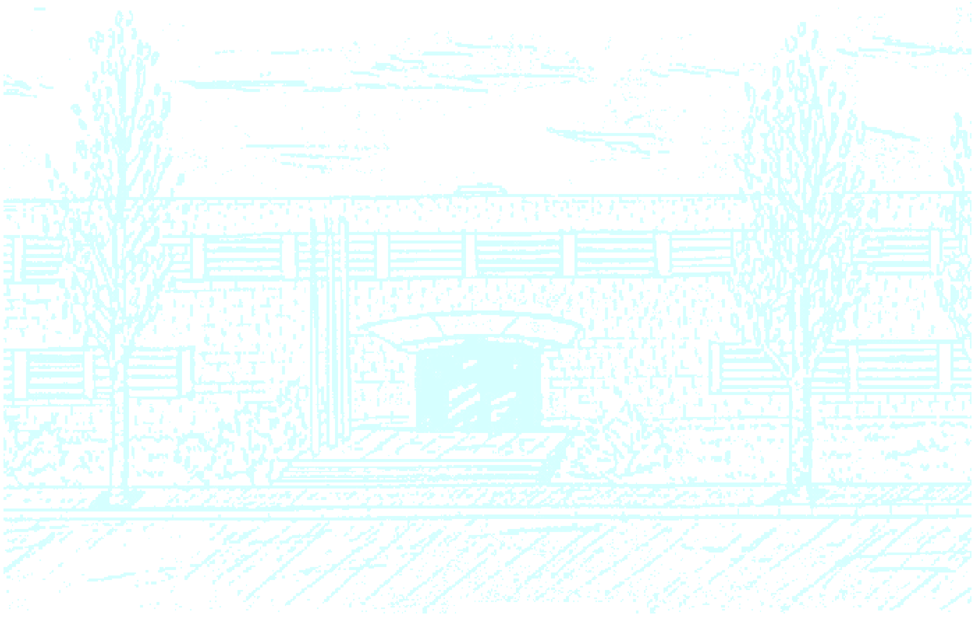
Title: Lattice-Based Threshold Cryptography

Author: Ferran Alborch Escobar

Advisor: Paz Morillo Bosch, Ramiro Martínez Pinilla

Department: Mathematics

Academic year: 2019-2020



Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

Lattice-Based Threshold Cryptography

Ferran Alborch Escobar

Supervised by Paz Morillo Bosch, Ramiro Martínez Pinilla

June, 2020

Thanks to the supervisors of this bachelor's degree thesis, Paz and Ramiro to introduce me to this subject and the huge help and enthusiasm they have given.

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 goal of this bachelor's degree thesis is to develop a post-quantum threshold cryptosystem, in particular based on Ring Learning with Errors, a lattice-based problem.

Keywords

Post-quantum cryptography, Lattices, Threshold cryptography

Contents

1. Introduction

To be able to hide information written in a message has been a more or less important problem for mankind ever since it invented writing: from ancient military leaders wanting to keep their messages about strategies secret, to nowadays, where virtually everything needs encryption due to the hyper-connected society we live in. Analogously, being able to break an encryption and read the message has also been a relevant problem for humanity, thus creating cat and mouse dynamics in cryptography, where ones chase after breaking the current encryption scheme while others chase after creating a new cryptosystem not yet breakable with the current tools at disposal.

The XXth century brought with it a new tool for both, the computer, and with it came the explosion of cryptography based on the discrete logarithm and prime factoring problems. The safety of these kinds of cryptography, given big enough numbers, enabled the huge development of the connected society, mainly internet, while still giving some protection to the information sent through this vast channel.

However, the rising of new computer systems as quantum computers threatens all cryptography based on the discrete logarithm and prime factoring problems, given that there is an algorithm that solves these problems in polynomial time in quantum computers, due to Peter Shor [21] in 1999.

Therefore, new "post-quantum" cryptography has been one of the main concerns of applied mathematics in the XXIst century so far. New brands of cryptography, such as *lattice-based* cryptography, *multivariate* cryptography, *hash-based* cryptography and more have been hot topic these last few years and great progress has been made. As stated in the title, this bachelor's degree thesis will be focused on lattice-based cryptography.

The internet and general web connectivity also spawned in the mid 1990s the concept of *threshold cryptography*, where the decryption protocol needs several players to cooperate for it to work, and less players than the ones needed cannot recover any relevant information of the message. *Secret sharing schemes*, that are the same concept but what the players have to recover can be anything instead of specifically an encrypted message, had been born before (specifically in 1979 in [20]), but its application in cryptography made less sense without the ability of immediate, or nearly, exchange of information between two or more parties. Threshold cryptography is for example used in electronic voting, where the various members of electronic polling stations must work together to decrypt the polling results.

The objective of this bachelor's degree thesis is to develop a threshold cryptosystem based on Ring Learning With Errors (RLWE), a lattice-based problem. For this intent we will take as starting point a cryptosystem presented by Bendlin and Damgård in [2] based on Learning With Errors (LWE), yet another lattice-based problem, and the goal is to take the main ideas and apply it to RLWE.

This work is structured as follows:

- **Theory**. We give notation and a brief introduction to lattices and to cryptography, so as to better understand the following sections.
- **Regev's cryptosystem**. We present the first lattice-based cryptosystem proposed by Regev in [18] (with the parameters proposed by Bendlin and Damgård in [2]) to familiarize ourselves with lattice-based crptography, and to ease the comprehension of Bendlin and Damgård's threshold cryptosystem which is based on Regev's.
- **Distributed cryptography**. After some preliminaries, we expose Bendlin and Damgård's threshold cryptosystem and distributed key generation.

- **RLWE threshold cryptosystem**. In this section we first present the RLWE cryptosystem proposed by Lyubashevsky, Peikert and Regev in [12], in which we will base our proposal of threshold cryptosystem. Then we expose our proposal, heavily inspired on Bendlin and Damgård's, and prove it correct and secure.
- **Conclusion**. We give a summary of the extent of our developments and how they could be furthered.

Inside each of these sections, and their respective subsections, we will use Definitions, Propositions, Lemmas, Theorems, Corollaries, Techniques, Protocols and Functionalities. The first five need no explaining, then Techniques will outline specific established procedures, Protocols will map out the threshold decryption and distributed key generation protocols, and Functionalities will define whatever functionalities we need to prove security through simulation. This differentiation is to highlight the difference between Protocols and Functionalities, while the first are implementable algorithms, the second, while mimicking the first, are only used to prove security of a certain protocol.

