbf{m}$ being its vector of coefficients.
- $\mathcal{C} \subseteq R_q \times R_q$.
- This is a public encryption scheme, we have $\mathcal{K}_s \subseteq R_q$ and $\mathcal{K}_p \subseteq R_q \times R_q$.
  - For any pair of keys $(\mathbf{pk}, \mathbf{s}) \in \mathcal{K}_p \times \mathcal{K}_s$ we will have $\mathbf{s} \longleftarrow \chi$ and $\mathbf{pk} = (\mathbf{a_E}, \mathbf{b_E}) = (\mathbf{a_E}, \mathbf{a_E} \cdot \mathbf{s} + \mathbf{e})$ where $\mathbf{a_E} \xleftarrow{\$} R_q$ and $\mathbf{e} \longleftarrow \chi$.
- $\mathcal{E} = \{\mathsf{E}_{\mathbf{pk}} : \mathbf{pk} = (\mathbf{a_E}, \mathbf{b_E}) \in \mathcal{K}_p\}$ such that given a message $\mathbf{m} \in \mathcal{M}$:

$$\begin{aligned} \mathsf{E}_{\mathbf{pk}} : \mathcal{M} &\to \mathcal{C} \\ \mathbf{m} &\mapsto (\mathbf{u}, \mathbf{v}) \end{aligned}$$

  where $(\mathbf{u}, \mathbf{v}) = (\mathbf{a_E} \cdot \mathbf{r_E} + \mathbf{e_u},\ \mathbf{b_E} \cdot \mathbf{r_E} + \mathbf{e_v} + \mathbf{m} \cdot \lfloor \frac{q}{2} \rfloor)$ with $\mathbf{r_E}, \mathbf{e_u}, \mathbf{e_v} \longleftarrow \chi$.
- $\mathcal{D} = \{\mathsf{D}_{\mathbf{s}} : \mathbf{s} \in \mathcal{K}_s\}$ such that given a ciphertext $(\mathbf{u}, \mathbf{v}) \in \mathcal{C}$:

$$\begin{aligned} \mathsf{D}_{\mathbf{s}} : \mathcal{C} &\to \mathcal{P} \\ (\mathbf{u}, \mathbf{v}) &\mapsto \mathbf{m} \end{aligned}$$

  where we will recover every bit of $\mathbf{m}$ by rounding every coefficient of

$$\mathbf{v} - \mathbf{s} \cdot \mathbf{u} = \mathbf{e} \cdot \mathbf{r_E} + \mathbf{e_v} - \mathbf{s} \cdot \mathbf{e_u} + \mathbf{m} \cdot \left\lfloor \frac{q}{2} \right\rfloor$$

  to 0 or $\lfloor \frac{q}{2} \rfloor$ (mod q) and then mapping 0 to 0 and $\lfloor \frac{q}{2} \rfloor$ to 1.

Now we will define the Threshold Decryption Protocol based on this encryption scheme and a Distributed Key Generation protocol to work together with it. In both protocols we will assume that a commitment scheme is used whenever two players exchange information. For clarity we use a TTP to generate the keys in the encryption protocol, however, what we are looking for is a totally distributed scheme, so we also define a Distributed Key Generation Protocol to take the place of the TTP in the threshold decryption protocol.

Protocol 2.24. Let χ be a distribution over R_q and $\Phi_\cdot(\cdot)$ a pseudo-random function with image in $\mathbb{I}^n$, being $\mathbb{I}$ an integer interval. Then the *Threshold Decryption Protocol* works as follows:

1. A TTP generates the keys $K_H \in \mathbb{Z}_q$ for every subset H of players of size t and distributes them according to the PRSS technique (Technique 2.9). It also generates the secret key $\mathbf{s} \sim \chi$ and the public key $(\mathbf{a_E}, \mathbf{b_E})$ as stated in Encryption Scheme 2.23. Then the TTP sends to the players $(\mathbf{a_E}, \mathbf{b_E})$ and Shamir shares of $\mathbf{s}$. We call $\mathbf{s}^j$ the Shamir share of $\mathbf{s}$ of player P_j, understood as a Shamir share on the vector of coefficients of $\mathbf{s}$.

2. Client receives ciphertext $\mathbf{c} = (\mathbf{u}, \mathbf{v})$, and sends all players $\mathbf{c}$.

3. Each player P_j computes $\tilde{\mathbf{e}}^j = \mathbf{v} - \mathbf{s}^j \cdot \mathbf{u}$ that is a Shamir share of $\tilde{\mathbf{e}} = \mathbf{e} \cdot \mathbf{r_E} + \mathbf{e_v} - \mathbf{s} \cdot \mathbf{e_u} + \lfloor \frac{q}{2} \rfloor \cdot \mathbf{m}$ with $\mathbf{e}, \mathbf{r_E}, \mathbf{e_v}, \mathbf{s}, \mathbf{e_u} \sim \chi$ if every player is honest given both Lemma 2.6 and Lemma 2.7.

4. Each player P_j computes its additive share of $\mathbf{x} := \sum_H \Phi_{K_H}(\mathbf{c})$ for every subset of $t+1$ players P_j belongs to in the following way: in order, the additive share $\mathbf{x}^j$ is the sum of all $\Phi_{K_H}(\mathbf{c})$ no player before has, but P_j does.

5. For every subset of $t+1$ players, they convert the additive share $\mathbf{x}^j$ share to $\tilde{\mathbf{x}}^j$ Shamir share of $\mathbf{x}$ using Technique 2.11.

6. Each player P_j computes, for every subset of $t+1$ players it belongs to, $\tilde{\mathbf{x}}^j + \tilde{\mathbf{e}}^j$ Shamir share of the vector of coefficients of $\mathbf{x} + \tilde{\mathbf{e}} \in R_q$, where $\mathbf{x} + \tilde{\mathbf{e}} \in R_q$ is understood as the polynomial in R_q with vector of coefficients $\mathbf{x} + \tilde{\mathbf{e}}$, and sends it to the client.

7. Client reconstructs $\mathbf{x} + \tilde{\mathbf{e}}$ for every subset of $t+1$, picks whichever value is repeated more times, then for every coefficient returns 0 if $\mathbf{x} + \tilde{\mathbf{e}}$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ and returns 1 otherwise.

Protocol 2.25. Let χ^{KG} be a distribution over R_q, $\mu = x + 2x^2 + \ldots + (n-1)x^{n-1} \in R_q$, and $\Phi_\cdot^{KG}(\cdot)$ a pseudo-random function with image in $\mathbb{I}_{KG}^n$, where $\mathbb{I}_{KG}$ is an integer interval. The *Distributed Key Generation Protocol* works as follows:

1. For the secret key $\mathbf{s} \in R_q$, each player P_j chooses its contribution $\mathbf{s}^j = (s_1^j, \ldots, s_n^j)$ with $\mathbf{s}^j \sim \chi^{KG}$. Then they act as the dealer in a NIVSS (Technique 2.10) to share every s_i^j to all players. All players verify the value broadcast when doing the NIVSS $s_i^j - \sum_H \phi_{K_H}^{KG}(\mu)$ is in the interval $\binom{u}{t}\mathbb{I}_{KG}$. Now all players have shares of every s_i^j and by their linearity also of $s_i = \sum_j s_i^j$. Then $\mathbf{s}$ is the polynomial in R_q with coefficients $(s_1, \ldots, s_n)$.

2. For the keys $K_H \in \mathbb{Z}_q$ that will be used for the PRSS in the threshold decryption, for every subset H of t players each player P_j chooses uniformly at random $K_{H_j} \in \mathbb{Z}_q$ their contribution on these keys and shares it with all the players using Shamir secret sharing (Technique 2.5). Then the players will have, by adding all the shares received by other players Shamir shares of $K_H = \sum_j K_{H_j}$. Finally all players send privately their shares on K_H to all the players in A the complement of H, so they can recover K_H.

3. For the contributions to $\mathbf{e} \in R_q$ proceed identically to when generating $\mathbf{s}$.

4. For $\mathbf{a_E} \in R_q$ every player P_j chooses its share $(a_{E,1}^j, \ldots, a_{E,n}^j)$ randomly in R_q^n and does a Shamir share of it. Then all players send to all players their share on all the $(a_{E,1}^j, \ldots, a_{E,n}^j)$ so every player

can recover (by adding the shares) $\left(\sum_j a^j_{E,1}, \dots, \sum_j a^j_{E,n}\right)$. The polynomial in R_q with coefficients $\left(\sum_j a^j_{E,1}, \dots, \sum_j a^j_{E,n}\right)$ will be $\mathbf{a_E}$.

5. Every player computes locally their Shamir shares on $\mathbf{b_E} = \mathbf{a_E} \cdot \mathbf{s} + \mathbf{e}$ by performing these same operations with the shares they have on $\mathbf{s}$ and $\mathbf{e}$ (having previously converted the shares to Shamir).

Finally, we have given the protocols with complete generality, but for the rest of the work we will be assuming the following constraints on the parameters, distributions and intervals:

- λ the security parameter.
- $n \in \mathbb{Z}_{>0}$ the dimension of the lattice.
- $q = 2^{\Theta(\lambda)}$ the integer modulo.
- $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ the ring of integers where we will work.
- $\ell \in \mathbb{Z}_{>0}$ the number of samples of encrypted messages expected.
- $u \in \mathbb{Z}_{>0}$ the number of players and $t \in \mathbb{Z}_{>0},\ t < u$ the number of corrupted players.
- $\alpha > 0$ and $\xi = \alpha \left(\frac{n\ell}{\log(n\ell)}\right)^{\frac{1}{4}}$ the parameters of the distributions.
- $\chi = \overline{\Psi}^n_{\frac{\xi}{q}}$ and $\chi^{KG} = \overline{\Psi}^n_{\frac{\xi}{q\sqrt{u}}}$ (these distributions will be properly defined in chapter 3).

3. Discrete Gaussian Measures and Correctness

As we have hinted after Definition 2.1, for any encryption scheme we define to be useful, we need to prove two significant properties: correctness and security. As such, since in [AE20] we proposed a new encryption scheme and threshold decryption protocol, we proved both these properties. However, the proofs could be greatly improved in various ways, for example clarity, specificity and structure. Therefore this chapter and the following chapter 4 will be presented in an effort towards giving better proofs of these properties. In this chapter we will tackle correctness.

From what we have seen before, the R-LWE problem relies on a small error distribution (see Definition 2.20) to cause slight disturbances to make solving a system of equations hard. However, one can consider the R-LWE for any error distribution χ^n, even if not all distributions will make solving the system of equations equally hard. Therefore, one may ask the question: which error distributions cause the R-LWE problem to be hard to solve?

The first requirement needed to answer this question is to properly define what the hardness of a problem is. For cryptographic purposes the usual approach is to give a reduction from a well-known and widely studied problem known, or at least supposed, to be hard (more detail on hardness reductions on chapter 4).

Therefore, the more accurate question to answer is: which error distributions yield reductions to R-LWE from a well-known and widely studied problem known, or at least supposed to be hard? And the answer is discrete Gaussian measures.

3.1 Lattice Discrete Gaussian Distribution

The first definition of a discrete Gaussian measure is, in broad strokes, taking the probability of the normal distribution over $\mathbb{R}^n$ at the points of $\mathbb{Z}^n$ and then multiplying that by some constant to make this a proper random variable. This is the distribution most used in places where a trapdoor (or one-way) function is needed to sample lattice points following a narrow distribution. As such it has been studied plenty and has well-known bounding results, though it does have some disadvantages as we will discuss afterwards. However, it is still useful to see how the bounds are proven.

3.1.1 Definition

This is how the discrete Gaussian distribution over $\mathbb{Z}^n$ is defined in [Lyu12] (going first through the continuous Gaussian distribution).

Definition 3.1 (Definition 3.1, 3.2 [Lyu12])**.** The *continuous spherical Gaussian distribution* over $\mathbb{R}^n$ centered at $\mathbf{v} \in \mathbb{R}^n$ and with standard deviation $\sigma > 0$ is defined by the density function

$$\rho^n_{\mathbf{v},\sigma}(\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\frac{\|\mathbf{x}-\mathbf{v}\|_2^2}{2\sigma^2}}$$

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

The *discrete spherical Gaussian distribution* over $\mathbb{Z}^n$ centered at $\mathbf{v} \in \mathbb{Z}^n$ with parameter σ is defined by the probability function

$$D^n_{\mathbf{v},\sigma}(\mathbf{x}) = \frac{\rho^n_{\mathbf{v},\sigma}(\mathbf{x})}{\rho^n_{\mathbf{v},\sigma}(\mathbb{Z}^n)}$$

with $\mathbf{x} \in \mathbb{Z}^n$ and $\rho^n_{\mathbf{v},\sigma}(\mathbb{Z}^n) = \sum_{\mathbf{z}\in\mathbb{Z}^n} \rho^n_{\mathbf{v},\sigma}(\mathbf{z})$ being the scaling coefficient necessary for $D_{\mathbf{v},\sigma}$ to be indeed a probability distribution.

Remark 3.2. We would like to note that in the discrete Gaussian measure, σ is not the standard deviation. In fact, computing the standard deviation of $D^n_{\mathbf{v},\sigma}$ is not easy. However, what is needed for correctness is that the tails of the distribution are bounded, and this can be seen through the parameter σ without needing the standard deviation. We would also like to note that from now on we will drop the "spherical" term when referring to it, since we will only use them and no elliptical distribution (the only difference is that with elliptical distributions each direction of the base has a different parameter).

We will always use distributions centered at $\mathbf{0}$, so we can even see more. Since now we have $\mathbf{v} = \mathbf{0}$, we can see that (omitting the $\mathbf{0}$ in the subtext for ease of notation)

$$\rho^n_\sigma(\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\sum_{i=1}^n -\frac{x_i^2}{2\sigma^2}} = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x_i^2}{2\sigma^2}} = \prod_{i=1}^n \rho_\sigma(x_i)$$

where when the dimension is 1 we do not use an exponent to ease the notation. Then we can infer that

$$D^n_\sigma \sim D_\sigma \times \cdots \times D_\sigma.$$

Therefore we can reduce ourselves to the study of D_σ over $\mathbb{Z}$ instead of having to deal with $\mathbb{Z}^n$.

Another topic we need to discuss is the fact that our lattices are not defined over the whole $\mathbb{Z}^n$ but rather over $\mathbb{Z}^n_q$. Therefore we will deal with the reduction modulo q of D_σ defined as standard:

$$\hat{D}_\sigma(i) = \sum_{k\in\mathbb{Z}} D_\sigma(i + kq)$$

for $i \in \mathbb{Z}_q$.

3.1.2 Bound

Even if the objective of the error distribution χ^n is to muddle the system of equations to make the solving difficult, we do not want the system to be too diffused, otherwise we would not be able to recover the information. In other words, if the distribution is too similar to being uniform, the information cannot be recovered even with the secret key.

Therefore, we need to bound the tails of our distribution, and given what we have seen before, a bound of $|\hat{D}_\sigma|$ would be enough. Then we note that given any distribution X over $\mathbb{Z}$ and $\hat{X}$ its standard reduction modulo q, with representatives $-\lfloor \frac{q}{2} \rfloor, \dots, \lfloor \frac{q}{2} \rfloor$, then

$$\Pr(|\hat{X}| > i) \leq \Pr(|X| > i)$$

for any i in $\mathbb{Z}_q$, since all elements between kq and $kq + i$ for $k \in \mathbb{Z}_{>0}$ are counted on the right but not on the left, and each of these probabilities is non-negative. We cannot say that it is strictly smaller since if the support of X is $\mathbb{Z}_q$ we will have equality, but otherwise the inequality is strict.