2. Theory

2.1 Notation

- $\langle \mathbf{a}, \mathbf{b} \rangle := \sum_{i=1}^{k} a_i b_i, \;\; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^k$ for some $k, q \in \mathbb{Z}_{>0}$.
- $X \sim \chi$ means X is a random variable following the distribution χ.
- Every element in $\mathbb{Z}_q^k$ for some $k \in \mathbb{Z}_{>1}$ is written in bold while the elements in $\mathbb{Z}_q$ are not.
- A function $f(x)$ is said to be *negligible* if $\forall k \in \mathbb{Z}_{>0}, \; \exists n_0 \in \mathbb{Z}_{>0}$ such that $\forall n \geq n_0, \; |f(n)| < \frac{1}{n^k}$.

2.2 Introduction to Lattices

In this subsection we will go through definitions and basic theorems related to lattices, since the encryption schemes later proposed will be based in lattice problems. This information can mostly be found in [16].

Definition 2.1. A *lattice* $\mathcal{L}$ is a set of points in an n-dimensional space $n \in \mathbb{Z}_{>0}$ (usually $\mathbb{R}^n$) such that:

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

Remark. Note that this properties give lattices a periodic structure. Usually a lattice is defined by a basis of vectors.

Definition 2.2. 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:

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

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.

Theorem 2.3. *Let* $\mathbf{B}, \mathbf{B}' \in \mathbb{R}^{n \times k}$. *Then,* $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}') \iff \exists \mathbf{U} \in \mathbb{Z}^{k \times k}$ *unimodular matrix such that* $\mathbf{B}' = \mathbf{B}\mathbf{U}$.

Proof. $\Leftarrow$) Assume $\mathbf{B}' = \mathbf{B}\mathbf{U}$, with $\mathbf{U}$ unimodular matrix. Then $\mathbf{B} = \mathbf{B}'\mathbf{U}^{-1}$, since $\mathbf{U}$ is invertible. Given that $\mathbf{U}, \mathbf{U}^{-1} \in \mathbb{Z}^{k \times k}$, this means that the vectors in $\mathcal{L}(\mathbf{B})$ are an integer linear combination of vectors in $\mathbf{B}'$ ($\mathcal{L}(\mathbf{B}) \subseteq \mathcal{L}(\mathbf{B}')$) and viceversa. Therefore $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$.

$\Rightarrow$) Assume $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$. This means that each column of $\mathbf{B}'$ is an integer linear combination of vectors of $\mathbf{B}$, therefore $\mathbf{b}'_i = \mathbf{B}\mathbf{u}_i$ with $\mathbf{u}_i \in \mathbb{Z}^k \; \forall i$, giving us $\mathbf{B}' = \mathbf{B}\mathbf{U}$ with $\mathbf{U} \in \mathbb{Z}^{k \times k}$. The same argument can be made in the other direction giving us $\mathbf{B} = \mathbf{B}'\mathbf{V}$, with $\mathbf{V} \in \mathbb{Z}^{k \times k}$. Adding it all together we get $\mathbf{B}' = \mathbf{B}'\mathbf{V}\mathbf{U}$. Then $\mathbf{B}'(\mathbf{V}\mathbf{U} - \mathbf{Id}) = 0$, and since $\mathbf{B}'$ is non-singular we get that $\mathbf{U}$ is a unimodular matrix. □

Remark. This last theorem is important since most lattice-based problems hardness depends on the basis defining the lattice given as an input, mainly the orthogonality of the basis.

Definition 2.4. The *orthogonality defect* of $\mathcal{L}(\mathbf{B})$ if $\mathbf{B} \in \mathbb{R}^{n\times n}$ is given by:

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

It can also be normalized taking the n-th root $\sqrt[n]{\delta(\mathbf{B})}$. A highly orthogonal basis has a low orthogonal defect.

Definition 2.5. 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.

Since hardness of most lattice problems depends on the orthogonality one may think how difficult finding a highly orthogonal base may be, but it turns out this is a difficult problem. There are several algorithms: the Lenstra-Lenstra-Lovasz (LLL) algorithm presented in [11], that performs gaussian elimination on the elements of the basis two by two; the Blockwise Korkine-Zolotarev (BKZ) reduction presented in [19], that improves the former using blocks of k 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.

Definition 2.6. 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:

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

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

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

Definition 2.7. The *dual lattice* of $\mathcal{L} \subset \mathbb{R}^n$ is $\mathcal{L}^* = \{\mathbf{w} \in \mathbb{R}^n | \langle \mathbf{w}, \mathbf{x}\rangle \in \mathbb{Z},\ \forall x \in \mathcal{L}\}$

Proposition 2.8. *Λ_q and $\Lambda_q^{\perp}$ are dual of each other up to normalization.*

Proof. We will divide the proof in two:

First we will see $\Lambda_q^{\perp}(\mathbf{A}) \subseteq q\Lambda_q(\mathbf{A})^*$:

$$\mathbf{y} \in \Lambda_q^{\perp}(\mathbf{A}) \subseteq \mathbb{Z}^n \Rightarrow \mathbf{y}^T\mathbf{A} \equiv 0 \pmod q \Rightarrow \mathbf{y}^T\mathbf{A} = q\mathbf{a}^T,\ \mathbf{a} \in \mathbb{Z}^m \Rightarrow (q^{-1}\mathbf{y})^T\mathbf{A} = \mathbf{a}^T \Rightarrow$$
$$(q^{-1}\mathbf{y})^T\mathbf{A}\mathbf{z} = \mathbf{a}^T\mathbf{z} \in \mathbb{Z},\ \forall \mathbf{z} \in \mathbb{Z}^m \Rightarrow (q^{-1}\mathbf{y}) \in \Lambda_q(\mathbf{A})^* \Rightarrow \mathbf{y} \in q\Lambda_q(\mathbf{A})^* \Rightarrow \Lambda_q^{\perp}(\mathbf{A}) \subseteq q\Lambda_q(\mathbf{A})^*$$

And now we will see $\Lambda_q^{\perp}(\mathbf{A}) \supseteq q\Lambda_q(\mathbf{A})^*$:

$$\mathbf{y} \in q\Lambda_q(\mathbf{A})^* \Rightarrow \mathbf{y} = q\mathbf{y}',\ \mathbf{y}' \in \Lambda_q(\mathbf{A})^* \Rightarrow \mathbf{y}^T\mathbf{A} = q\mathbf{y}'^T\mathbf{A} \Rightarrow \mathbf{y}^T\mathbf{A} = q\mathbf{a},\ \mathbf{a} \in \mathbb{Z}^m$$
$$\Rightarrow \mathbf{y}^T\mathbf{A} \equiv 0 \pmod q \Rightarrow \mathbf{y} \in \Lambda_q^{\perp}(\mathbf{A}) \Rightarrow \Lambda_q^{\perp}(\mathbf{A}) \supseteq q\Lambda_q(\mathbf{A})^*$$

□

Now we will define the problems on which security of lattice-based cryptography are usually based.

Definition 2.9. 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 in n dimension of the lattice [1], with only exponential algorithms known to solve it [8] 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.

The SVP problem is called Short Integer Solution (SIS) when described using the dual lattice, since finding short vectors of the lattice is equivalent to finding short integer vectors such that $\mathbf{Ax} = \mathbf{0}$.

Definition 2.10. Given $\mathbf{B}$ a base of the lattice $\mathcal{L}(\mathbf{B})$ and a vector $\mathbf{t} \in \mathbb{R}^n$ called the target vector,the *Approximate Closest Vector Problem* (γ-CVP) is finding a vector $\mathbf{u} \in \mathcal{L}(\mathbf{B})$ such that if

$$\mathbf{v} = \underset{\mathbf{w}\in\mathcal{L}(\mathbf{B})}{\arg\min} \|\mathbf{t} - \mathbf{w}\| \text{ then } \|\mathbf{u} - \mathbf{v}\| \leq \gamma\|\mathbf{t} - \mathbf{v}\|$$

However, working with a general lattice is not very efficient, therefore we use what we call ideal lattices, since its structure helps using faster computations, like using the Fast Fourier Transform, and no algorithm solving the previous problems is known to be able to take advantage of this structure.

Definition 2.11. Given a vector $\mathbf{f} \in \mathbb{R}^n$ we call the *transformation matrix* $\mathbf{F}$ the following matrix:

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

Definition 2.12. An *ideal lattice* is a lattice $\mathcal{L}$ which basis is the matrix $\mathbf{A} = [\mathbf{a}, \mathbf{Fa}, \dots, \mathbf{F}^{n-1}\mathbf{a}]$, where $\mathbf{F}$ is the transformation matrix of $\mathbf{a} \in \mathbb{Z}^n$.

Remark. These are called ideal lattices because we can see them as ideals in the ring $R_q = \mathbb{Z}_q[x]/\langle f_n(x)\rangle$ where $f_n(x) \in \mathbb{Z}_q[x]$ is some polynomial. Therefore we identify any vector $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{Z}_q^n$ with a polynomial $v(x) = v_1 + v_2x + \dots + v_nx^{n-1} \in R_q$. It is easily observed that by construction, multiplication of polynomials a, b in R_q is equivalent to multiplying the matrix $\mathbf{A}$ constructed from the vector $\mathbf{a}$ and its transformed matrix with the vector $\mathbf{b}$. Therefore the points in $\mathcal{L}(\mathbf{A})$ are the polynomials in $\langle \mathbf{a}\rangle \in R_q$.

We usually choose $f_n(x) = x^n + 1$ with n a power of 2, since it gives us good security reductions and the possibility to use the Fast Fourier Transform.

2.3 Introduction to cryptography

In this subsection we will go through some needed definitions related to cryptography.

Definition 2.13. [14] A *cryptosystem* or *encryption scheme* is a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ such that:

- $\mathcal{P}$ is a set called *plaintext space*.
- $\mathcal{C}$ is a set called *cyphertext space*. Where the elements in this space live depends heavily on the encryption and decryption protocols.
- $\mathcal{K}$ is a set called *key space*. Where the elements in this space live depends heavily on the encryption and decryption protocols.
- $\mathcal{E} = \{\mathrm{E}_k : k \in \mathcal{K}\}$ is a set of functions $\mathrm{E}_k : \mathcal{P} \times R \to \mathcal{C}$ called *encryption functions*. These functions depend on the key k. R is a *randomness space* because some encryption (and decryption) protocols use random values.
- $\mathcal{D} = \{\mathrm{D}_k : k \in \mathcal{K}\}$ is a set of functions $\mathrm{D}_k : \mathcal{C} \times R \to \mathcal{P}$ called *decryption functions*. These functions depend on the key k.
- $\forall e \in \mathcal{K},\ \exists d \in \mathcal{K}$ such that $\mathrm{D}_d(\mathrm{E}_e(m)) = m \ \ \forall m \in \mathcal{P}$.

Note that if we have $d = e$ 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*. Then we must have $d \in \mathcal{K}_s$ and $e \in \mathcal{K}_p$.

When using a cryptosystem to transfer confidential data, one needs several properties to ensure the transfer is indeed confidential.

Definition 2.14. A cryptosystem is said to be *correct* if the decryption protocol gives correct output except with negligible probability.

Definition 2.15. An *adversary* $\mathcal{A}$ is a Probabilistic Polynomial Time (PPT) algorithm or a tuple of them that maintains a state, so as to be able to have information of previous executions. Its abilities, characteristics and objectives depend on what kind of security you want to prove in a given cryptosystem.

An adversary is said to be *passive* if $\mathcal{A}$ can see all the messages and internal data of a corrupted player but still follows the protocol. It is called *semi-honest* if, as before, $\mathcal{A}$ can see all the messages and internal data of a corrupted player and corrupted players follow the protocol but may stop at any time. It is called *active* if besides stopping any corrupted player, $\mathcal{A}$ can cause them to deviate arbitrarily from the protocol. Furthermore, any of the types of adversaries mentioned before can be *static* if the set of corrupted players is set at the beginning of the computation of the protocol or *adaptative* if the set of corrupted players may vary during the application of the protocol [2].

Definition 2.16. [6] A cryptosystem is said to be *semantically secure* if given only cyphertext c and the public key k it is infeasible for an adversary $\mathcal{A}$ to derive any significant information about the message.