This means that we only need a bound on $|D_\sigma|$, and we will give a particular case of the bound given in Lemma 3.3 in [Lyu12] in $\mathbb{Z}$, since we do not need $\mathbb{Z}^n$.

Lemma 3.3. *For any $s, \sigma > 0$ and let $T = D_\sigma$ (to ease notation), then*

$$\mathbb{E}\left[e^{\frac{s}{\sigma^2}T}\right] \leq e^{\frac{s^2}{2\sigma^2}}.$$

Proof. Let $s, \sigma > 0$, then

$$\mathbb{E}\left[e^{\frac{s}{\sigma^2}T}\right] = \sum_{z\in\mathbb{Z}} \Pr(T=z)e^{\frac{sz}{\sigma^2}} = \sum_{z\in\mathbb{Z}} T(z)e^{\frac{sz}{\sigma^2}} = \sum_{z\in\mathbb{Z}} \frac{\rho_\sigma(z)}{\rho_\sigma(\mathbb{Z})}e^{\frac{sz}{\sigma^2}} =$$
$$= \sum_{z\in\mathbb{Z}} \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{\overline{z}\in\mathbb{Z}} e^{-\frac{\overline{z}^2}{2\sigma^2}}} e^{\frac{2sz}{2\sigma^2}} = \sum_{z\in\mathbb{Z}} \frac{e^{-\frac{(z-s)^2}{2\sigma^2}}}{\sum_{\overline{z}\in\mathbb{Z}} e^{-\frac{\overline{z}^2}{2\sigma^2}}} e^{\frac{s^2}{2\sigma^2}} =$$
$$= e^{\frac{s^2}{2\sigma^2}} \frac{\sum_{z\in\mathbb{Z}} e^{-\frac{(z-s)^2}{2\sigma^2}}}{\sum_{\overline{z}\in\mathbb{Z}} e^{-\frac{\overline{z}^2}{2\sigma^2}}} \leq e^{\frac{s^2}{2\sigma^2}}$$

where the last inequality is derived from Lemma 2.9 on [MR07], giving us what we wanted to see. □

Now we can use this inequality to prove the bound on the probability of $|D_\sigma|$.

Lemma 3.4. *For any $\sigma, k > 0$ then,*

$$\Pr\left(|D_\sigma| > k\right) \leq 2e^{-\frac{k^2}{2\sigma^2}}.$$

Proof. Let $\sigma, k > 0$ and using the same notation as before $T = D_\sigma$, then

$$\Pr\left(|D_\sigma| > k\right) = 2\Pr\left(D_\sigma > k\right) = 2\Pr\left(e^{\frac{s}{\sigma^2}T} > e^{\frac{s}{\sigma^2}k}\right) \leq 2\frac{\mathbb{E}\left[e^{\frac{s}{\sigma^2}T}\right]}{e^{\frac{s}{\sigma^2}k}} \leq 2\frac{e^{\frac{s^2}{2\sigma^2}}}{e^{\frac{sk}{\sigma^2}}} = 2e^{\frac{s^2}{2\sigma^2}-\frac{sk}{\sigma^2}}$$

where we have used (in this order) that D_σ is symmetric with respect to zero, the Markov inequality and Lemma 3.1.2. Since the inequality holds for any s it holds true for $s = k$ in particular, which is the minimum of the function $f(s) = \frac{s^2}{2\sigma^2} - \frac{sk}{\sigma^2}$. This means that the result is the best bound of this type we can find. □

3.2 Rounded Discrete Gaussian Distribution

The discrete Gaussian distribution has many benefits, for example the ease to give mathematical reductions or tail bounds that are easy to prove, but has one major drawback. Sampling from D_σ is difficult to compute, to the point that all the known ways to sample at the moment are either efficient or secure, with this or being exclusive [HLS18]. Therefore, one would need to either sacrifice computation time or security if you want to use D_σ on an implementation.

That is why we will use another discrete Gaussian distribution, similar enough to D_σ that some of the benefits are still there, but different enough so that there is an efficient and secure way to sample it (as long as there is an efficient way to sample a continuous Gaussian distribution over $\mathbb{R}$). This will be the distribution we will use further on, so from now on every time we mention a discrete Gaussian distribution will be this one unless the opposite is stated.

3.2.1 Definition

This is how the discrete Gaussian distribution over $\mathbb{Z}_q$ is defined in [Reg09] (going through a reduction modulo 1).

Definition 3.5 ([Reg09])**.** The *continuous Gaussian distribution* over $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ with parameter $\sigma > 0$ is defined by the density function

$$\Psi_\sigma(r) = \sum_{k \in \mathbb{Z}} \rho_\sigma(r - k)$$

for $r \in [0, 1)$, where ρ_σ is the Gaussian function over $\mathbb{R}$ defined on section 3.1.1.

The *discrete Gaussian distribution* over $\mathbb{Z}_q$ with parameter $\sigma > 0$ is defined by the probability function

$$\overline{\Psi}_\sigma(i) = \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}} \Psi(r)dr$$

for $i \in \mathbb{Z}_q$. Note that in this definition we are taking $\{0, \dots, q-1\}$ as representatives for $\mathbb{Z}_q$, and for 0 the integral is over $[0, \frac{1}{2}] \cup [q - \frac{1}{2}, q]$.

Remark 3.6. Note that the remark 3.2 about the parameter not being the standard deviation of the distribution but being enough to give a bound applies in the same way for this distribution.

We would like to emphasize why this new discrete Gaussian is efficiently sampled given an efficient sampling of a normal distribution. From the definition of the distribution over $\mathbb{T}$ it can be seen that for every $r \in [0, 1)$ we are assigning to r the probability of all the real numbers that have r as their decimal part, so if $Y \sim \Psi_\sigma$ then $Y = X \pmod 1$ where $X \sim N(0, \sigma)$ and reducing modulo 1 means taking only the decimal part. Furthermore, from the definition of the distribution over $\mathbb{Z}_q$, it is clear that if $Z \sim \overline{\Psi}_\sigma$ then $Z = \lfloor qY \rceil \pmod q$ with $Y \sim \Psi_\sigma$. And both taking the decimal part, multiplying by q and rounding can be efficiently computed.

Similarly as with the lattice discrete Gaussian distribution, we need the distribution to be over $\mathbb{Z}_q^n$. To do so we define the distribution

$$\overline{\Psi}_\sigma^n = \overline{\Psi}_\sigma \times \cdots \times \overline{\Psi}_\sigma$$

over $\mathbb{Z}_q^n$ as we wanted. And as before, studying $\overline{\Psi}_\sigma$ gives us everything we need about $\overline{\Psi}_\sigma^n$.

3.2.2 Bound

As in section 3.1.2 we want to bound the tails of the distribution. However, we cannot use the same method used in that section, since our distribution now is not a reduction modulo q from a distribution over $\mathbb{Z}$, but instead comes from a distribution over $\mathbb{T}$.

To overcome this, we will define yet another distribution over $\mathbb{Z}_q$ which is easier to bound and prove that it is equivalent to $\overline{\Psi}_\sigma$ for some σ.

Therefore now we define the rounded Gaussian distribution over $\mathbb{Z}$.

Definition 3.7. The *rounded Gaussian distribution* over $\mathbb{Z}$ with parameter $\sigma > 0$ is defined by the probability function

$$\Omega_\sigma(z) = \int_{z-\frac{1}{2}}^{z+\frac{1}{2}} \rho_\sigma(x)dx$$

for $z \in \mathbb{Z}$ and ρ_σ the Gaussian function over $\mathbb{R}$ defined on section 3.1.1.

The distribution we are interested in is $\hat{\Omega}_\sigma$, its standard reduction modulo q as it is defined in section 3.1.2. It is clear once again from the definition that if $Y \sim \Omega_\sigma$, then $Y = \lfloor X \rceil$ with $X = N(0, \sigma)$, hence the name rounded Gaussian.

Now we can see the relation between $\hat{\Omega}$ and $\overline{\Psi}$.

Lemma 3.8. *For any $\sigma \in \mathbb{R}$ we have that $\hat{\Omega}_\sigma$ is indeed a random variable and in fact we have that $\hat{\Omega}_\sigma = \overline{\Psi}_{\frac{\sigma}{q}}$.*

Proof. First we need to see that $\hat{\Omega}_\sigma$ is a random variable. Indeed

$$\sum_{i=0}^{q-1} \hat{\Omega}_\sigma(i) = \sum_{i=0}^{q-1} \sum_{k\in\mathbb{Z}} \Omega_\sigma(i+kq) = \sum_{k\in\mathbb{Z}} \sum_{i=0}^{q-1} \Omega_\sigma(i+kq) = \sum_{\overline{k}\in\mathbb{Z}} \Omega_\sigma(\overline{k}) =$$
$$= \sum_{\overline{k}\in\mathbb{Z}} \int_{\overline{k}-\frac{1}{2}}^{\overline{k}-\frac{1}{2}} \rho_\sigma(x)dx = \int_{-\infty}^{+\infty} \rho_\sigma(x)dx = 1.$$

Now we can see that $\hat{\Omega}_\sigma(i) = \overline{\Psi}_{\frac{\sigma}{q}}(i)$ for all $i \in \mathbb{Z}_q$, and therefore $\hat{\Omega}_\sigma = \overline{\Psi}_{\frac{\sigma}{q}}$ as random variables.

$$\hat{\Omega}_\sigma(i) = \sum_{k\in\mathbb{Z}} \Omega_\sigma(i+kq) = \sum_{k\in\mathbb{Z}} \int_{i+kq-\frac{1}{2}}^{i+kq+\frac{1}{2}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x}{\sqrt{2}\sigma}\right)^2} dx =$$
$$= \sum_{k\in\mathbb{Z}} \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{q(y+k)}{\sqrt{2}\sigma}\right)^2} q \cdot dy = \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}} \sum_{k\in\mathbb{Z}} \frac{1}{\sqrt{2\pi}\frac{\sigma}{q}} e^{-\left(\frac{y+k}{\sqrt{2}\frac{\sigma}{q}}\right)^2} dy = \overline{\Psi}_{\frac{\sigma}{q}}(i)$$

where we have used the change of variables $y = \frac{x-kq}{q}$ and the dominated convergence theorem. □

Therefore, if we know a bound for $\hat{\Omega}$, we know a bound for $\overline{\Psi}$. Given this result, we can now bound the distributions using the fact that Ω is a rounded Gaussian and Mill's inequality:

$$\Pr(|N(0,\sigma)| > t) = 2\int_t^{+\infty} \rho_\sigma(x)dx \leq \sqrt{\frac{2}{\pi}} \frac{e^{-\left(\frac{t}{\sqrt{2}\sigma}\right)^2}}{t}.$$

Lemma 3.9. *For all $c, \sigma > 0$ then,*

$$\Pr\left(\left|\overline{\Psi}_{\frac{\sigma}{q}}\right| > c\right) < 2\sqrt{\frac{2}{\pi}} e^{-\left(\frac{\lceil c\rceil - \frac{1}{2}}{\sqrt{2}\sigma}\right)^2}.$$

Proof. Let $c, \sigma > 0$, then

$$\Pr\left(\left|\overline{\Psi}_{\frac{\sigma}{q}}\right| > c\right) = \Pr\left(\left|\hat{\Omega}_\sigma\right| > c\right) < \Pr(|\Omega_\sigma| > c) = 2\sum_{j=\lceil c\rceil}^{+\infty} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \rho_\sigma(x)dx =$$
$$= 2\int_{\lceil c\rceil-\frac{1}{2}}^{+\infty} \rho_\sigma(x)dx \leq \sqrt{\frac{2}{\pi}} \frac{e^{-\left(\frac{\lceil c\rceil-\frac{1}{2}}{\sqrt{2}\sigma}\right)^2}}{\lceil c\rceil - \frac{1}{2}} \leq 2\sqrt{\frac{2}{\pi}} e^{-\left(\frac{\lceil c\rceil-\frac{1}{2}}{\sqrt{2}\sigma}\right)^2}$$

where we have used (in this order) Lemma 3.8, the reasoning in section 3.1.2 (note that now we can say that it is strictly smaller since the support of Ω_σ is $\mathbb{Z}$), the fact that Ω_σ is symmetric respect 0, the previous known result and the fact that $\lceil c\rceil - \frac{1}{2} \geq \frac{1}{2}$ for any $c > 0$, thus seeing what we wanted. □

3.3 Correctness

This whole section is motivated by the fact that a basic property needed by any encryption scheme is correctness. In broad strokes, what is needed is that for any ciphertext received, the probability of incorrectly recovering the original message given the secret key is negligibly small. More formally we can put it the following way.

Definition 3.10. Let $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. The encryption scheme is said to be *correct* if for all $e \in \mathcal{K}_p$ exists some computable $d \in \mathcal{K}_s$ such that with λ the security parameter

$$\Pr(D_d(E_e(m)) \neq m) = \text{ neg}(\lambda)$$

for all $m \in \mathcal{M}$.

In our particular encryption scheme (defined in section 2.5), we need to change a little bit this definition. Since we are executing the protocol against an active adversary (an adversary that can arbitrarily deviate from the protocol), we will have $\binom{u}{t+1}$ decryptions of every message, and we need to see that a majority is correct. Therefore we need that a majority of reconstructions satisfy that $\Pr((|\mathbf{x} + \hat{\mathbf{e}}|)_i > \frac{q}{4})$ is negligible for all i, where $\hat{\mathbf{e}} = \mathbf{e} \cdot \mathbf{r_E} + \mathbf{e_v} - \mathbf{s} \cdot \mathbf{e_u}$ and $(\cdot)_i$ denotes the ith coefficient of an element in R_q. To do so we will use the bound given in Lemma 3.9, but we still need two more technical results to be able to properly finish the proof. Let us prove them.

Lemma 3.11. *Let A, B, C three random variables such that $C \leq A + B$ and A, B are independent. Then,*

$$\begin{cases} \Pr(A > a) < \lambda_a \\ \Pr(B > b) < \lambda_b \end{cases} \implies \Pr(C > a + b) < \lambda_a + \lambda_b.$$

Proof. Let A, B, C three random variables such that $C \leq A + B$. Then note that $C > a + b$ if and only if either $A > a$ or $B > b$, therefore the event $C > a + b$ is contained in the event $(A > a) \cup (B > b)$, and so $\Pr(C > a + b) < \Pr((A > a) \cup (B > b))$. Then we get that:

$$\begin{aligned} \Pr(C > a + b) < \Pr((A > a) \cup (B > b)) = \Pr(A > a) + \Pr(\mathrm{B} > \mathrm{b}) - \Pr(A > a)\Pr(B > b) \leq \\ \leq \Pr(A > a) + \Pr(B > b) < \lambda_a + \lambda_b \end{aligned}$$

as we wanted to see. □

Remark 3.12. Note that this is generally a very coarse bound on the random variable C, but we do not need the bound to be any tighter for the proof of correctness.