However proving semantical security is usually hard so we will define another type of security:

Definition 2.17. [10] A cryptosystem is said to be *Indistinguishable Chosen Plaintext Attack Secure (IND-CPA Secure)* if for any adversary $\mathcal{A}$ generating two same length messages and sending them to an oracle, where the oracle decides a message with a fair coin toss and returns it encrypted, the difference between the probability of the adversary guessing which message was encrypted and $\frac{1}{2}$ is negligible.

Lemma 2.18. *If a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-CPA Secure then it is semantically secure.*

For proof of this lemma refer to [7].

Definition 2.19. [9] (Informal). A *commitment scheme* is a cryptographic algorithm that allows a player to commit to a chosen statement without showing it to other players. The statement may be revealed afterwards.

3. Regev's cryptosystem

In this section we will go through Regev's cryptosystem, one of the first lattice-based cryptosystems presented, and it has heavily influenced latter lattice-based cryptography. It is a system based on the Learning With Errors problem (LWE), and was presented by Oded Regev in [18]. However we will present the choice of parameters given by Rikke Bendlin and Ivan Damgård in [2], since it is the cryptosystem in which their threshold cryptosystem is based.

First of all we need some definitions and notation.

Definition 3.1. [2] $n \in \mathbb{Z}_{>0}$ is said to be *B-smooth* if given the prime factorization of n, $n = \prod_{i=0}^{k} p_i$, then it verifies that $p_i \leq B$ and n is square free.

Definition 3.2. [2] Let χ be a probability distribution in $\mathbb{Z}_q$, $q \in \mathbb{N}$, $n \in \mathbb{Z}_{>0}$ and $\mathbf{s} \in \mathbb{Z}_q^n$.

Then $A_{\mathbf{s},\chi}$ is the distribution in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ given by $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a} \in \mathbb{Z}_q^n$ is chosen uniformly at random and $e \sim \chi$.

Definition 3.3. [2] The *decisional LWE problem* is being able to distinguish a sample from $A_{\mathbf{s},\chi}$ from the uniform distribution in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ with a probability that is non-negligibly bigger than $\frac{1}{2}$.

Definition 3.4. [2] The *search LWE problem* is being able to find $\mathbf{s}$ given polinomially many samples from $A_{\mathbf{s},\chi}$ with non-negligible probability.

Definition 3.5. [18] Ψ_β, $\beta \in \mathbb{R}^+$ is the distribution in $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ obtained by sampling a Gaussian random variable X, $X \sim N(0, \frac{\beta}{2\pi})$ and then reducing modulo 1. Therefore:

$$\Psi_\beta(r) \;=\; \sum_{k=-\infty}^{\infty} \frac{1}{\beta} e^{-\pi(\frac{r-k}{\beta})^2}, \; \forall r \in [0,1)$$

Definition 3.6. [18] The *discretization to* $\mathbb{Z}_q$, $q \in \mathbb{Z}_{>0}$ of any distribution in $\mathbb{T}$ ($\Psi : \mathbb{T} \to \mathbb{R}^+$), noted as $\overline{\Psi} : \mathbb{Z}_q \to \mathbb{R}^+$ is sampling from Ψ, multiplying by q, and then rounding to the closest integer. Therefore:

$$\overline{\Psi}(i) := \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}} \Psi(x)\, dx$$

Remark. We will further use $\overline{\Psi}_\beta$ built using the two definitions above. Note that Ψ_β has mean 0 and standard deviation β, and $\overline{\Psi}_\beta$ has a standard deviation bounded by βq. Also note that if $Y \sim \overline{\Psi}_\beta$ then $Y = \lfloor qX \rceil \pmod q$, with $X \sim \Psi_\beta$.

Given these definitions we can proceed to explain the cryptosystem, first stating its parameters, and then how it works.

The parameters are [2]:

- $n \in \mathbb{Z}_{>0}$ will be the dimension of the vectors. It is also called *security parameter* since security usually depends on the size of n.
- $q \in \mathbb{Z}_{>0}$ such that $q = 2^{\Theta(n)}$ and is B-smooth with $B = O(n^k)$ for some $k \in \mathbb{Z}_{>0}$ will be the modulo. It is also called *main parameter*.

- $m \in \mathbb{Z}_{>0}$ such that $m = O(n^3)$ will be the size of the public key.
- $\chi \sim \overline{\Psi}_\beta$ with $\beta = q^{\alpha-1}$ for $\alpha = 1/4$ will be the distribution followed by the disturbances. We have chosen this particular value of β for simplicity, though any β between 0 and 1 could be used. The standard deviation is bounded by $q\beta = q^\alpha = q^{\frac{1}{4}}$.

Now given all the parameters we can define the five components of the cryptosystem's tuple [2].

Definition 3.7. Given the parameters stated before, *Regev's cryptosystem's* tuple is as follows:

- $\mathcal{P} = \{0, 1\}$. We need to encrypt bit a bit.
- $\mathcal{C} = \mathbb{Z}_q^n \times \mathbb{Z}_q$.
- This is a public key encryption scheme, we have $\mathcal{K}_s = \mathbb{Z}_q^n$ and $\mathcal{K}_p = \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$.
- The secret key will be $\mathbf{s} \in \mathcal{K}_s$ chosen uniformly at random. The public key $(\mathbf{A}, \mathbf{b}) \in \mathcal{K}_p$, where $\mathbf{A}$ is chosen uniformly at random and $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod q$, where $\mathbf{e} \in \mathbb{Z}_q^m$ such that $e_i \sim \chi$. We will note as $\mathbf{A}_i$ the row number i of $\mathbf{A}$.
- $\mathcal{E} = \{\mathrm{E}_k : k \in \mathcal{K}_p\}$ such that, given $\mathbf{r} = (r_1, \dots, r_m) \in \{0, 1\}^m$ chosen uniformly at random and a bit γ:

  $$\begin{aligned} \mathrm{E}_k : \mathcal{P} &\to \mathcal{C} \\ \gamma &\mapsto (\mathbf{a}, b) \end{aligned}$$

  Where $(\mathbf{a}, b) = (\sum_{i=1}^m r_i\mathbf{A}_i, \gamma \left\lfloor \frac{q}{2} \right\rfloor + \langle \mathbf{r}, \mathbf{b}\rangle)$.
- $\mathcal{D} = \{\mathrm{D}_\mathbf{s} : \mathbf{s} \in \mathcal{K}_s\}$ such that $c = (\mathbf{a}, b)$:

  $$\begin{aligned} \mathrm{D}_\mathbf{s} : \mathcal{C} &\to \mathcal{P} \\ c &\mapsto \gamma \end{aligned}$$

  Where γ is 0 if after computing $v := b - \langle \mathbf{a}, \mathbf{s}\rangle \pmod q$ is closer to 0 than to $\left\lfloor \frac{q}{2} \right\rfloor$, and is 1 otherwise.