Corollary 3.13. *Let $A_1, \ldots, A_n$ independent random variables such that $\Pr(A_i > a) < \lambda$, and let C another random variable such that $C \leq \sum_{i=1}^{n} A_i$. Then $\Pr(C > na) < n\lambda$.*

Proof. Direct by using induction on Lemma 3.11. □

Lemma 3.14. *Let $A, B, C \geq 0$ three random variables such that $C \leq A \cdot B$, and A, B are independent. Then,*

$$\begin{cases} \Pr(A > a) < \lambda_a \\ \Pr(B > b) < \lambda_b \end{cases} \implies \Pr(C > a \cdot b) < \lambda_a + \lambda_b.$$

Proof. Let A, B, C three random variables such that $C \leq A \cdot B$. Then, in an analogous way to Lemma 3.11, note that $C > a \cdot b$ if and only if either $A > a$ or $B > b$, therefore the event $C > a \cdot b$ is contained in the event $(A > a) \cup (B > b)$, and so $\Pr(C > a \cdot b) < \Pr((A > a) \cup (B > b))$. Then we get, as in the proof of Lemma 3.11 that:

$$\Pr(C > a \cdot b) < \lambda_a + \lambda_b$$

as we wanted to see. □

Remark 3.15. Once again, this is a generally very coarse bound, but we do not need it any tighter to prove correctness.

With these technical results in hand we can tackle the proof of correctness in our encryption scheme, redoing the proof given in [AE20].

Theorem 3.16 (Theorem 5.5 [AE20])**.** *Let $c = \Omega(\sqrt{\lambda})$, $0 < d < 1$ and let $\mathbb{I}^n = [-(c\xi)^2(2n+1)q^d,\ (c\xi)^2(2n+1)q^d]^n$ be the interval image of $\Phi_\cdot(\cdot)$. Assume that $1 < \xi < \frac{1}{c}\sqrt{\frac{q}{4(2n+1)\left(\binom{u}{t}q^d+1\right)}}$ Then the decryption protocol will have correct output except with negligible probability.*

Proof. Let $\hat{\mathbf{e}} = \mathbf{e} \cdot \mathbf{r_E} + \mathbf{e_v} - \mathbf{s} \cdot \mathbf{e_u}$. Like we have said before, what we want to see is that $\Pr((|\mathbf{x} + \hat{\mathbf{e}}|)_i > \frac{q}{4})$ is negligible $\forall i$.

Since the product in R_q is done through the anticyclic matrix (as we have seen in section 2.4), we know that:

$$(|\hat{\mathbf{e}}|)_i \leq |e_i \cdot r_{E_1}| + |e_{i-1} \cdot r_{E_2}| + \ldots + |e_{i+2} \cdot r_{E_{n-1}}| + |e_{i+1} \cdot r_{E_n}| + |e_{v_i}| + |s_i \cdot e_{u_1}| + \ldots + |s_{i+1} \cdot e_{u_n}|$$

and therefore, since $\mathbf{e}, \mathbf{r_E}, \mathbf{e_v}, \mathbf{s}, \mathbf{e_u} \sim \overline{\Psi}^n_{\frac{\xi}{q}}$, if we have that $\Pr\left(\left|\overline{\Psi}_{\frac{\xi}{q}}\right| > k\right) < \mu$ for some k, then using Corollary 3.13 and Lemma 3.14 we get that

$$\Pr((|\hat{\mathbf{e}}|)_i > 2nk^2 + k) < (4n+1)\mu. \tag{1}$$

From Lemma 3.9 we know that

$$\Pr\left(|\overline{\Psi}_{\frac{\xi}{q}}| > c\xi\right) < 2\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\lceil c\xi \rceil - \frac{1}{2}}{\sqrt{2}\xi}\right)^2} \tag{2}$$

so we have that $k = c\xi$.

We also know that by construction:

$$(|\mathbf{x}|)_i \leq \binom{u}{t}(c\xi)^2(2n+1)q^d. \tag{3}$$

And by hypothesis:

$$\xi < \frac{1}{c}\sqrt{\frac{q}{4(2n+1)\left(\binom{u}{t}q^d+1\right)}} \Rightarrow k^2 < \frac{q}{4(2n+1)\left(\binom{u}{t}q^d+1\right)} \Rightarrow k^2(2n+1)\left(\binom{u}{t}q^d+1\right) < \frac{q}{4}. \tag{4}$$

Therefore giving us:

$$\begin{aligned}
\Pr\left((|\mathbf{x}+\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) &\leq \Pr\left((|\mathbf{x}|)_i + (|\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) < \\
&< \Pr\left((|\mathbf{x}|)_i + (|\hat{\mathbf{e}}|)_i > k^2(2n+1)\left(\binom{u}{t}q^d + 1\right)\right) = \\
&= \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(2n+1)\left(\binom{u}{t}q^d + 1\right) - (|\mathbf{x}|)_i\right) \leq \\
&\leq \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(2n+1)\left(\binom{u}{t}q^d + 1\right) - \binom{u}{t}k^2(2n+1)q^d\right) = \\
&= \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(2n+1)\right) < \Pr\left((|\hat{\mathbf{e}}|)_i > 2nk^2 + k\right) < \\
&< (4n+1)2\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\lceil c\xi\rceil - \frac{1}{2}}{\sqrt{2}\xi}\right)^2}
\end{aligned}$$

where we have used (in this order) the triangular inequality, the final inequality in (4), the inequality in (3), the fact that $c, \xi > 1$ and therefore $k > 1$ so $k^2 > k$ and the inequalities (1) and (2) coupled together.

And finally, using that $c = \Omega(\sqrt{\lambda})$ and $\xi > 1$ we get that

$$\Pr\left((|\mathbf{x}+\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) = \text{neg}(\lambda).$$

□

We would like to note that the proof of this theorem is only valid when we can ensure that $\mathbf{s}$ and $\mathbf{e}$ follow the distribution $\overline{\Psi}$. However, Protocol 2.25 does not generate this distribution. How we deal with this will be explored with more detail in chapter 4.

4. Definitions and Proofs of Security

This section will be laid out in an aim to prove security of Encryption Scheme 2.23 and both Protocol 2.24 and 2.25. What we want to achieve in respect to the security proofs in [AE20] is twofold: on the one hand we want to completely remake the proofs in a more standard fashion, since security proofs usually do not revolve around functionalities but rather attack games; and in the other hand we also want to fill some little holes left unexplained in [AE20] so as to give more comprehensible and autocontained results. Let us get into it.

4.1 Definitions of Security

In contrast with correctness of an encryption scheme, security has many different ways in which it can be defined. This is due to the fact that you always want a decryption to be correct (or always except with negligible probability) but security depends on how is the adversary we want to defend us against (information available, computational power) and what we want to ensure (that the adversary cannot know what message was encrypted or that he cannot distinguish which message has been encrypted from a pool of plaintexts).

These different "securities" are defined revolving around an attack game between the challenger and the adversary, in which we say that an encryption satisfies that type of security if the advantage of the adversary in this particular game is negligible.

The following definitions of the attack games, advantages and securities are taken from [BS20], and we will move from weaker to stronger notions of security.

4.1.1 Message Recovery Security

This security notion is quite straightforward from its name, you do not want the adversary to recover the message associated to a given ciphertext. Let us properly define its attack game and advantage.

Attack Game 4.1 (Attack Game 2.2 [BS20])**.** Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Given an adversary $\mathcal{A}$, the *message recovery attack game* proceeds as follows:

- The challenger computes $m \xleftarrow{\$} \mathcal{M}, e \xleftarrow{\$} \mathcal{K}_p, c = E_e(m)$, and sends c to $\mathcal{A}$. In case of public key encryption schemes the challenger also sends e to the adversary.
- The adversary outputs a message $\hat{m} \in \mathcal{M}$.

Let W be the event in which $\hat{m} = m$, then we define $\mathcal{A}$'s *message recovery advantage* as:

$$\mathrm{MRAdv}[\mathcal{A}, \mathcal{S}] := \left| \Pr(W) - \frac{1}{|\mathcal{M}|} \right|.$$

Definition 4.2 (Definition 2.3 [BS20])**.** An encryption scheme $\mathcal{S}$ is said to be *secure against message recovery attacks* if for all efficient adversaries $\mathcal{A}$, the value $\mathrm{MRAdv}[\mathcal{A}.\mathcal{S}]$ is negligible.

4.1.2 Semantic Security

Message recovery security is the weakest notion of security, an adversary could be able to know information of the message without being able to specifically recover the message. For example, the parity of the message could be leaked or other properties about it. That is the reason why semantic security was defined.

In general terms, what semantic security wishes to achieve is that the adversary cannot distinguish if an encrypted message is one of two selected by $\mathcal{A}$. Let us define its attack game and advantage.

Attack Game 4.3 (Attack Game 2.1 [BS20])**.** Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Given an adversary $\mathcal{A}$, the *semantic security attack game*, after the challenger computes $b \xleftarrow{\$} \{0,1\}$, follows with the following experiment b:

- The adversary computes two same-length messages $m_0 \neq m_1 \in \mathcal{M}$ and sends them to the challenger.
- The challenger computes $e \xleftarrow{\$} \mathcal{K}_p, c = E_e(m_b)$ and sends c to $\mathcal{A}$. In case of public key encryption schemes the challenger also sends e to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0,1\}$.

Let W_b be the event in which $\mathcal{A}$ outputs 1 in the experiment b, then we define $\mathcal{A}$'s *semantic security advantage* as:

$$\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}] := |\Pr(W_0) - \Pr(W_1)|.$$

Definition 4.4 (Definition 2.2 [BS20])**.** An encryption scheme $\mathcal{S}$ is said to be *semantically secure* if for all efficient adversaries $\mathcal{A}$, the value $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ is negligible.

Note that in the public key encryption cases, where the public key is known by the adversary, $\mathcal{S}$ cannot be semantically secure if its encryption function is deterministic. Therefore we will need to add some measure of randomness to ensure security, that is the reason why we use samples from random variables.

We also need to formally see that indeed this is a stronger security notion that security against message recovery attacks, so we will see semantic security implies security against message recovery attacks. We will give a version of the proof given in [BS20].

Lemma 4.5 (Theorem 2.7 [BS20])**.** *Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Then if $\mathcal{S}$ is semantically secure, $\mathcal{S}$ is secure against message recovery attacks.*

Proof. We will prove the contrapositive: that if $\mathcal{S}$ is not secure against message recovery attacks, then it is not semantically secure. In other words, we want to see that given access to an efficient adversary $\mathcal{A}$ such that it has non-negligible message recovery advantage, we can construct another efficient adversary $\mathcal{B}$ that interacts with $\mathcal{A}$ as the challenger and has non-negligible semantic security advantage.

Let $\mathcal{B}$ be an adversary of the semantic security attack game. Its behaviour is as follows:

- Generate $m_0, m_1 \xleftarrow{\$} \mathcal{M}$ and send them to the challenger of the semantic security attack game.
- Receive from the challenger $c = E_e(m_b)$ (and if we are in public key encryption receive e too).
- Act as the challenger in a message recovery attack game with $\mathcal{A}$ and as such send $\mathcal{A}$ the ciphertext c (and e if we are in public key encryption).

- Receive $\hat{m}$ from $\mathcal{A}$ and return to the challenger 0 if $\hat{m} = m_0$ and 1 if $\hat{m} = m_1$, abort otherwise.

Now we need to compute the semantic security advantage for $\mathcal{B}$. Note that the probability of $\mathcal{B}$ outputting 1 in experiment 1 is $\Pr(\hat{m} = m_1)$, and that in experiment 0 in the semantic security game, the message recovery game with $\mathcal{A}$ is independent of m_1, therefore the probability of $\mathcal{A}$ outputting m_1 is $\frac{1}{|\mathcal{M}|}$. Adding it all together we get:

$$\mathrm{SSAdv}[\mathcal{B}, \mathcal{S}] = |\Pr(W_0) - \Pr(W_1)| = \left| \frac{1}{|\mathcal{M}|} - \Pr(\hat{m} = m_1) \right| = \mathrm{MRAdv}[\mathcal{A}, \mathcal{S}] = \text{ non-negligible}$$

as we wanted to see. □

Apart from this way to define the semantic security advantage, there is another equivalent way to do so which may be easier to use in some situations. The attack game is the same as Attack Game 4.3, but we define a new advantage.

Definition 4.6 (Attack Game 2.4 [BS20])**.** Let W be the event in which $\hat{b} = b$. Then we define the *bit-guessing semantic security advantage* as

$$\mathrm{SSAdv}^*[\mathcal{A}, \mathcal{S}] = \left| \Pr(W) - \frac{1}{2} \right|.$$

Now we need to see that these two advantages are equivalent, so that we can interchange them without worrying. The proof follows the one in [BS20].

Lemma 4.7 (Theorem 2.10 [BS20])**.** *For every encryption scheme $\mathcal{S}$ and adversary $\mathcal{A}$ we get that*

$$\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}] = 2 \cdot \mathrm{SSAdv}^*[\mathcal{A}, \mathcal{S}].$$

Proof. Let $\mathcal{S}$ be an encryption scheme and $\mathcal{A}$ an adversary. Then,

$$\begin{aligned} \Pr(\hat{b} = b) &= \Pr(\hat{b} = 0 | b = 0)\Pr(b = 0) + \Pr(\hat{b} = 1 | b = 1)\Pr(b = 1) = \\ &= \frac{1}{2}(\Pr(\hat{b} = 0 | b = 0) + \Pr(\hat{b} = 1 | b = 1)) = \\ &= \frac{1}{2}(1 - \Pr(\hat{b} = 1 | b = 0) + \Pr(\hat{b} = 1 | b = 1)). \end{aligned}$$

Therefore we get that

$$\begin{aligned} \mathrm{SSAdv}[\mathcal{A}, \mathcal{S}] = |\Pr(W_0) - \Pr(W_1)| &= |\Pr(\hat{b} = 1 | b = 0) - \Pr(\hat{b} = 1 | b = 1)| = \\ &= |2\Pr(\hat{b} = b) - 1| = 2 \cdot \mathrm{SSAdv}^*[\mathcal{A}, \mathcal{S}] \end{aligned}$$

as we wanted to see. □

This means that whenever looking at semantic security, it is indifferent if we look at it as the probability the adversary distinguishing the ciphertexts being negligible or as the probability of the adversary guessing being negligibly close to $\frac{1}{2}$, since 2 times a negligible value remains negligible.

4.1.3 CPA Security

Semantic security is a very useful notion of security to make sure no information is leaked when encrypting a message. But there is a catch, it only contemplates encrypting one message with a given key. What happens if you want to use the same key repeatedly to encrypt different messages? Is your encryption scheme still secure?

This is what Chosen Plaintext Attack (CPA) security was defined for. In broad strokes, it is a very similar concept to semantic security in that you cannot distinguish between encryptions of chosen plaintexts, but now you can make several queries to the adversary (a polynomial amount to be specific). Let us define its attack game and advantage.

Attack Game 4.8 (Attack Game 5.2 [BS20])**.** Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Given an adversary $\mathcal{A}$, the *Chosen Plaintext Attack (CPA) attack game*, after the challenger computes $b \xleftarrow{\$} \{0,1\}$, follows with the following experiment b:

- The challenger chooses $e \xleftarrow{\$} \mathcal{K}_p$ (and sends it to the adversary if we are in public key encryption).
- The adversary submits polynomially many queries to the challenger. For $i = 1, 2, ...,$ $\mathcal{A}$ submits two same-length messages $m_{0_i}, m_{1_i} \in \mathcal{M}$. The challenger computes $c_i = E_e(m_{b_i})$ and sends it to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0,1\}$.

Let W_b be the event in which $\mathcal{A}$ outputs 1 in the event b, then we define $\mathcal{A}$'s *CPA advantage* as:

$$\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}] := |\Pr(W_0) - \Pr(W_1)|.$$

Definition 4.9 (Definition 5.2 [BS20])**.** An encryption scheme $\mathcal{S}$ is said to be *CPA secure* if for all efficient adversaries $\mathcal{A}$, the value $\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}]$ is negligible.