Now given the encryption and decryption scheme we need to prove it correct and secure. We will first prove it is correct for a certain type of distribution χ and then we will prove that $\overline{\Psi}_\beta$ verifies that condition.

Theorem 3.8. *[2] If $\forall k \in \{0, 1, \dots, m\}$ and $\tilde{e} \sim \sum_{i=1}^k \chi$ it holds that:*

$$\Pr(|\tilde{e}| \geq \sqrt[3]{q}) \leq 2^{-O(n)}$$

then except with negligible probability the decryption protocol will give correct output.

The proof is not done in [2], but we will give a proof of the theorem.

Proof. We know that the decryption is based on the following operation:

$$\begin{aligned} v := b - \langle \mathbf{a}, \mathbf{s}\rangle &\equiv_q \gamma \left\lfloor \frac{q}{2} \right\rfloor + \langle \mathbf{r}, \mathbf{b}\rangle - \langle \sum_{i=1}^m r_i \mathbf{A}_i, \mathbf{s}\rangle \\ &\equiv_q \gamma \left\lfloor \frac{q}{2} \right\rfloor + \sum_{i=1}^m r_i(\langle \mathbf{A}_i, \mathbf{s}\rangle + e_i) - \sum_{i=1}^m r_i\langle \mathbf{A}_i, \mathbf{s}\rangle \equiv_q \gamma \left\lfloor \frac{q}{2} \right\rfloor + \sum_{i=1}^m r_i e_i \end{aligned}$$

Now given this value v we know the decryption algorithm returns 0 if $|v| \leq \lfloor \frac{q}{4} \rfloor$ and returns 1 if $|v| > \lfloor \frac{q}{4} \rfloor$, if we look at the elements of $\mathbb{Z}_q$ from $-\lfloor \frac{q}{2} \rfloor$ to $\lfloor \frac{q}{2} \rfloor$. Then the protocol gives correct output except with negligible probability if and only if $|\langle \mathbf{r}, \mathbf{e} \rangle| < \lfloor \frac{q}{4} \rfloor$ except with negligible probability, and we also know since $\mathbf{r} \in \{0,1\}^m$ that $\tilde{e} := \langle \mathbf{r}, \mathbf{e} \rangle \sim \sum_{i=1}^{k} \chi$ if $k = \sum_{i=1}^{m} r_i$, since $e_i \sim \chi$. Then:

$$\Pr\left(|\langle \mathbf{r}, \mathbf{e} \rangle| \geq \left\lfloor \frac{q}{4} \right\rfloor \right) \leq \Pr(|\tilde{e}| \geq \sqrt[3]{q}) \leq 2^{-O(n)} < \frac{1}{O(n^k)} \; \forall k \in \mathbb{Z}_{>0}$$

and therefore negligible. □

Remark. Note that in the last line we have used that $\sqrt[3]{q} \leq \frac{q}{4}$ if $q \geq 8$ and since $q = 2^{\Theta(n)}$ this holds for $n \geq 3$. We can suppose $n \geq 3$ because for these kind of encryption schemes to be secure they usually use $n \approx 2^{10}$ which is obviously bigger than 3.

Lemma 3.9. *[2] For the choice of parameters made, for any $k \in \{0,1,\dots,m\}$, a constant $c \in (0,4)$ and $e \sim \sum_{i=1}^{k} \chi$ it holds that:*

$$\Pr(|e| \geq \sqrt[c]{q}) \leq 2^{-O(n)}$$

We will go through a more extended version of the proof given in [2].

Proof. Since $\chi = \overline{\Psi}_\beta$ we have that $e = \sum_{i=1}^{k} \lfloor qX_i \rceil \pmod q$, with $X_i \sim \Psi_\beta$. We also know, if $e' := \sum_{i=1}^{k} qX_i \pmod q$, that:

$$|e - e'| = \left| \sum_{i=1}^{k} \lfloor qX_i \rceil \pmod q - \sum_{i=1}^{k} qX_i \pmod q \right| =$$
$$= \left| \sum_{i=1}^{k} \lfloor qX_i \rceil - qX_i \pmod q \right| \leq \left| \sum_{i=1}^{k} \frac{1}{2} \pmod q \right| = \frac{k}{2} < m$$

Now, since $m = O(n^3)$ and $q = 2^{\Theta(n)}$ we also know $m < \frac{\sqrt[c]{q}}{2}$ for a big enough n, so it is sufficient to prove that $\Pr(|e'| \geq \frac{\sqrt[c]{q}}{2}) \leq 2^{-O(n)}$. Given that Ψ_β has mean 0 and standard deviation bounded by $\sqrt[4]{q}$ by construction, e' has mean 0 and standard deviation $\sqrt{k} \cdot \sqrt[4]{q}$. Using Chebyshev's inequality:

$$\Pr\left(|e'| \geq \frac{\sqrt[c]{q}}{2} \right) = \Pr\left(|e'| \geq \frac{\sqrt[c]{q}}{2\sqrt{k}\sqrt[4]{q}} \cdot \sqrt{k}\sqrt[4]{q} \right) \leq \Pr\left(|e'| \geq t \cdot \sqrt{k} \cdot \sqrt[4]{q} \right) \leq \frac{1}{t^2}$$

where $t = \frac{\sqrt[c]{q}}{2\sqrt{m}\sqrt[4]{q}}$. Using $m = O(n^3)$ and $q = 2^{\Theta(n)}$ again we get $2\sqrt{m} \leq \sqrt[d]{q}$ for some constant $d > 0$. This holds for n big enough since $2\sqrt{m} = O(n^{\frac{3}{2}})$ and $\sqrt[d]{q} = O(2^{\frac{1}{d}O(n)})$. Then $t \geq \frac{\sqrt[c]{q}}{\sqrt[d]{q}\sqrt[4]{q}} = q^{\frac{1}{c} - \frac{1}{d} - \frac{1}{4}}$, resulting in $\frac{1}{t^2} \leq \frac{1}{q^{2(\frac{1}{c} - \frac{1}{d} - \frac{1}{4})}}$. Since we want to see for which d this is negligible, and q is exponential in n we need to see when the exponent is bigger than zero:

$$2\left(\frac{1}{c} - \frac{1}{d} - \frac{1}{4}\right) > 0 \iff \frac{1}{d} < \frac{1}{c} - \frac{1}{4} \iff \frac{1}{d} < \frac{4-c}{4c} \iff d > \frac{4c}{4-c}$$

And given that $c < 4$ we can always pick such d, proving the lemma.

□

With the Lemma 3.9 and the Theorem 3.8 we have proven the correctness of the cryptosystem. Let us prove semantical security now.

Theorem 3.10. *[2] The cryptosystem is semantically secure under the assumption that* GAPSVP *is hard in the worst case.*

Like before we will give a slightly more fleshed out proof than the one given in [2].

Proof. To prove this we will reduce the GAPSVP to the decision LWE problem, and we will after prove that distinguishing between encryptions of 0 and 1 is at least as hard as the decision LWE problem, proving it then IND-CPA secure, which in turns implies it is semantically secure.

Firstly we will reduce in polynomial time the GAPSVP to the search LWE problem. To do so we will take advantage that we have chosen q to be exponentially large in n, and therefore we are able to use the reduction given by Chris Peikert in [15].

Secondly, we will reduce in polynomial time the search LWE problem to the decision LWE problem. To do this reduction we will base ourselves in the reduction given by Regev in [18], where it does exhaustive search over all elements in $\mathbb{Z}_q$. However, Regev used in this proof a modulus q which was prime, not B-smooth as we have. This is easily solved by doing the reduction modulo p_i each of the primes in q and recombine them using the Chinese Remainder Theorem.

Now we have reduced the GAPSVP, which we know that the best algorithms to solve it are exponential, to the decision LWE problem. This means that given an oracle that returns the solution to any given decision LWE problem we have an algorithm in polynomial time that solves GAPSVP. Therefore, given that GAPSVP is harder than polynomial, the decision LWE problem must be too.

Finally we need to prove that distinguishing between encryptions of 0 and 1 is at least as hard as the decision LWE problem. This was also proved by Regev in [18]. The basic idea is that given an algorithm that distinguishes between the encryptions of 0 and 1, which means an algorithm *Dist* that receives as input a cyphertext $C \in \mathcal{C}$ and outputs V or F, such that $|p_0 - p_1|$ is not negligible, where $p_0 = \Pr(Output = V | C = \mathrm{E}_k(0))$ and $p_1 = \Pr(Output = V | C = \mathrm{E}_k(1))$, with E_k encryption function; then *Dist* will also distinguish between an encryption of 0 and a random element of $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$ or between an encryption of 1 and a random element of $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$ since $p_r = \Pr(Output = V | C = r)$ with r a random element of $\mathbb{Z}_q^n \times \mathbb{Z}_q^n$ will need to be non-negligibly different from p_0 or p_1. Therefore distinguishing between encryptions of 0 and 1 is at least as hard as distinguishing if an element of $\mathbb{Z}_q^n \times \mathbb{Z}_q$ comes from a distribution $A_{\mathbf{s},\chi}$ (an encryption) or is an element picked at random, which is the decision LWE problem. □

4. Distributed cryptography

In this section we will go through, after some preliminaries, the threshold cryptosystem proposed by Bendlin and Damgård in [2], that works around Regev's cryptosystem. We will then use this cryptosystem as base to create our cryptosystem around Ring Learning With Errors.

4.1 Preliminaries

Definition 4.1. [20] A *threshold secret sharing scheme* of threshold t and u players is a scheme such that given some data D it divides it to 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).

Definition 4.2. We will call a *threshold cryptosystem* a secret sharing scheme where what we try to recover is plaintext from cyphertext.

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

Technique 4.3. [20] *Shamir secret sharing over $\mathbb{Z}_q$* allows u players share a secret $s \in \mathbb{Z}_q$ so that only $t+1$ players ($t < u$) can solve the secret together and whatever group of players of size less or equal than t cannot obtain any information on s. The algorithm works as follows:

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

Remark. 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:

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

Proposition 4.4. *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. We have $(a^1, \dots , a^u) \in \mathbb{Z}_q^u$ Shamir shares of $a \in \mathbb{Z}_q$ with threshold t, $t < u$, 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 (without loss of generality we can assume them the $t+1$ first), 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 (without loss of generality we can assume them the $t+1$ first), 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 (without loss of generality we can assume them the $t+1$ first), 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:

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

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

Proposition 4.5. *Let $(a^1, \dots, a^u) \in \mathbb{Z}_q^u$ be Shamir share of $a \in \mathbb{Z}_q$ with threshold $t+1$ $(t < u)$ 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 $t+1$ $(t < u)$.*

Proof. We know that for any $t+1$ shares of a (without loss of generality we can assume them the $t+1$ first) 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 $(b + a^1, \dots, b + a^n)$ (without loss of generality we can assume them the $t+1$ first), 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 $g(0) = b + a \pmod q$:

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

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

Corollary 4.6. *Let $\lambda_j = \prod_{k \neq j} \frac{i_k}{i_k - i_j}$ the Lagrange coefficients, for any degree n. Then $\sum_{j=1}^{n+1} \lambda_j = 1$.*

Proof. Let us do Lagrange interpolation of $f(z) = 1$, polynomial of degree at most n:

$$f(0) = \sum_{j=1}^{n+1} 1 \cdot \lambda_j = 1 \Rightarrow \sum_{j=1}^{n+1} \lambda_j = 1$$

□

Definition 4.7. [22] A *pseudorandom function* $\Phi_{\cdot}(\cdot)$ 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 predict the output of a given input only knowing other evaluations.

Technique 4.8. [5] *PseudoRandom Secret Sharing in $\mathbb{Z}_q$ (PRSS)* allows u players to non-interactively share a common random value with a threshold of t players $(t \leq u)$ given a pseudorandom function $\Phi_{\cdot}(\cdot)$ that given a seed and a cyphertext outputs values in the interval $I = [a, b]$ and whatever group of players of size less or equal than t cannot obtain any information on s. The algorithm works as follows:

- For each subset H of t players we define 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 pseudorandom number they are sharing is $x := \sum_H \Phi_{K_H}(c)$, with c a cyphertext. Since there are $\binom{u}{t}$ such subsets H, we know $x \in [\binom{u}{t}a, \binom{u}{t}b]$.