Note that in order to achieve CPA security, an encryption scheme $\mathcal{S}$ cannot have a deterministic encryption function, even in symmetric cryptography where the adversary does not have the key. Otherwise, CPA security is easily broken by sending two queries and just changing one of the messages.

Furthermore, now it is even more clear that if an encryption scheme $\mathcal{S}$ is CPA secure, then it is semantically secure. This is due to the fact that the semantic security attack game (Attack Game 4.3) is exactly the same that the CPA attack game (Attack Game 4.8) making only one query, and the advantages are computed in the same way. Therefore we will get

$$\max_{\mathcal{A}} \{\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}]\} \geq \max_{\mathcal{A}} \{\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]\}$$

so if $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ in non-negligible, then $\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}]$ will be too, thus seeing what we wanted.

Moreover, in the public-key encryption realm (which is where our encryption scheme stands) we have that if an encryption scheme $\mathcal{S}$ is semantically secure then it is also CPA secure. The basic idea on which this is based is the fact that given that the adversary knows the public key, the adversary can compute as many ciphertexts as wanted, so extending the semantic security attack game onto the CPA attack game should not vastly change the information the adversary knows. The proof is taken from [BS20].

Theorem 4.10 (Theorem 11.1 [BS20]). *Let $\mathcal{S}$ be a public-key encryption scheme. Then if $\mathcal{S}$ is semantically secure, $\mathcal{S}$ is CPA secure. In particular, for every CPA adversary $\mathcal{A}$ that plays Attack Game 4.8 with respect to $\mathcal{S}$, and which makes at most Q queries to its challenger, there exists a semantic security adversary $\mathcal{B}$ such that*

$$\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}] = Q \cdot \mathrm{SSAdv}[\mathcal{B}, \mathcal{S}].$$

Proof. Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $\mathcal{A}$ be the adversary that plays the CPA Attack Game 4.8 with respect to $\mathcal{S}$ making at most Q queries to its challenger. To prove what we want to see we will need to define another type of experiments for Attack Game 4.8, the j Hybrid experiments played against the same CPA challenger. Let us see them:

Hybrid experiment j: For $j \in \{0, \dots, Q\}$ we define the following experiment:

- The challenger computes the keys (e, d) with the key generation protocol and forwards the public key e to $\mathcal{A}$.
- For every query $m_{i0}, m_{i1},\ i \in \{1, \dots, Q\}$ forwarded by the adversary, the challenger computes the following:
  - **if i $\leq$ j:** $c_i = E_e(m_{i1})$
  - **if i $>$ j:** $c_i = E_e(m_{i0})$

  and forwards c_i to $\mathcal{A}$.
- $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.

First note that both Experiment 0 and Experiment 1 used to define CPA advantage in Attack Game 4.8 are a particular case of these Hybrid experiments, in particular Experiment 0 is Hybrid experiment 0 and Experiment 1 is Hybrid experiment Q. We will also define (to ease notation) the probability that $\mathcal{A}$ outputs 1 in Hybrid experiment j as p_j, which exists and is "fair game", since even though CPA security is defined using only Experiment 1 and Experiment 0 the rest of Hybrid experiments can be run for any adversary attacking the CPA security of the scheme. Now if we define $W_{\mathcal{A}j}$ the event which $\mathcal{A}$ outputs 1 in experiment j of the hybrid Attack Game we get that:

$$\mathrm{CPAAdv}[\mathcal{A}, \mathcal{S}] = |\Pr(W_{\mathcal{A}Q}) - \Pr(W_{\mathcal{A}0})| = |p_Q - p_0|.$$

Now we need to define an efficient adversary $\mathcal{B}$ that attacks the semantic security game (Attack Game 4.3) acting as a challenger towards $\mathcal{A}$.

Adversary $\mathcal{B}$: The adversary $\mathcal{B}$ against Attack Game 4.3 works as follows:

- Receives e from its challenger, forwards it to $\mathcal{A}$ and chooses $w \xleftarrow{\$} \{1, \dots, Q\}$.
- For every query $m_{i0}, m_{i1},\ i \in \{1, \dots, Q\}$ received from $\mathcal{A}$ computes the following:
  - **if i $<$ w:** $c_i = E_e(m_i 1)$
  - **if i $=$ w:** Send m_{i0}, m_{i1} to the challenger and receive c_i.
  - **if i $>$ w:** $c_i = E_e(m_{i0})$

  and forwards c_i to $\mathcal{A}$.
- $\mathcal{B}$ outputs $\hat{b}$ received from $\mathcal{A}$.

Now let us compute the semantic security advantage of $\mathcal{B}$. Let $W_{\mathcal{B}b}$ the event which $\mathcal{B}$ outputs 1 in experiment b. Then we have:

$$\mathrm{SSAdv}[\mathcal{B},\mathcal{S}] = |\Pr(W_{\mathcal{B}1}) - \Pr(W_{\mathcal{B}0})| = \left|\sum_{j=1}^{Q} \Pr(w=j)\cdot(\Pr(W_{\mathcal{B}1}|w=j) - \Pr(W_{\mathcal{B}0}|w=j))\right| =$$

$$= \left|\frac{1}{Q}\sum_{j=1}^{Q} \Pr(W_{\mathcal{A}j-1}) - \Pr(W_{\mathcal{A}j})\right| = \left|\frac{1}{Q}\sum_{j=1}^{Q} p_{j-1} - p_j\right| = \frac{1}{Q}|p_0 - p_Q|.$$

So we get, as we wanted to see, that

$$\mathrm{CPAAdv}[\mathcal{A},\mathcal{S}] = Q\cdot\mathrm{SSAdv}[\mathcal{B},\mathcal{S}].$$

□

4.2 Standard Proofs of Security

Now that we have properly defined what security is, we need to solve another problem. How do we mathematically prove that the advantages are negligible? It is clear that the probabilities on which these advantages are based upon are usually not easily computed, so we need to find other indirect ways to ensure security. Here is where we discuss the fact that there are two main currents in dealing with this issue: heuristic security and mathematical security.

On the one hand we have *heuristic security*. It is a security model that in essence asserts that any cryptographic protocol is secure as long as there is no known efficient attack against it. It is a very useful model since if one desires to effectively "break" the cryptographic protocol, then one will need an efficient attack against it, so as long none is known it is not unreasonable to think that the protocol is secure. Furthermore, since you only need to make sure that no efficient attack is known, the protocols with security based in this model are usually more efficient than the ones with mathematical security. However, there are drawbacks, since the only way to break security for the protocol is finding an attack to it in particular, so protocols based on heuristic security are rendered obsolete much more often that the ones based on mathematical security. A couple of examples of heuristic security are the Advanced Encryption Standard (AES), the NIST symmetric encryption standard, and Secure Hash Algorithms (SHA), the NIST hash algorithm standards; the first of which remains secure up to this day (since 2001), while the second has seen several iterations (SHA-0 to SHA-3) since they have been broken several times.

On the other hand we have *mathematical security*. The idea behind this security notion is to be able to mathematically prove a given cryptographic protocol to be secure. To do so, the usual method is to use security reductions to the protocol from a well-studied problem that is known (or assumed) to be hard to solve. The gist of these security reductions is to mathematically prove that if an adversary had a non-negligible advantage against the cryptographic protocol, then it would be able to solve the hard problem. This implies, since the problem is known (or assumed) to be hard, that breaking the protocol is hard. This is a more stable sense of security, since the problems where security is based are usually very well studied and documented, so the discovery of any algorithm efficiently solving them is usually unlikely.

Note that mathematical security implies heuristic security, since if an efficient attack is known against the cryptographic protocol, then through the security reduction the hard problem would be efficiently solved, giving a contradiction with it being hard. In this master's thesis we will prove mathematical security of three things: the cryptosystem, the threshold decryption protocol and the key generation protocol, the first of which we will tackle in this subsection, and the other two on the next.

4.2.1 Reducing Security of Encryption Scheme to R-LWE

We will split the proof of mathematical security of our cryptosystem in two distinct parts: reducing the security of the cryptosystem to the decisional R-LWE problem, and then reducing the decisional R-LWE problem to K-DGS, a well-known lattice problem assumed to be hard to solve. We will make this splitting because the first reduction will be for any distribution χ, while the second reduction will be specifically for the distribution $\overline{\Psi}^n$. The first reduction will follow the ideas from the reduction of Regev's encryption scheme to LWE given in [Reg09].

Theorem 4.11. *Given χ a distribution over R_q, there exists a reduction to the security of the Encryption Scheme 2.23 from the decisional R-LWE$_\chi$ problem.*

Proof. What we want to see is that given an efficient adversary $\mathcal{A}$ who has non-negligible semantic security advantage, we can construct an efficient adversary $\mathcal{B}$ with access to $\mathcal{A}$ who given an instance of the decisional R-LWE problem, it can solve it with probability non-negligibly bigger than $\frac{1}{2}$.

Let $(\overline{\mathbf{a}}_i, \overline{\mathbf{b}}_i) \in R_q \times R_q$ be an instance of the decisional R-LWE problem. What we need $\mathcal{B}$ to do is to be able to output whether a polynomial amount of instances are samples of the distribution $A_{\mathbf{s},\chi}$ or of the uniform distribution over $R_q \times R_q$, in other words, we want to know whether $\overline{\mathbf{b}}_i = \overline{\mathbf{a}}_i \cdot \overline{\mathbf{s}} + \overline{\mathbf{e}}$ for some $\overline{\mathbf{s}} \in R_q$ and $\overline{\mathbf{e}} \longleftarrow \chi$.

Note that any adversary $\mathcal{A}$ who breaks semantic security may be of one of two types. Either $\mathcal{A}$ has non-negligible semantic security advantage against the encryption scheme when $(\mathbf{a_E}, \mathbf{b_E})$ are generated independently uniformly at random (instead of having $\mathbf{b_E} = \mathbf{a_E} \cdot \mathbf{s} + \mathbf{e}$) or it does not. We will construct two different adversaries for these cases.

Assume firstly that $\mathcal{A}$ has a negligible semantic security advantage against the encryption scheme when $(\mathbf{a_E}, \mathbf{b_E})$ are generated independently uniformly at random. Let $(\mathbf{a}_1, \mathbf{b}_1)$ be an instance of the R-LWE$_\chi$ problem, then we define the following attack game.

Attack Game 1: The attack game goes as follows:

- Set the public key to $(\overline{\mathbf{a}}_1, \overline{\mathbf{b}}_1)$ and send it to $\mathcal{A}$.
- Receive $\mathbf{m}_{01}, \mathbf{m}_{11}$ from the adversary, and choose $b_1 \xleftarrow{\$} \{0, 1\}$.
- Compute $\mathbf{u}_1 = \overline{\mathbf{a}}_1 \cdot \mathbf{r_E} + \mathbf{e_u}$ and $\mathbf{v}_1 = \overline{\mathbf{b}}_1 \cdot \mathbf{r_E} + \mathbf{e_v} + \mathbf{m}_{b_1} \lfloor \frac{q}{2} \rfloor$ with $\mathbf{r_E}, \mathbf{e_u}, \mathbf{e_v} \longleftarrow \chi$, and send $(\mathbf{u}_1, \mathbf{v}_1)$ to $\mathcal{A}$.
- Receive $\hat{b}_1$ from the adversary.

Then $\mathcal{B}$ will work as follows. When given the instances, it picks $(\overline{\mathbf{a}}_1, \overline{\mathbf{b}}_1)$ and performs the Attack Game 1 with $\mathcal{A}$ a polynomial amount of times. Then it computes the advantage:

$$\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}] = \left| \frac{\text{Number of queries where } \hat{b}_1 = b_1}{\text{Total number of queries}} - \frac{1}{2} \right|.$$

We know from how we have defined the adversary $\mathcal{A}$, since $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ is non-negligible, if the instances follow the distribution $A_{\mathbf{s},\chi}$ then $\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}]$ will be non-negligible and if the instances are uniform over $R_q \times R_q$ then $\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}]$ will be negligible. This means that with non-negligible probability $\mathcal{B}$ can solve the decisional R-LWE$_\chi$ problem as we wanted.

Assume now that $\mathcal{A}$ has a non-negligible semantic security advantage against the encryption scheme when $(\mathbf{a_E}, \mathbf{b_E})$ are generated independently uniformly at random. Let, once again, $(\mathbf{a}_1, \mathbf{b}_1)$ and $(\mathbf{a}_2, \mathbf{b}_2)$ be two instances of the R-LWE$_\chi$ problem, then we define the following attack game.

Attack Game 2: The attack game goes as follows:

- Set the public key to $(\bar{\mathbf{a}}_1, \bar{\mathbf{a}}_2)$ and send it to the adversary.
- Receive $\mathbf{m}_{02}, \mathbf{m}_{12}$ from the adversary, and choose $b_2 \xleftarrow{\$} \{0,1\}$.
- $\mathbf{u}_2 = \bar{\mathbf{b}}_1$ and $\mathbf{v}_2 = \bar{\mathbf{b}}_2 + \mathbf{m}_{b_2 2}\lfloor \frac{q}{2} \rfloor$ and send $(\mathbf{u}_1, \mathbf{v}_1)$ to $\mathcal{A}$.
- Receive $\hat{b}_2$ from the adversary.

Then $\mathcal{B}$ will work as follows. When given the instances, it picks two of them $(\bar{\mathbf{a}}_1, \bar{\mathbf{b}}_1)$ and $(\bar{\mathbf{a}}_2, \bar{\mathbf{b}}_2)$, and performs the Attack Game 2 with $\mathcal{A}$ a polynomial amount of times. Then it computes the advantage:

$$\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}] = \left| \frac{\text{Number of queries where } \hat{b}_2 = b_2}{\text{Total number of queries}} - \frac{1}{2} \right|.$$

We know from how we have defined the adversary $\mathcal{A}$, since $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ is non-negligible, if the instances follow the distribution $A_{\mathbf{s},\chi}$ then $\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}]$ will be non-negligible and if the instances are uniform over $R_q \times R_q$ then $\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}]$ will be negligible, since if $\bar{\mathbf{b}}_i$ are uniformly at random then $\mathbf{v}_2$ is independent from the public key and $\mathbf{m}_{b_2 2}$. This means that with non-negligible probability $\mathcal{B}$ can solve the decisional R-LWE$_\chi$ problem as we wanted, since it is possible to distinguish a non-negligible random variable from a negligible random variable with a non-negligible probability. □

Remark 4.12. Note that the proof of Theorem 4.11 is not a constructive proof, in the sense that it does not give an efficient adversary $\mathcal{B}$ that for any adversary $\mathcal{A}$ with non-negligible advantage against the encryption scheme, solves the decisional R-LWE$_\chi$ problem. However, we have proven that for any such $\mathcal{A}$ exists an efficient $\mathcal{B}$ solving the decisional R-LWE$_\chi$ problem, and this is enough for security. Since we are postulating that the decisional R-LWE$_\chi$ problem is hard to solve, if an efficient $\mathcal{A}$ existed we could efficiently solve the hard problem.

4.2.2 Reducing R-LWE to K-DGS

We now need to see that solving the decision R-LWE$_{\overline{\Psi}}$ is a hard problem to solve, in our case as hard to solve as the Discrete Gaussian Sampling over K (K-DGS), where K is the field such that R is its ring of integers, in other words, $R = \mathcal{O}_K$. Thankfully, this job has already been done in [PRSD17], though to do so properly we need to give some clarifications about different ways to define the R-LWE distribution.