Remark. For any subset of $t+1$ players that wants to retrieve x they have all the keys, since each key is given to $u-t$ players.

Since this sharing is a type of additive sharing and the Shamir shares are polynomial shares, we will need a way to convert one to the other.

Technique 4.9. [3] *Converting between additive shares and polynomial shares* allows to, given $(a^1, \dots, a^{t+1})$ additive shares of a secret a of threshold t (i.e. $a = \sum_{i=1}^{t+1} a^j$), convert them to $(a'^1, \dots, a'^{t+1})$ Shamir shares of a for $t+1$ players of 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 $a'^j = f(j) := \sum_{k=1}^{t+1} f_k(j)$.

Remark. The resulting shares are truly Shamir shares of the secret a, since essentially what we did was do a Shamir share of each additive share and then adding them all together, and we have seen already that sum of Shamir shares gives Shamir shares of the sum, in this case a.

4.2 Bendlin and Damgård's proposal

Now that we have covered all the preliminaries we can present the encryption scheme with threshold t of Bendlin and Damgård presented in [2]. We will first present the encryption scheme, which uses a Trusted Third Party (TTP) to generate the keys, secondly we will prove its correctness and finally we will present two functionalities we will use to prove security for passive adversaries corrupting $t \leq u-1$ players and we will see that this proof works for proving security of this same protocol against a semi-honest adversary corrupting $t < \frac{u}{2}$ players and against an active adversary corrupting $t < \frac{u}{3}$ adversaries, all of it based on [2].

We will use the notion of a client, external to the threshold decryption protocol, who has an encrypted message and wants to decrypt it with the help of the players and chooses which $t+1$ of them help him, to make comprehension easier.

Protocol 4.10. The *decryption protocol* works as follows, if we take $I = [-\sqrt{q}, \sqrt{q}]$ the image of $\Phi.(\cdot)$:

1. A Trusted Third Party generates the keys $K_H \in \mathbb{Z}_q$ for every subset H of players of size t and distributes them according to PRSS (Technique 4.8). It also generates the secret key $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$ and the public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n+1}$ as in Regev's cryptosystem and sends to the players $(\mathbf{A}, \mathbf{b})$ and Shamir shares of $\mathbf{s}$.
2. Client receives cyphertext $(\mathbf{a}, b)$, decides which $t+1$ players will decypher the text and sends them the cyphertext and a list of the players participating in the decryption.
3. Each player P_j computes $\tilde{e}^j = b - \langle \mathbf{a}, \mathbf{s}^j \rangle$ that is a Shamir share of $\tilde{e} = e + \lfloor \frac{q}{2} \rfloor \cdot \gamma$ with $\langle \mathbf{r}, \mathbf{e} \rangle$ being $\mathbf{r}$ the random vector in $\{0,1\}^m$ and $\mathbf{e}$ the vector of errors, because of Proposition 4.4 and Proposition 4.5.

4. Each player P_j computes its additive share of $x := \sum_H \Phi_{K_H}(c)$ in the following way: in order, the additive share x^j is the sum of all $\Phi_{K_H}(c)$ no player before has, but P_j does. Then it converts this share to $\tilde{x}^j$ Shamir share of x using Technique 4.9.

5. Each player P_j computes $\tilde{x}^j + \tilde{e}^j$ Shamir share of $x + \tilde{e}$ and sends it privately to the client.

6. Client reconstructs $x + \tilde{e}$ using Lagrange interpolation, then returns 0 if $x + \tilde{e}$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ and returns 1 otherwise.

Remark. Note that the only difference between this threshold decryption protocol and Regev's scheme is that we add distortion x as a random number in the encryption. This randomness is necessary so as to not give any relevant information on the error e when sharing the decryption, since that would leak some information on the secret key.

Theorem 4.11. *Let $\binom{u}{t} < \frac{1}{4}\sqrt{q} - 1$ and assume that*

$$\Pr(|e| \geq \lfloor \sqrt{q} \rfloor) \leq 2^{-O(n)}$$

Then except with negligible probability the decryption protocol will have correct output.

Proof. We will prove it first for an encryption of 0 and afterwards for an encryption of 1.

Assume we have an encryption of 0. Then the client will have reconstructed $x + e$ from the shares. We want to see $|x + e| < \frac{q}{4}$ except with negligible probability. We know by definition that $|x| < \binom{u}{t}\sqrt{q}$, so by the first assumption of the theorem we have that $|x| < \frac{q}{4} - \sqrt{q}$. We also know by the second assumption of the theorem that $\Pr(|e| \leq \lfloor \sqrt{q} \rfloor) \geq 1 - 2^{-O(n)}$. Therefore $\Pr(|x + e| \leq \lfloor \frac{q}{4} \rfloor) \geq 1 - 2^{-O(n)}$, so $\Pr(|x + e| > \lfloor \frac{q}{4} \rfloor) \leq 2^{-O(n)}$ and the probability of error in output is negligible.

Assume we have an encryption of 1. Then the client will have reconstructed $x + e + \lfloor \frac{q}{2} \rfloor$. Now we want to see that $\Pr(|x + e + \frac{q}{2}| < \frac{q}{4}) < 2^{-O(n)}$, because the decryption will fail then. Since we are in $\mathbb{Z}_q$ we know that $|x + e + \frac{q}{2}| < \frac{q}{4} \iff |x + e| > \frac{q}{4}$. Now we have:

$$\Pr\left(|x + e + \frac{q}{2}| < \frac{q}{4}\right) = \Pr\left(|x + e| > \frac{q}{4}\right) = 1 - \Pr\left(|x + e| \leq \frac{q}{4}\right) \leq 2^{-O(n)}$$

□

Remark. Note that we have already proven in Lemma 3.9 that our distribution of errors satisfies the property required for the theorem. Also note that correctness for the cryptosystem places an upper bound for u the number of players if we fix q, or a lower bound for the modulus q if we fix u.

Now that we have proven correctness we will go forward to present the three functionalities we will use to prove security, and afterwards prove the security of the cryptosystem.