Let K be a number field with R its ring of integers. Let $R^\vee$ be the fractional codifferential ideal of K ($R^\vee = \{x \in K \mid \mathrm{Tr}(xR) \subset \mathbb{Z}\}$), and let $\mathbb{T}^R = K_\mathbb{R}/R^\vee$. Let $q \geq 2$ be an integer modulus. Let us unpack this. Firstly in our specific case of K being a cyclotomic field with $n = 2^k$ for some case, we have $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$, so in turn it can be seen that $R^\vee$ is isomorphic to R. Secondly, $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$ which is isomorphic to $\mathbb{R}^n$, so looking it component by component $\mathbb{T}^R$ could be seen as isomorphic to $\mathbb{T}^n$ with $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. With this out of the way we can see their definition.

Definition 4.13 (Definition 2.14 [PRSD17])**.** For $s \in R_q^\vee$ and an error distribution ψ over $K_\mathbb{R}$, the R-LWE distribution $A_{s,\psi}$ over $R_q \times \mathbb{T}^R$ is sampled by independently choosing $a \xleftarrow{\$} R_q$ and an error term $e \leftarrow \psi$, and outputting $(a, b = (a \cdot s)/q + e \mod R^\vee)$.

Now our postulate is that this definition taking as Ψ an n-dimensional spherical continuous Gaussian with parameter ξ (which is a distribution used in [PRSD17]) and then raising it to R_q again, is equivalent to our Definition 2.20 using $\overline{\Psi}_{\frac{\xi}{q}}$. It can be seen as one, since a spherical Gaussian in $\mathbb{R}^n$ can be seen as the product of n independent Gaussians over $\mathbb{R}$ with the same standard deviation. Then in essence what we are doing in Definition 4.13 is multiply a times s, then divide the result by q (which we can since we are seeing the elements in $K_{\mathbb{R}}$ which is a field) and adding the error distribution. Then we reduce it modulo $R^\vee$ thus landing in $\mathbb{T}^R$. Now if we look it component by component we have in essence computed $a \cdot s/q$ and then added to each component a sample of $\Psi_{\frac{\xi}{q}}$, so when raising it again to $R_q^\vee$ (by multiplying by q and rounding) we get that $q(a \cdot s/q) = a \cdot s \in R_q^\vee$ and to every component we have added an independent sample taken from $\overline{\Psi}_{\frac{\xi}{q}}$. Therefore, if ρ_ξ^n is the spherical Gaussian with parameter ξ, given an adversary who solves $R\text{-LWE}_{\overline{\Psi}_{\frac{\xi}{q}}}$ it is easy to give an adversary who solves $R\text{-LWE}_{\rho_\xi^n}$.

Therefore we can apply the following result from [PRSD17]. Let ρ_ξ^n be a spherical Gaussian with parameter ξ and $\rho_{\mathbf{r}}^n$ be an elliptical Gaussian defined as the product of n 1-dimensional Gaussians such that the standard deviation $\sigma_i \leq r_i$ with $\mathbf{r} = (r_1, \dots, r_n)$.

Lemma 4.14 (Corollary 7.3 [PRSD17])**.** *There is a polynomial-time quantum reduction from $K\text{-DGS}_\gamma$ to the (average-case, decision) problem of solving $R\text{-LWE}_{\rho_\xi^n}$ using l samples with $\xi = \alpha \left(\frac{nl}{\log(nl)}\right)^{\frac{1}{4}}$, $\alpha > 0$ and*

$$\gamma(\mathcal{I}) = \max\left\{\eta(\mathcal{I}) \cdot \frac{\sqrt{2}}{\alpha} \cdot \omega\left(\sqrt{\log(n)}\right), \quad \frac{\sqrt{2n}}{\lambda_1(\mathcal{I}^\vee)}\right\}$$

as long as $\alpha q \geq \omega\left(\sqrt{\log(n)}\right)$.

Therefore, using the "translation" we have stated before, we can say that solving the decisional $R\text{-LWE}_{\overline{\Psi}_{\frac{\xi}{q}}}$ is as hard as solving $K\text{-DGS}_\gamma$, a lattice based problem which is believed to be hard to solve. This means, adding the reduction from security of the encryption scheme to R-LWE, that breaking our cryptosystem is a hard problem to solve.

Remark 4.15. We would also like to note that this reduction would only be directly appliable to the decryption protocol when we can assure that both $\mathbf{s}$ and $\mathbf{e}$ follow the distribution $\overline{\Psi}$. To be able to apply it to our case when the keys are generated through Protocol 2.25 we would need to do a deeper level proof, which we do prioritize for this Master's Degree Thesis. However, we can say that it is folklore that all currently known ways to attack R-LWE cannot distinguish the distribution of errors (some current implementation use the uniform distribution), so we can say that our encryption scheme is secure against currently known attacks.

4.3 Non-leakage of Information in Threshold Schemes

In the previous subsection we have seen that breaking the security of Encryption Scheme 2.23 is as hard as solving the hard lattice problem $K\text{-DGS}_\gamma$. However, the attack games considered are always an adversary against a unique entity known as the challenger. This means that we cannot directly apply this security result, since both Protocol 2.24 and Protocol 2.25 are distributed, meaning that the actions usually performed by the unique challenger are performed by several players.

Therefore, we need to see that the adversary does not gain any extra information by interacting with the distributed protocol, so the previous security analysis still holds true. We will start first with the Protocol 2.24 (threshold encryption protocol), seeing that an adversary $\mathcal{A}$ cannot distinguish between interacting with the protocol or with random inputs. Furthermore, we will also give the adversary the ability to choose its shares of the secret key and the PRSS keys, since it makes the game easier and it only serves to see that the protocol's security is even stronger than what is usually required. We will give a revamped proof to a theorem on [AE20].

Theorem 4.16 (Theorem 5.9 [AE20])**.** *Assuming that $\Phi^R_\cdot(\cdot)$ is a secure pseudo-random function modeled as a random oracle, that the keys K_H have been securely generated and distributed and that the secret key s has been securely generated and shared, the Decryption Protocol is secure against a passive and static adversary, corrupting up to $t = u - 1$ players.*

Proof. We want to construct an Attack Game in which the adversary cannot distinguish between the protocol executed correctly or with random values to show that the distribution does not leak anything about the secret key nor the error e.

Let B denote the set of corrupted players and C the set of honest players. The Attack Game works as follows. Assume that the challenger knows the secret key s and the K_H such that $B \supseteq H$ (the keys that the adversary does not know) which have been securely generated. Assume that the challenger sends to the adversary $\mathcal{A}$ the ciphertext (u, v) and then $\mathcal{A}$ submits (s'_B, K_{H_B}, d'_B) as the challenge, where s'_B are the shares on the secret key of the corrupted players, K_{H_B} are the keys K_H such that $B \not\supseteq H$ (the keys $\mathcal{A}$ knows) chosen by $\mathcal{A}$, and d'_B are the shares on the decryption of the corrupted players. Then the challenger generates consistent shares on s for the players not in B.

Once all these preliminaries are done, the challenger chooses $b \xleftarrow{\$} \{0, 1\}$ and proceeds as following:

- **If b = 0:** The challenger uses the decryption protocol to compute the shares of the decryption d'_C for the honest players. It computes the decrypted message m and outputs (d'_C, m).
- **If b = 1:** The challenger computes for every H such that $B \supseteq H$ some element (taking $k = c\xi$)

$$r_H \in \left[-\frac{k^2}{2}(2n+1)q^d,\ \frac{k^2}{2}(2n+1)q^d\right]^n$$

  uniformly at random and we denote as y the polynomial in R_q with vector of coefficients $\sum_{B \not\supseteq H} \Phi^R_{K_H}(c) + \sum_{B \supseteq H} r_H$. Then the challenger generates d'_C consistent shares of $y + m\lfloor \frac{q}{2} \rfloor$ (the challenger knows m since it can be computed using the protocol, since everything needed is known) and outputs (d'_C, m).

Finally $\mathcal{A}$ outputs $\tilde{b} \in \{0, 1\}$, meaning whether it thinks it has interacted with the protocol or with a simulation, and the Game concludes.

It is clear that m will be correct in both cases given the proof of correctness of the protocol (Theorem 3.16), and furthermore, $y + \lfloor \frac{q}{2} \rfloor \cdot m$ will be an effective "decryption" of m in the sense that every coefficient will be closer to 0 if $m_i = 0$ and closer to $\lfloor \frac{q}{2} \rfloor$ if $m_i = 1$, also by direct consequence of the proof of correctness (what we are adding to $\lfloor \frac{q}{2} \rfloor \cdot m$ in the worst case scenario is smaller than $\frac{q}{4}$).

Therefore we only need to see that d'_C are indistinguishable whether they are computed with $b = 0$ or with $b = 1$. Let us see it. First y and x are computationally indistinguishable to the adversary given the properties of pseudo-randomness of $\Phi^R_\cdot(\cdot)$. We now want to see that y and $y + e \cdot r_E + e_v - s \cdot e_u$ are

computationally indistinguishable. It is clear that y_i is the sum of at least one element taken uniformly at random from an interval of size $2(c\xi)^2(2n+1)q^d$ and it is known that $e_i, (r_E)_i, (e_v)_i, s_i$ and $(e_u)_i$ are distributed with overwhelming probability in an interval of size $2(c\xi)$, so $(e \cdot r_E + e_v - s \cdot e_u)_i$ is distributed in an interval smaller than $2(2n+1)(c\xi)^2$ because of the product of polynomials being done through the anticyclic matrix (using the same reasoning as in the proof of correctness). We also know that $2(c\xi)^2(2n+1)$ is exponentially smaller than $2(c\xi)^2(2n+1)q^d = 2(c\xi)^2(2n+1)2^{d\Theta(\lambda)}$, so the way y and $y + e \cdot r_E + e_v - s \cdot e_u$ are distributed are statistically indistinguishable. Therefore $y + m\lfloor \frac{q}{2} \rfloor$ and $x + e \cdot r_E + e_v - s \cdot e_u + m\lfloor \frac{q}{2} \rfloor$ are computationally indistinguishable.

Finally, adding it all together we get that the output (d'_C, m) is computationally indistinguishable whether it has been computed with $b = 0$ or with $b = 1$, so

$$\left| \Pr\left(\tilde{b} = b\right) - \frac{1}{2} \right| = \text{neg}(\lambda)$$

as we wanted to see. □

Now after Theorem 4.16 we have only seen that Protocol 2.24 is secure when the keys are securely generated and against a passive adversary corrupting $t \leq u - 1$ players, but it is standard to see that the same protocol is secure against a semi-honest adversary corrupting $t < \frac{u}{2}$ players and against an active adversary corrupting $t < \frac{u}{3}$ players if instead of the client choosing $t+1$ players he sends the ciphertext to all players and all combinations of $t+1$ decrypt it, giving a majority of correct outputs.

The reason behind this is that we have already seen that no information is leaked, so the only thing required is to see that the adversary cannot abort the protocol or cause an incorrect output. In case of a semi-honest adversary (who can only cause players to abort), it is clear to see that if we have more honest players than corrupt the protocol as a whole will not abort so $t < \frac{u}{2}$ is enough. In case of an active adversary (who can cause players to deviate arbitrarily from the protocol), what is needed is that if all combinations of $t+1$ players decrypting the message, there needs to be a majority of combinations of $t+1$ players with no corrupt players. This gives us that $t < \frac{u}{3}$ is enough.

However, we still need to see that Protocol 2.25 correctly and securely generates the keys used in Protocol 2.24. We will first discuss the correctness of the key generation, which can be seen with not much difficulty. For the elements chosen uniformly at random it is obvious that the sum of elements chosen uniformly at random in $\mathbb{Z}_q$ generates elements uniformly distributed in $\mathbb{Z}_q$. For elements computed as a sum of rounded discrete Gaussians just a single honest player is needed to be able to rely on the hardness of R-LWE. As long as $s_j, e_j \sim \overline{\Psi}$, then

$$a \cdot \sum_i s_i + \sum_i e_i = a \cdot s_j + e_j + a \sum_{i \neq j} s_i + \sum_{i \neq j} e_i$$

will be indistinguishable from random since $a \cdot s_j + e_j$ is.

We know that the distribution Ψ_σ is the distribution in $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ obtained by sampling a Gaussian random variable X, $X \sim N(0, \sigma)$ and then taking mod 1 by only keeping the decimal part, an operation that behaves well with addition. Therefore,we can assure that

$$\sum_{i=1}^{u} \overline{\Psi}_{\frac{\xi}{q\sqrt{u}}} \leq \overline{\Psi}_{\frac{\xi}{q}} + u.$$

using how rounding to the integers works and the fact that the sum of continuous Gaussians is a continuous Gaussian. However we cannot say that the sum of rounded discrete Gaussians is a rounded discrete Gaussian.

This means that we will need to slightly change the proof of Theorem 3.16 if we want it to hold when the key generation is done through Protocol 2.25 against an active adversary. However, this can easily be done by using Lemma 3.13, which will only add a factor u in the proof, which does not influence the asymptotic analysis.

Now we only need to see that 2.25 is secure. What we will see is that no information about the secret key, the errors or the keys K_H is leaked when distributing, so we need to see that the adversary cannot distinguish between interacting with the protocol (where nobody sets any value) or a simulation where the challenger chooses each value generated by the protocol (s, e, a_E, K_H).

In this case we will prove directly security against an active adversary corrupting up to $t < \frac{u}{3}$ players since, unlike with Theorem 4.16, the naive implementation that is secure against a passive adversary could not be extended to an active adversary. We will once again give a revamped proof to a theorem in [AE20].

Theorem 4.17 (Theorem 5.14 [AE20]). *Let $c = \Omega(\sqrt{n})$ and $0 < d < 1$, we define $\mathbb{I}_{KG} = [-\frac{c\xi}{\sqrt{u}}q^d, \frac{c\xi}{\sqrt{u}}q^d]$. Assuming that the image interval of the pseudo-random function $\Phi_{\cdot}^{KG}(\cdot)$ is $\mathbb{I}_{KG}^n$, then the Key Generation Protocol is secure against an active and static adversary, corrupting up to $t < \frac{u}{3}$ of the players.*

Proof. We want to construct an Attack Game in which the adversary cannot distinguish between the protocol executed correctly or someone (the challenger) setting previous values for the parameters.

Let B denote the set of corrupt players and C the set of honest players. The Attack Game works as follows. Assume the adversary $\mathcal{A}$ submits $(s'_B, e'_B, K'_{H_B}, a'_{E_B})$ shares of the corrupt players of s, e, K_H and a_E respectively as the challenge (assuming s'_B and e'_B have gone through NIVSS and are therefore of the correct size). Then the challenger chooses $b \xleftarrow{\$} \{0, 1\}$ uniformly at random and proceeds as follows:

- **If b = 0:** The challenger uses the key generation protocol to generate a_E, b_E and the shares s'_C, e'_C, K'_{H_C} and a'_{E_C} and outputs $(a_E, b_E, s'_C, e'_C, K'_{H_C}, a'_{E_C})$.
- **If b = 1:** The challenger chooses $s, e \sim \overline{\Psi}_{\frac{\xi}{q}}$, $a_E \in R_q$ uniformly at random and every $K_H \in \mathbb{Z}_q$ uniformly at random. Then for s and e it computes consistent Shamir shares s'_C, e'_C and simulates running the NIVSS and the conversion from additive shares to Shamir shares for every coefficient. For K_H and a_E it computes consistent Shamir shares K'_{H_C}, a'_{E_C}. Then using all the shares the challenger computes a_E and b_E and outputs $(a_E, b_E, s'_C, e'_C, K'_{H_C}, a'_{E_C})$.

Finally $\mathcal{A}$ outputs $\tilde{b} \in \{0, 1\}$, meaning whether it thinks it has interacted with the protocol or with a simulation, and the Game concludes.

Since every step is done exactly in the same order and are from indistinguishable distributions, it is easy to see that $(a_E, b_E, s'_C, e'_C, K'_{H_C}, a'_{E_C})$ are indistinguishable to the adversary whether they have been computed with $b = 0$ or $b = 1$, and since the noise in the NIVSS is exponentially larger that what we want to cover (as we have seen in the proof of Theorem 4.16) no information is leaked so

$$\left| \Pr\left(\tilde{b} = b\right) - \frac{1}{2} \right| = \text{neg}(\lambda)$$

as we wanted to see. □

Remark 4.18. Due to how we have implemented the PRSS share (explanation more in detail in section 5.2.2) mixing additive and Shamir secret sharing, the proof of Theorem 4.17 is a little sloppy in the exact ways to simulate the change from additive to Shamir shares. However, as we will also comment in section 5.2.2, we are planning as future work to change this implementation to one that deals purely with Shamir shares, causing this last proof to clean up considerably.

Finally, combining all the security theorems and reductions in this chapter we can give the Main Theorem of our encryption scheme.

Theorem 4.19 (Theorem 5.16 [AE20]). *Let* $c = \Omega(\sqrt{\lambda})$, $0 < d < 1$, $1 < \xi < \frac{1}{c}\sqrt{\frac{q}{4(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right)}}$, $\mathbb{I}^n_{KG}$ *defined as in Theorem 4.17 be the interval image of the pseudo-random function* $\Phi^{KG}_{\cdot}(\cdot)$ *and* $\mathbb{I}^n = [-(c\xi)^2(n\sqrt{u}q^d+1),(c\xi)^2(n\sqrt{u}q^d+1)]^n$ *be the interval image of the pseudo-random function* $\Phi_{\cdot}(\cdot)$. *If* $K\text{-}DGS_\gamma$ *is hard, then encryption under* $\mathbf{s}$ *generated by the Key Generation Protocol is CPA secure against any active and static polynomial time adversary corrupting* $t < \frac{u}{3}$ *players acting through both protocols and decryption under* $\mathbf{s}$ *is correct except with negligible probability.*

Proof. Security is direct from all the security results and observations in this chapter. We only need to prove correctness against an active adversary.

We remember from the proof of Theorem 3.16 that what we want to see is that the $\Pr((|\mathbf{x}+\hat{\mathbf{e}}|)_i > \frac{q}{4})$ is negligible for all i. We also remember from that proof that

$$(|\hat{\mathbf{e}}|)_i \leq |e_i \cdot r_{E_1}| + |e_{i-1} \cdot r_{E_2}| + \ldots + |e_{i+2} \cdot r_{E_{n-1}}| + |e_{i+1} \cdot r_{E_n}| + |e_{v_i}| + |s_i \cdot e_{u_1}| + \ldots + |s_{i+1} \cdot e_{u_n}|.$$

Here is where the differences start. On Theorem 3.16 we could assure that $\mathbf{s}, \mathbf{e} \sim \overline{\Psi}^n_{\frac{\xi}{q}}$. Now, given that the keys have been generated through Protocol 2.25, we can only say that $\mathbf{s} = \mathbf{s}_h + \mathbf{s}_c$ where the contributions of the honest players follow a the distribution $\overline{\Psi}_{\frac{\xi}{q\sqrt{u}}}$, and for the contributions of the corrupt players we can only assure that $\|\mathbf{s}^j_c\|_\infty \leq \frac{|\mathbb{I}_{KG}|}{2} = \frac{c\xi}{\sqrt{u}}q^d$. The analysis for $\mathbf{e}$ is the same as for $\mathbf{s}$.

To ease the proof we will take a very rough upper bound and assume all contributions to both $\mathbf{s}$ and $\mathbf{e}$ are made by corrupt players, so we have noting $k := c\xi$, $\|\mathbf{s}_i\|_\infty = \|\mathbf{e}_i\|_\infty \leq \sqrt{u}kq^d$. Then by applying both Lemma 3.14 and Lemma 3.11 we get that:

$$\Pr(|e_i \cdot r_{E_j}| > \sqrt{u}k^2q^d) = \Pr(|s_i \cdot e_{u_j}| > \sqrt{u}k^2q^d) < 2\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\lceil c\xi\rceil - \frac{1}{2}}{\sqrt{2}\xi}\right)^2}$$

which in turn gives us

$$\Pr((|\hat{\mathbf{e}}|)_i > k(2n\sqrt{u}kq^d+1)) < (2n+1)2\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\lceil c\xi\rceil - \frac{1}{2}}{\sqrt{2}\xi}\right)^2}.$$

We also know that by construction

$$(|\mathbf{x}|)_i \leq \binom{u}{t}k^2(n\sqrt{u}q^d+1). \tag{5}$$

And by hypothesis:

$$\xi < \frac{1}{c}\sqrt{\frac{q}{4(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right)}} \Rightarrow k^2 < \frac{q}{4(\sqrt{u}nq^d+1)(\binom{u}{t}+1)} \Rightarrow k^2(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right) < \frac{q}{4}. \tag{6}$$

Therefore giving us:

$$\begin{aligned}
\Pr\left((|\mathbf{x}+\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) &\leq \Pr\left((|\mathbf{x}|)_i + (|\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) < \\
&< \Pr\left((|\mathbf{x}|)_i + (|\hat{\mathbf{e}}|)_i > k^2(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right)\right) = \\
&= \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right) - (|\mathbf{x}|)_i\right) \leq \\
&\leq \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(\sqrt{u}nq^d+1)\left(\binom{u}{t}+1\right) - \binom{u}{t}k^2(n\sqrt{u}q^d+1)\right) = \\
&= \Pr\left((|\hat{\mathbf{e}}|)_i > k^2(\sqrt{u}nq^d+1)\right) < \Pr((|\hat{\mathbf{e}}|)_i > k(2n\sqrt{u}kq^d+1)) < \\
&< (2n+1)2\sqrt{\frac{2}{\pi}}e^{-\left(\frac{\lceil c\xi\rceil-\frac{1}{2}}{\sqrt{2}\xi}\right)^2}
\end{aligned}$$

in a very similar fashion as in Theorem 3.16.

And finally, using that $c = \Omega(\sqrt{\lambda})$ and $\xi > 1$ we get that

$$\Pr\left((|\mathbf{x}+\hat{\mathbf{e}}|)_i > \frac{q}{4}\right) = \mathrm{neg}(\lambda).$$

□

5. Implementation

Once we have some new protocols described and proved correct and secure, the next logical step is to give a try to their implementation. This is due to several reasons. Firstly, if the protocols are to be used in any real life application, an implementation will be needed. Secondly, an implementation will be used to fine-tune the parameters used in the protocols to get more efficient yet secure performances. Finally the implementation will be used to analyze the computational and storage cost of the protocols.

This chapter will be divided in three parts: the first will cover how an implementation should deal with an active adversary sending non-matching values to honest players, an occurrence we call disputes, in a theoretical level; the second part will explain how we have implemented the simulation of the protocols, detailing what libraries, functions and ideas have been used; and the third part will deal with the analysis of the prototype implementation we have made.

5.1 Solving Disputes

The first important appreciation to be done while managing the implementation of especially the key generation, is that the protocol, as it is stated in Protocol 2.25, needs all the players submissions to properly generate a secure set of keys. Then if at any point a commitment check or an interval check in NIVSS fails because the *active* adversary has arbitrarily deviated from the protocol, the computations must halt and cannot come to an end without the contribution of every player. Therefore, Protocol 2.25 could be stuck in loop without being able to finish, thus making this particular key generation protocol not desirable. This occurrence of two players disagreeing on a privately sent value is called a *dispute*.

There are several ways one may think to try solving disputes. The first naive idea one may come to, is if we are using a secure (i.e. impregnable) communication medium that stores the encrypted messages and commitments, the affected player may publish its secret key and the messages involved. Then the rest of the players decide by majority which of the two players is corrupt and kick him out to start the protocol again with less players. This dispute resolution method has some glaring issues, mainly that a new set of public and secret keys have to be setup per player every time a dispute occurs, and since usually these kinds of keys are generated and distributed by an outside authority it would be desirable not to burn through them quickly.

Another more elaborate way to go to solve disputes would be to use a zero knowledge proof of knowledge of the matching of the commitment with the message. Let us define this concept.

Definition 5.1 (Definition 3.1, informal [BFM88])**.** A *Zero Knowledge Proof of Knowledge* is an interactive protocol in which a party named the *Prover* proves that he knows a value x satisfying some property to another party named the *Verifier* with the following properties:

1. **Completeness:** The probability of proving the knowledge of x given that the Prover knows x is overwhelming.
2. **Soundness:** The probability of proving the knowledge of x given that the Prover does not know x is negligible.
3. **Zero Knowledge:** The proof gives no other information that the Prover knows x. This is formalized by proving that there exists a simulator without access to x or the Prover such that for every statement to be proved it can simulate a transcript that looks like a correct interaction between the Prover and the Verifier.

Given this concept one could think that if we could give a zero knowledge proof of knowledge where we show that the value received and the commitment received do not match, then if the rest of the players decide by majority that the proof of knowledge is valid, we can kick the corrupt player out and start again with less players. However, this method still has issues, since it requires quite a lot of interaction, which is generally not advisable given that it is very costly.

What we will end up using will be going more in the line of the concept of *dispute control* introduced by Beerliová-Trubíniová and Hirt in [BTH06]. The basic idea behind dispute control is to divide the different phases of the protocol into different parts, then if a dispute between two players is detected (where at least one of them is corrupted), the execution of the part is slightly tweaked so that those two players do not need to interact again, thus giving a finite amount of possible disputes to $t(t-1)$, thus having a finite amount of part "resets" needed. However, this is a very general concept designed to be applicable to many different multi-party protocols. Using the specific properties of our key generation protocol and the idea of localizing pairs of disputing players at least one of them corrupt, we can give a dispute solving criteria that ensures Protocol 2.25 ends with only one iteration of the protocol and gives a correct output, compatible with Protocol 2.24 and not compromising its security.

The idea is that every time an honest player finds an inconsistency (usually in the commitments) he generates a dispute against whoever has fed him the inconsistent value. Then both players, the accuser and the accused, are removed from that section of the protocol and their contributions are discarded in that part. What we need to see is that this yields a correct output that does not compromise the security of the decryption phase. What we need to see is that the values output by a subset of players such that the proportion of corrupted players is still less than a third, are either indistinguishable from a correctly generated output or still follow the properties needed to be secure, i.e. reducible to an R-LWE distribution that is secure.

In case of the values that are uniformly distributed over an interval, it is obvious that the sum of fewer values uniformly distributed over that same interval (modulo the interval) still gives a uniformly distributed value over the interval.

Note that given the way we generate the secret key, we can see $\mathbf{s}$ as $\mathbf{s} = \mathbf{s}' + \mathbf{s}_e$, where $\mathbf{s}'$ is the contribution of the honest players and $\mathbf{s}_e$ is the contribution of the corrupt players. We know that $\mathbf{s}'$ is the sum of several samples of a rounded discrete Gaussian; however, we cannot say the same about $\mathbf{s}_e$, we only know that each contribution of the adversary is bounded.

Furthermore, the way we will handle the disputes we will have that

$$\#\{\text{Contributions following the distribution}\} \geq \tau := \#\{\text{Honest players}\} - \#\{\text{Disputes}\}.$$

We also know that every correct contribution follows a $\overline{\Psi}_{\frac{\xi}{q\sqrt{u}}}$. Using that for an active adversary we will have a maximum of $t < \frac{u}{3}$ corrupt players and t disputes, we get the following bounds

$$1 \geq \frac{\tau}{u} \geq \frac{u-2t}{u} > \frac{u-\frac{2u}{3}}{u} = \frac{3u-2u}{3u} = \frac{1}{3}$$

This means that we only need to see if the scheme is secure for the parameter $\frac{\xi}{\sqrt{u}q}$, which we have done using the LWE estimator by Albrecht et al. given in [APS15].

If we add the condition that any player can only be in one dispute (any dispute involving a player that has already accused or been accused after the first is made is ignored) and we make sure that no honest player can accuse any other honest player (this can be achieved through an impregnable communication

channel), then the key generation protocol will finish the first time it is attempted, and the $\overline{u}$ players that remain will satisfy, if $\overline{t}$ is the number of corrupt players that are left and $0 \leq x \leq t$ is the number of disputes

$$\overline{t} \leq t - x < \frac{u}{3} - x \leq \frac{u}{3} - \frac{2x}{3} = \frac{u - 2x}{3} = \frac{\overline{u}}{3}$$

since we know that no dispute will involve two honest players. Therefore we can say that the decryption protocol is still secure.

Finally for this section we would like to remark that we have only discussed disputes in Protocol 2.25 even though there is also interaction in Protocol 2.24, namely when changing the shares from additive to Shamir. The reason why no dispute resolution is needed in Protocol 2.24 is because the adversary only has access to (and therefore ability to compromise) the decryptions of those subsets of $t+1$ players where there is a corrupted player. However, these subsets are the minority by construction (since $t < \frac{u}{3}$), so we do not really mind if the decryption of these subsets is compromised. This means that not only dispute resolution is not needed in the decryption phase, we can even drop the commitment scheme too if it is so needed for performance issues.

5.2 Implementation Techniques

The final step before being able to properly analyze the behaviour of our simulation is to actually outline how the main components were programmed and which tools we used to do so. Before anything, however, we would like to remark that the implementation was not created as a completed version which can be used directly from code for any use due to time constraints among other reasons. Therefore, the intent of this implementation is to gauge how feasible our proposed protocols are in a real life implementation, given that as far as we know this is the first implementation of a threshold protocol based on the R-LWE problem. An example of this can be find in the way we simulate the protocol. Instead of doing multiple simultaneous processes we execute every step of the protocols for every player and then take the maximum time for every step as an estimate of the time spent by the multi-processed protocol in that step.

To do this implementation we have coded in C language, since the protocols are costly both in processing time and storage, and with using a very low-level language we gain time. We have also used several libraries, most importantly OpenSSL for cryptographic primitives such as secure Pseudo-Random Generators and the Secure Hash Algorithms; and FLINT (Fast Library for Number Theory), a very vast library from which we have used its capabilities to deal with multiple precision integers and polynomials with multiple precision integers as coefficients. Furthermore FLINT requires both the GMP and the MPFR libraries to be installed; for more details refer to the GitHub repository in Appendix A where the relevant programs are stored and these technicalities are better explained.

Without further ado now we can explain which techniques we used to implement the cryptographic primitives; which decisions we have taken towards using them and why; and if there is any other alternative of note.

5.2.1 Shamir Secret Sharing

The first inconvenient we faced was with Shamir Secret Sharing. Given that in essence what we share is every one of the coefficients (since we only perform linear operations onto the shares as elements in R_q), we can assume that we are sharing elements in $\mathbb{Z}_q$. However, given how Lagrange interpolation works for the reconstruction to work it is usually required that the sharing is done in a field.

Now we had to options: either view $\mathbb{Z}_q$ as a field (or nearly), or make all operations in $\mathbb{Q}$. For the sake of generality and to be truly able to use any q, we opted to go with the second option. However this has a downside, because we are using more memory than would otherwise be required. This is due to the fact that seeing the whole Shamir "process" as taking part in $\mathbb{Q}$, we cannot reduce modulo q any of the operations we do with shares, therefore instead of polynomials with coefficients in $\mathbb{Z}_q$ we have polynomials with coefficients far bigger than what is required.

Furthermore, the other route may be also a plausible solution if you do not need any $q \in \mathbb{Z}$ to be eligible as a modulo. This is because even if it is usually stated that a field is required to use Lagrange interpolation, as we can see in Technique 2.5 it is only required that the substraction of evaluation points has an inverse. Therefore, given that our evaluation points are the cardinals of the players, we would only need all prime factors of q to be bigger than u, which given that in real life applications the number of players tend to be small, it is not a big concession to make. That would allow us to do all operations modulo q, and maybe slightly improve the performance of the implementation, with the drawback of not being able to use any q.

5.2.2 PRF, PRSS and NIVSS

Next we needed a way to securely implement a PRF to be able to perform both the PRSS and NIVSS techniques (Tecniques 2.9 and 2.10). It is clear that simply using a Pseudo-Random Generator would not be useful to us, since we would need to have the same output given the same parameters, thus compromising the PRG. Therefore we used an HMAC which is a message authentication code based on hashes. It was proven in [Bel06] that an HMAC is a PRF under the condition that the underlying compression function is a PRF. To ensure this property is satisfied we use the HMAC based around SHA-3.

In regards to the PRSS (and NIVSS) implementation, we approached the matter trying to minimize the number of operations made with elements in R_q since, being polynomials with really big coefficients, they are probably the most time and memory consuming. To achieve the minimum number of operations, for every allowed subset of t+1 each player adds to its contribution in the PRSS all the contributions that have not been used by a player prior to him. The way we compute this is through the recursion seen in Algorithm 1, where we look for every H "forbidden subset" from which a player has the key. *State* stands for the players already picked for H, *Indexes* stands for the players still eligible and *Size* for how many players left to pick until we get t, so it is $t - length(State)$.

Algorithm 1: PRSS share

```
Input: State, Indexes, Size, Share, K_H, μ
if Size == 0 then
    Share+=Φ_{K_State}(μ);
else
    if length(Indexes) > Size then
        PRSS share(State, Indexes[2:], Size, Share, K_H, μ)
    end
    PRSS share(State+Indexes[1], Indexes[2:], Size-1, Share, K_H, μ)
end
```

The idea then is that for each allowed subset of $t+1$ players $B = (P_{i_1}, \dots, P_{i_{t+1}})$ with $i_j < i_{j+1}$, the initial *State* we send to player P_{i_j} are all i_k such that $i_k < i_j$, the initial *Indexes* will be all indexes not in *State* except i_j and *Size* will be $t - length(State)$. Then the first player will add the contributions of all

subsets he does not belong, the second player all the ones where neither him nor the first player do not belong (since any subset where the first player belongs will already have been counted), and so on, thus minimizing the number of operations needed to achieve the PRSS share.

However, there is another way of doing the PRSS share which even if it makes far more operations, we have recently found it may be beneficial. The method, which is described in [CDI05], makes use of some cleverly defined polynomials to be able to uninteractively compute a valid Shamir share of the PRSS. This has two benefits. The first one is the obvious one of being able to skip the interaction round needed for the usual change from additive sharing to Shamir sharing. The other one, which will probably save a lot of time, is that the share does not depend on the subset of players. This coupled with the properties of the PRF leads us to conjecture that instead of needing $u \geq 3t+1$ for an active adversary, we only require $u \geq 2t+2$, which is only one more than the absolute minimum required against active adversaries. This conjecture stems from the fact that due to the properties of the PRF, the probability of outputting two incorrect values that are equal in any subset with corrupted players is negligible. Therefore, if we have $2t+2$ players, we have at least two subsets of $t+1$ honest players which will output the same correct value. And given that the probability of outputting two equal incorrect values is negligible, we would not need anything else. Why reducing the number of needed players will allow to save a lot of time will be better seen in section 5.3.2.

5.2.3 Commitment Scheme

Finally in this subsection we only need to comment on how and where we have implemented the commitment scheme. In regards to how we have used hashes, the commitment of any element is its hash concatenated with a random string (its opening), in this case SHA-2 since as far as we know it is still secure enough. However, should the need arise it can be easily swapped for any other secure hash.

In regards to where, we have only needed to implement one round of commitments, since we only care about it on the step where all contributions are created. Once all the contributions have been properly created and sent we do not care about any malicious activities the adversary may do, since all the rest of steps are done localized for every allowed subset of $t+1$ players without any exterior interference. Therefore, since in the end a majority of the subsets will recover everything correctly we do not need any further commitment rounds. This commitment scheme serves the purpose of disabling the ability of the adversary to decide their contribution to the keys depending on the contributions of the honest players.

5.3 Analysis of the Simulation

In this final subsection we will discuss the results we have obtained from the simulation. First we will discuss what exact parameters we have chosen and why, and then we will discuss the proper results. The specifications of the system where we have executed the programs are found in Table 1, and once again remind that all relevant programs can be found in the GitHub repository through the link in Appendix A.

5.3.1 Choice of Parameters

We have two main constraints while choosing parameters: that the conditions for the correctness theorem are satisfied (Theorem 3.16), and that the instance of the R-LWE problems onto which we reduce our security is not easy to solve. To be able to verify this last fact we will use the LWE hardness estimator developed by Albrecht et al. in [APS15]. This is a sage module that provides functions for estimating the

Equipment	Version
Operating System	Ubuntu 18.04.5 LTS
CPU	Intel® Core™ i5-8500
Memory	15,4 GiB
Word Size	64 bits
CPU Clock Speed	3.00GHz x 6

Table 1: Specifications of the computer

concrete security of certain LWE instances by facing them against most well-known attacks. Note that it is a LWE and not a R-LWE estimator, however, given that there is no known way to distinguish the ideal case from the regular case in the lattice problem we can use the LWE estimator.

First we will want to reduce all the parameters we have to just n, q and $\alpha := \frac{\xi}{q}$, the parameters the estimator uses. We take $c = \sqrt{\lambda}$, which obviously satisfies $c = \Omega(\sqrt{\lambda})$, and d such that $\binom{u}{t} q^d + 1 = q^{\frac{1}{4}}$. We can see that the following parameters satisfy the requisites of Theorem 3.16:

$$\xi = \sqrt{\frac{q}{4\lambda(2n+1)q^{\frac{1}{2}}}} \implies \text{SigmaEnc} := \frac{\xi}{q} = \sqrt{\frac{1}{4\lambda(2n+1)q\sqrt{q}}}$$

$$\text{InterDec} := (c\xi)^2(2n+1)q^d = \frac{q}{4\sqrt{q}}q^d = \frac{q\left(q^{\frac{1}{4}}-1\right)}{4\sqrt{q}\binom{u}{t}}$$

where we will have $\mathbb{I}_{\text{Dec}}$ =[-InterDec,InterDec]. We have made these arbitrary decisions due to the fact that taking any decimal power of a multiple precision integer is not easy, whilst taking the squared root is well implemented. Therefore we will choose powers of $\frac{1}{2}$ whenever we can get away with it. We will also note that given that the bounds for Theorem 4.19 are essentially very similar, these parameters are still valid for an active adversary (although the choice of d would be different).

Furthermore, we will take $q \approx 2^{100}$, in this case the next prime next to 2^{100} to be able to compare with doing Shamir share in $\mathbb{Z}_q$ as a field if we implement it further down the line. We have previously mentioned that we take $q^{\Theta(\lambda)}$, though we could take $q = 2^{\Theta(n)}$ following the line of [BD10], the LWE protocol in which this proposal is based. However, Bendlin and Damgård mention in this article that due to Hermite's constant in the LLL algorithm to solve the SVP problem (which is hypothesized to be 1.02) the constant before n in $q = 2^{cn}$ tends to be very small. Therefore, since we need $q \approx 2^{100}$ to be able to securely drown the contributions in the Shamir share (we need q exponential over the security parameter), we need to take a rather large n. This is supported by the results found through the LWE estimator, which when taking q the next prime after 2^{100} and $\alpha = \text{SigmaEnc}$ we get that we need at least $n = 2^{11}$ to get 100 bits of security (and the previous powers of two from 2^8 had between 30 and 40 bits of security). It is also worthy of note that we use n a power of two since we know that these polynomials are cyclotomic and generate an R_q which is well behaved, however, other choices of n may still verify the security properties. Another favourable point towards using powers of 2 is that even if the current implementation uses Karatsuba to multiply polynomials, we are working on a variant of the Partial Fast Fourier Transform also called Number Theoretic Transform to multiply polynomials faster, which requires n to be a power of 2.

For the Key Generation protocol to have the following sigma and interval to satisfy the requisites of

Theorem 4.17.

$$\text{SigmaKG} := \frac{\xi}{q\sqrt{u}} = \frac{\text{SigmaEnc}}{\sqrt{u}}$$

$$\text{InterKG} := \text{InterDec}$$

where we will have $\mathbb{I}_{\text{KG}}$ =[-InterKG,InterKG].

5.3.2 Results of the Simulation

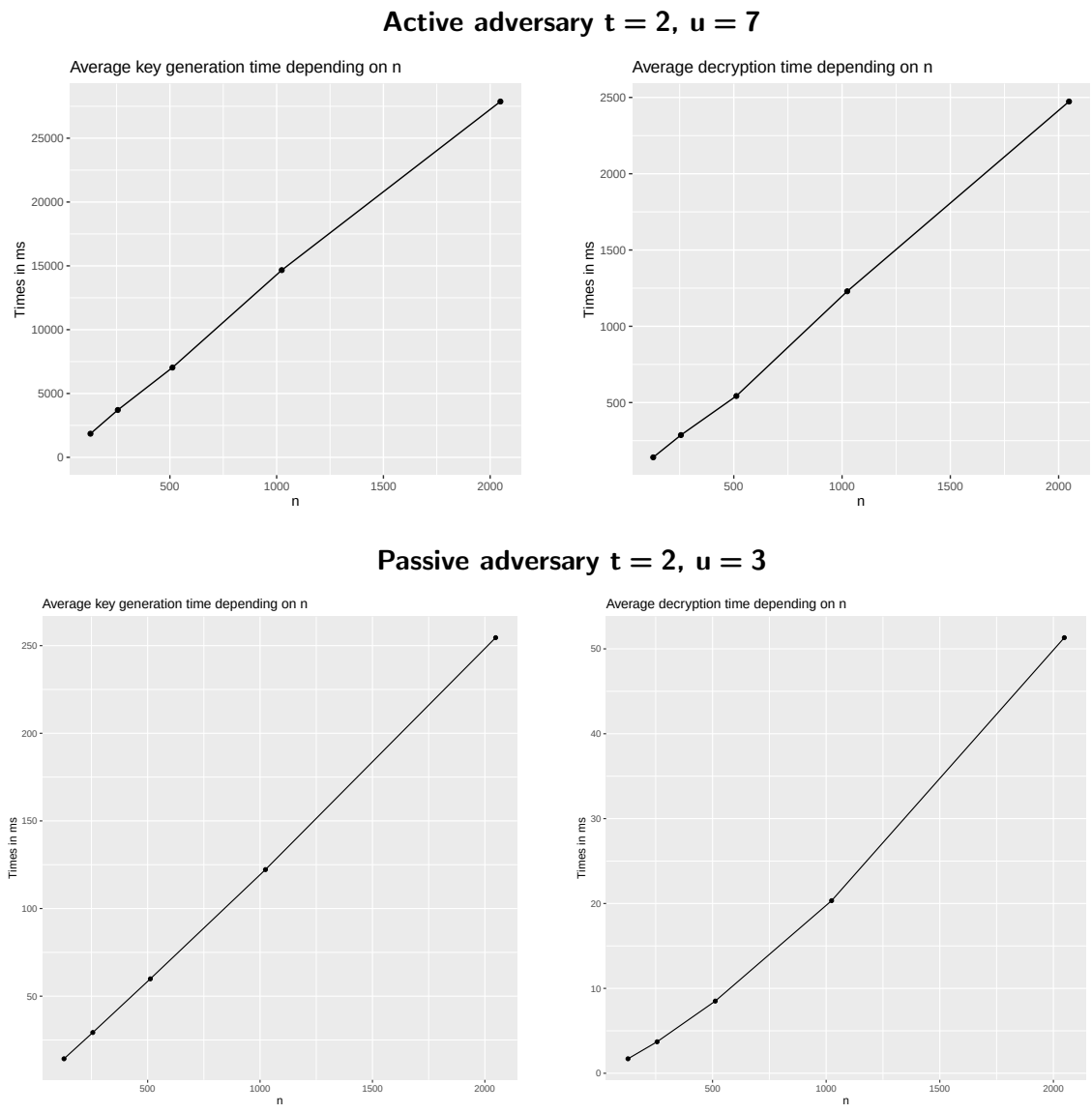

Figure 1: Times of the simulation for $n = 128, 256, 512, 1024, 2048$

We have done four extensive analysis: with respect to n in an active adversary setting and in a passive adversary setting and with respect to t in an active adversary setting and in a passive adversary setting. We

have not analysed extensively the dependence over q or ξ for two main reasons. Firstly, both q and ξ come virtually set by the need of q being exponentially large in its case and by the need to satisfy the conditions on Theorem 3.16 in case of ξ. Secondly, after some preliminary analysis doing some single executions, we found that greatly changing the values of q and ξ (from 17 to 2^{256} in case of q and from 10^{-30} to 0.1 in case of ξ) had little repercussions on execution times.

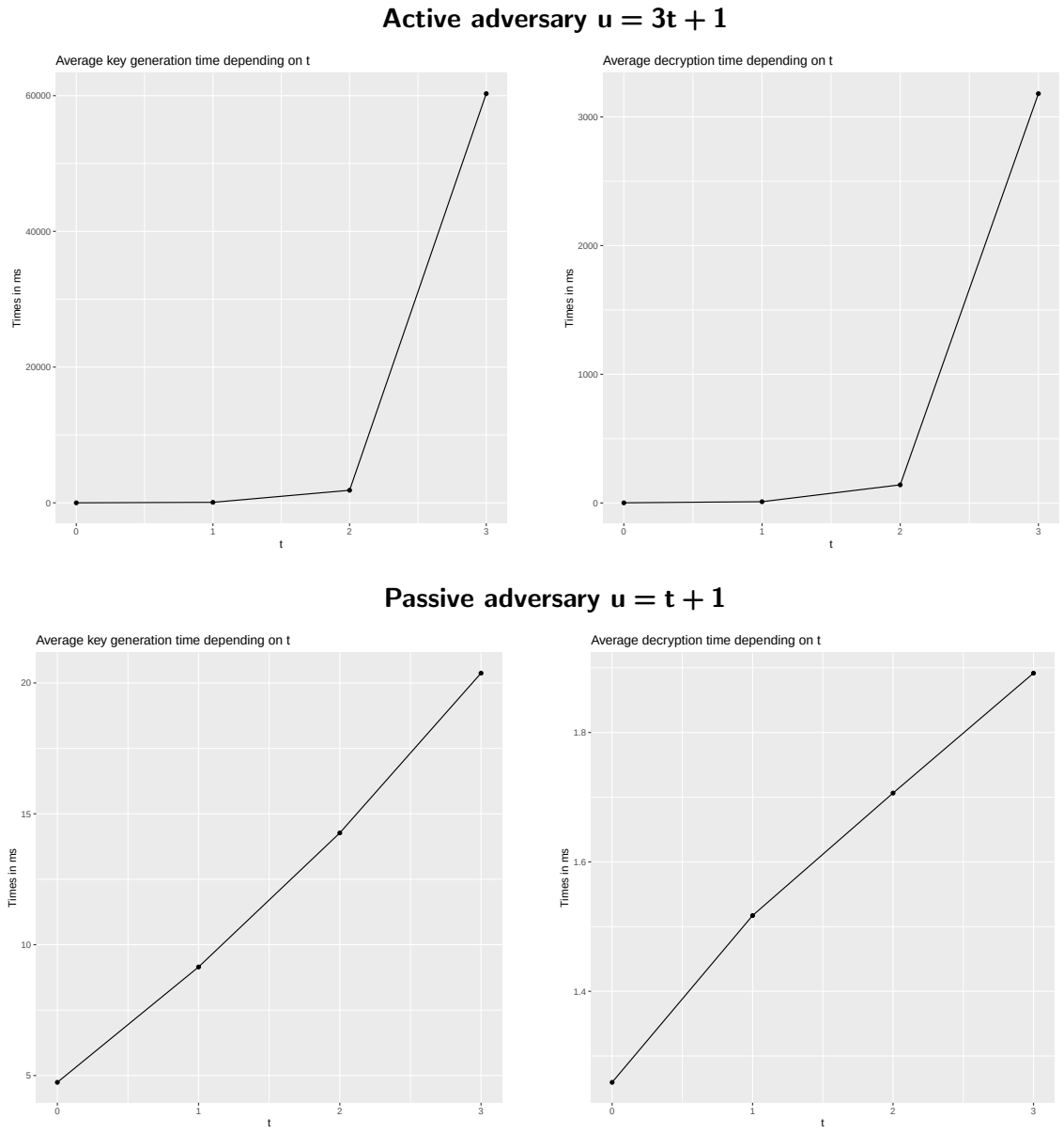

Figure 2: Times of the simulation for $t = 0, 1, 2, 3$

In the case of the analysis with respect to n in the active adversary case, we wanted to see how the times scaled as a function of n leaving q fixed at the next prime after 2^{100}, ξ computed accordingly from q and n, $t = 2$ and $u = 3t + 1 = 7$. We used $t = 2$ since this value or similar ones are one of the most frequently used values of t. Since what we wanted to see was how times scaled we have executed from $n = 2^7$ to $n = 2^{11}$, even if the first four values of n are not secure, so we could see the progression. For

every value of n we have performed 1000 iterations of the protocol and have averaged the times (that always coincided in the first significant digit, so there was not much standard deviation). In the first half of Figure 1 we can see the results for this case and it is clear that we have that the time scales linearly with n for both protocols in the range of n relevant to our study, which is a little better than we expected, given that we are performing several products of polynomials through the Karatsuba algorithm which scales with $n^{\log_2(3)} > n^{1.5}$, so the product must be dominated by some linear operation like the addition of polynomials or the initialization of these polynomials. Also of note is the fact that the time spent on the Key Generation simulation is consistently an order of magnitude higher than the Decryption simulation, however this is not too much of a problem since the Key Generation protocol should only be performed once for a number of decryptions, so it being more time expensive is not too much of a bother.

For the execution with $n = 2048$, which is the one that matters to us since it is the one that is secure, we get that the execution of the Key Generation simulation uses 2 GB of RAM, while the Decryption simulation uses less than 200 MB of RAM. This is a reasonable amount of storage, however we would like to note that there is a small issue with storage in the FLINT library we have not been able to completely solve yet, so if you execute too many iterations inside the program it will probably make your computer suffer, that is the reason why in our executions we have used the aid of a python script to overcome this problem.

In relation to the difference between the active adversary case and the passive adversary case, we fix the number of corrupt players and we compute $u = 3t + 1 = 7$ or $u = t + 1 = 3$. We care for this analysis because even if the most secure implementation is where we consider the active adversary, in some applications like some current electronic voting schemes they only require security against a passive adversary. As we can see in the second half of Figure 1, the Key Generation simulation for a passive adversary behaves linearly as well, however, in the Decryption simulation we can see the behaviour of $n^{1.5}$. This is due to the fact that there will be a much smaller proportion of additions in front of products, and therefore the product dominates over the addition in this case. This gives us the idea that for very few operations (which translates directly onto very few subgroups) the dominant behaviour is $n^{1.5}$ instead of n, but given that the number of operations will be small this does not give problems. The other results of importance can be found in Table 2, where we can see that the passive adversary setting needs times that are consistently two orders of magnitude smaller than in the active adversary setting, and gives us very promising execution times.

Key Generation		
n	Active	Passive
128	1851.72 ms	14.26 ms
256	3709.40 ms	29.26 ms
512	7033.16 ms	59.89 ms
1024	14664.54 ms	122.21 ms
2048	27872.93 ms	254.56 ms

Decryption		
n	Active	Passive
128	140.98 ms	1.70 ms
256	286.95 ms	3.70 ms
512	543.04 ms	8.50 ms
1024	1229.64 ms	20.33 ms
2048	2473.32 ms	51.32 ms

Table 2: Time comparison between an active and passive adversary

Once again, for the execution with $n = 2048$, the one that is secure, we get that the execution of the Key Generation simulation against a passive adversary uses approximately 20 MB of RAM while the Decryption simulation uses less than 10 MB of RAM. We would like to note that given how we have implemented this simulation, the values of storage needed are a very rough upper bound of the storage needed in a real life implementation. For starters, spreading this to u servers would mean that we need to at least divide the storage by u. Furthermore, due to several quirks of our implementation we believe that

even this is a rough upper bound of the storage needed.

In the case of the analysis with respect to t in the active adversary case, we wanted to see how the time scaled as a function of t having fixed $n = 128$, q as the next prime after 2^{100} and ξ computed accordingly. We decided to use $n = 128$ instead of $n = 2048$ due to time constraints. For every value of t from 0 to 3 (the case t=0 is analogous to not distributing the protocols and we stay at t=3 because the times grow very fast) we have performed 100 iterations of the protocol (once again due to time constraints) and have averaged the times (once again the standard deviation was small). In Figure 2 we can see the results which informs us that the time grows at least exponentially with t (it can potentially grow in a factorial way), which we already anticipated, given that the number of subsets grows through a binomial. This is the reason why being potentially able to reduce the amount of players required through the conjecture we have mentioned in Section 5.2.2 would be such a big deal. For example, in a preliminary analysis, using $t = 2$ and $u = 2t + 2 = 6$ already gives us better results, we get from around 28 seconds in the key generation phase with $t = 2$ and $u = 7$ to around 14 seconds, while the decryption phase comes down from around 2.5 seconds to around 1.5 seconds.

Regarding the difference between the active adversary case and the passive adversary case, we fix $u = t + 1$ and we iterate over $t = 0, 1, 2, 3$. We care for this analysis since we want to know if it still scales as badly with the number of corrupt players. It is clear from Figure 2 that this is obviously not the case, as it was expected. This difference is mainly due to the fact that the number of operations scales roughly with $\binom{u}{t}$, and therefore we get the following two very distinct ratios when going from t to $t + 1$. Then we get that in case of the active case the time scales roughly as an exponential function (if you put the graphic in logarithmic scale you get a line, we have left the linear scale to better compare cases), while the passive adversary case scales linearly or better with t. Once again we see that more than the amount of players, what this proposals struggles more is the growth of $\binom{u}{t}$.

6. Conclusions

In this thesis we have elevated the protocols proposed in [AE20] to another level of comprehension and applicability through four main ways: by improving and in some cases completely revamping the proofs given in [AE20]; by producing an original dispute resolution protocol for the Key Generation phase (since the Decryption phase did not need one); by doing a detailed analysis of which conditions are needed for the underlying R-LWE problem to be hard to solve; and by giving an implementation of a simulation of both protocols and analyzing the processing time and storage needed for them. Let us go through the conclusions we have been able to get from each one of them.

For the new proofs of both the correctness and security in [AE20], a couple of improvements stem from them. On the one hand, by improving the accuracy and legibility of all proofs we have made it easier to communicate the knowledge and, in case it was necessary, make any improvements or corrections to them. On the other hand, by completely revamping the security proofs from functionality-based to attack game-based we have for one part conformed to a more standard cryptographic proof style, and for the other we have improved composability, since attack games can be nested which can be of use in future applications or proofs.

In regards to the dispute resolution, it is a necessary protocol for any real life application of a protocol that regards functioning against an active adversary and requires input of all players, even if we were not aware of so when writing the future works for [AE20]. By giving our original dispute resolution protocol, we check that box and, even better, since our dispute resolution does not require any resets of the Key Generation protocol without compromising either correctness nor security, we give a solution to our problem in particular that is better than the general solution in [BTH06].

For the detailed analysis of the security, we have given a much more detailed and rigorous study of the security reductions than in [AE20]. This coupled with the analysis done with the LWE estimator developed in [APS15], we have been able to verify that indeed the conditions necessary for the protocol to be correct and the conditions necessary for the protocol to be secure are not mutually exclusive and that by combining them both we still get a R-LWE instance that is secure for reasonable parameters.

Finally in relation to the implementation of the simulation the main result we were going after was to see whether our proposal, already proven correct and secure, was in any way viable for a real life application. When answering this question we will be focusing mainly on the realm of electronic voting, since it is the main focus of the research group and where we were expecting of maybe applying our results. When seen through those eyes, then our protocols are totally viable against a passive adversary for any type of election, 51 ms for decrypting is a reasonable time per vote. While going up against an active adversary our protocols are viable for small elections, up to the order of 10000 votes.

Even with all the objectives we have achieved, this thesis still spawns future works we will tackle as soon as possible. On the side of the theorems and bounds, they can be optimized, especially the ones for ξ and $\mathbb{I}$. This would allow us to increase the bits of security to give us a little more wiggle room. Also the conjecture in Section 5.2.2 should be proven true (or not), to be able to improve the bound for corrupt players in an active setting, thus improving execution times. On the side of the implementation we have several upgrades in mind. Firstly, by modifying the PRSS share implementation to the one alluded in 5.2.2 we would be able to skip completely the interaction in the Decryption phase, which would be beneficial for a future implementation. Secondly, by implementing Shamir share modulo q as stated in Section 5.2.1, we could compare it with the current implementation to see if either computation time or storage is saved doing so. Thirdly, by implementing the fast product based on a variation of the Fast Fourier Transform we

could also compare with the current product and see if there is any improvement in computation times. Finally, for the codes in the GitHub repository in Appendix A to be able to blossom into a completely secure implementation of the protocol the codes should be analyzed carefully through the eyes of someone having secure programming in mind. The current codes where not made with that in mind for the sake of being able to see whether the implementation was viable or not, therefore attacks like Side Channel Attacks or similar will likely be able to disrupt the security of our implementation.

References

[AASA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al., *Status report on the second round of the NIST post-quantum cryptography standardization process*, US Department of Commerce, NIST (2020).

[AE20] Ferran Alborch Escobar, *Lattice-based threshold cryptography*, B.S. thesis, Universitat Politècnica de Catalunya, (2020).

[Ajt98] Miklós Ajtai, *The shortest vector problem in L_2 is NP-hard for randomized reductions*, Proceedings of the thirtieth annual ACM symposium on Theory of computing, (1998), pp. 10–19.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott, *On the concrete hardness of learning with errors*, Journal of Mathematical Cryptology **9** (2015), no. 3, 169–203.

[BD10] Rikke Bendlin and Ivan Damgård, *Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems*, Theory of Cryptography Conference, Springer, (2010), pp. 201–218.

[Bel06] Mihir Bellare, *New proofs for NMAC and HMAC: Security without collision-resistance*, Annual International Cryptology Conference, Springer, 2006, pp. 602–619.

[Ben80] Paul Benioff, *The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines*, Journal of statistical physics **22** (1980), no. 5, 563–591.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali, *Non-interactive zero-knowledge and its applications*, Proceedings of the twentieth annual ACM symposium on Theory of computing, 1988, pp. 103–112.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai, *Threshold cryptosystems from threshold fully homomorphic encryption*, Annual International Cryptology Conference, Springer, 2018, pp. 565–596.

[BS20] Dan Boneh and Victor Shoup, *A graduate course in applied cryptography*, 2020.

[BTH06] Zuzana Beerliová-Trubíniová and Martin Hirt, *Efficient multi-party computation with dispute control*, Theory of Cryptography Conference, Springer, 2006, pp. 305–328.

[Cat05] Dario Catalano, *Efficient distributed computation modulo a shared secret*, Contemporary Cryptology, Springer, 2005, pp. 1–39.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai, *Share conversion, pseudorandom secret-sharing and applications to secure computation*, Theory of Cryptography Conference, Springer, 2005, pp. 342–362.

[ElG85] Taher ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE transactions on information theory **31** (1985), no. 4, 469–472.

[GN08] Nicolas Gama and Phong Q Nguyen, *Finding short lattice vectors within Mordell's inequality*, Proceedings of the fortieth annual ACM symposium on Theory of computing, 2008, pp. 207–216.

[HLS18] Andreas Hülsing, Tanja Lange, and Kit Smeets, *Rounded gaussians: fast and secure constant-time sampling for lattice-based crypto*, 21st IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC 2018), Springer, 2018, pp. 728–757.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász, *Factoring polynomials with rational coefficients*, Mathematische annalen **261** (1982), 515–534.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, *On ideal lattices and learning with errors over rings*, Journal of the ACM (JACM) **60** (2013), no. 6, 1–35.

[Lyu12] Vadim Lyubashevsky, *Lattice signatures without trapdoors*, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 738–755.

[MR07] Daniele Micciancio and Oded Regev, *Worst-case to average-case reductions based on Gaussian measures*, SIAM Journal on Computing **37** (2007), no. 1, 267–302.

[MVOV18] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A Vanstone, *Handbook of applied cryptography*, CRC press, 2018.

[PE17] H. Pilaram and T. Eghlidos, *A lattice-based changeable threshold multi-secret sharing scheme and its application to threshold cryptography*, Scientia Iranica **24** (2017), no. 3, 1448–1457.

[PRSD17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz, *Pseudorandomness of Ring-LWE for any ring and modulus*, Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, 2017, pp. 461–473.

[Reg09] Oded Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Journal of the ACM (JACM) **56** (2009), no. 6, 1–40.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[SE94] Claus-Peter Schnorr and Martin Euchner, *Lattice basis reduction: Improved practical algorithms and solving subset sum problems*, Mathematical programming **66** (1994), no. 1, 181–199.

[Sha79] Adi Shamir, *How to share a secret*, Communications of the ACM **22** (1979), no. 11, 612–613.

[Sho99] Peter W Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM review **41** (1999), no. 2, 303–332.

[SRB16] Kunwar Singh, C Pandu Rangan, and AK Banerjee, *Lattice-based identity-based resplittable threshold public key encryption scheme*, International Journal of Computer Mathematics **93** (2016), no. 2, 289–307.

[ZXJ+14] Xiaojun Zhang, Chunxiang Xu, Chunhua Jin, Run Xie, and Jining Zhao, *Efficient fully homomorphic encryption from RLWE with an extension to a threshold encryption scheme*, Future Generation Computer Systems **36** (2014), 180–186.

A. Link to Repository

All relevant codes for the implementation can be found in the following GitHub repository, last commit made on June 18 2021:

https://github.com/FerranAlborch/RLWE-based-distributed-key-generation-and-threshold-decryption