Functionality 4.12. We define the functionality $\mathcal{F}_{KeyGen}$ that works as following:

1. When given "start" by all players, $\mathcal{F}_{KeyGen}$ chooses a secret key $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$ uniformly at random and constructs the public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n+1}$, as in Regev's cryptosystem.

2. For each subset H of size t of players $\mathcal{F}_{KeyGen}$ chooses key $K_H \in \mathbb{Z}_q$ uniformly at random.

3. $\mathcal{F}_{KeyGen}$ receives from the adversary, for each corrupted player P_j, the shares $s_i^j \in \mathbb{Z}_q$, $\forall i = 1, \ldots, n$.

4. Using Lagrange interpolation and the shares s_i^j received from the adversary, $\mathcal{F}_{KeyGen}$ computes a polynomial $f(x_1, \dots, x_n) = (f_1(x_1), \dots, f_n(x_n))$, $f_i(x_i) \in \mathbb{Z}_q[x]$ of degree t that goes through the shares and such that $f(0) = \mathbf{s}$, and then computes the Shamir shares for every $i = 1, \dots, n$.

5. $\mathcal{F}_{KeyGen}$ sends privately to every player P_j their share $\mathbf{s}^j = (s_1^j, \dots, s_n^j)$ of the secret key and K_H for all H such that $P_j \notin H$.

6. $\mathcal{F}_{KeyGen}$ sends public key to all players and adversary.

Remark. Note that we give the adversary the ability to choose which shares of the secret the corrupted players get. This is done this way since we will need to know what are the corrupted players when proving security. The easiest way to do so is like this, and given the fact that this kind of adversary is stronger than usual, the protocol will also be secure under weaker adversaries.

Functionality 4.13. We define the functionality $\mathcal{F}_{KeyGen-and-Decrypt}$ or $\mathcal{F}_{KG-D}$ that implements the same steps as Protocol 5.10:

1. When receiving "start" of all players send "start" to $\mathcal{F}_{KeyGen}$, receives all the shares of $\mathbf{s}$ and all the K_H and forwards it to every player.

2. When receiving "decrypt $(\mathbf{a}, b)$" from the client, send "decrypt $(\mathbf{a}, b)$" to all players participating in the decryption and the adversary, and then reconstruct the encrypted bit γ.

3. In the next round, send γ and the shares of every player to the client and adversary.

Functionality 4.14. We define the new functionality $\mathcal{F}_{SimDecrypt}$ that works as follows, assuming B the set of corrupted players:

1. When given "start" by all players in B $\mathcal{F}_{SimDecrypt}$ gives "start" to $\mathcal{F}_{KeyGen}$ and receives all the shares of $\mathbf{s}$ and the keys K_H.

2. $\mathcal{F}_{SimDecrypt}$ sends the shares of $\mathbf{s}$ and their respective K_H to the players in B.

3. When receiving "decrypt $(\mathbf{a}, b)$" from the client, $\mathcal{F}_{SimDecrypt}$ sends "decrypt $(\mathbf{a}, b)$" to all players in B, receives their shares of the decryption and decrypts the bit γ with these shares, the shares of the secret key of the honest players and the decryption protocol.

4. For the honest players $\mathcal{F}_{SimDecrypt}$ must simulate shares to broadcast in the next round, so for every K_H the adversary does not know (i.e. $B \subseteq H$) $\mathcal{F}_{SimDecrypt}$ generates a random integer $r_H \in [-\sqrt{q}, \sqrt{q}]$, and we define $y = \sum_{B \not\subseteq H} \Phi_{K_H}(c) + \sum_{B \subseteq H} r_H$. Now using Lagrange interpolation $\mathcal{F}_{Sim}$ computes shares of $y + \gamma \lfloor \frac{q}{2} \rfloor$ using Lagrange interpolation, the shares of the decryption received by the players in B in step 3 and γ calculated in step 3.

5. In the next round send γ bit corresponding to cyphertext $(\mathbf{a}, b)$ and the shares of $y + \gamma \lfloor \frac{q}{2} \rfloor$.

Theorem 4.15. *When given access to the functionality $\mathcal{F}_{KeyGen}$ and assuming that $\Phi.(\cdot)$ is a pseudo-random function, the decryption protocol is secure. The adversary is assumed to be passive and static, corrupting up to $t = u - 1$ players.*

Proof. To prove this we will use that the Regev's cryptosystem is semantically secure (as we have proved in Theorem 3.10), therefore we only need to prove that no information is leaked when sharing the secret key and the decryption. We will do so by creating a functionality $\mathcal{F}_{SimDecrypt}$ that simulates what $\mathcal{F}_{KG-D}$ does but using random input instead of using the secret key, and then proving that both functionalities are statistically indistinguishable to the adversary. This will mean that the adversary cannot differentiate random input from the secret key, thus meaning that no information about the secret key is leaked when sharing the decryption.

Now we need to prove that indeed both functionalities are indistinguishable to the adversary. The shares of the adversary, the K_H the adversary receives and the interaction between the functionalities and the adversary are obviously indistinguishable. γ will be correct in both cases given Theorem 4.11, and furthermore, $y+\gamma\lfloor\frac{q}{2}\rfloor$ will be an effective "decryption" of γ in the sense that it will be closer to 0 if $\gamma = 0$ and closer to $\frac{q}{2}$ if $\gamma = 1$, also by direct consequence of Theorem 4.11 (what we are adding to $\gamma\lfloor\frac{q}{2}\rfloor$ in the worst case scenario is smaller).

The only thing we need to see now is that $y+\gamma\lfloor\frac{q}{2}\rfloor$ and $x+e+\lfloor\frac{q}{2}\rfloor$ are indistinguishable to the adversary. y and x are computationally indistinguishable to the adversary given the properties of pseudorandomness of $\Phi.(\cdot)$. Now we know that y is the sum of at least one element taken uniformly at random from an interval of size $2\sqrt{q}$, and e is distributed in an interval of size $2\sqrt[3]{q}$, which is exponentially smaller (in n) than $2\sqrt{q}$. Therefore, the way y and $y+e$ are distributed are statistically indistinguishable, therefore since y and x are computationally indistinguishable, y and $x+e$ are statistically indistinguishable. Therefore, the outcome is the one expected. $\square$

Remark. What this proof says is, broadly, that given two "decryptions" of γ (in the same sense as before), one deriving from the secret key and one randomly constructed, they are indistinguishable. This means that by using the pseudorandomly generated number x the cryptosystem effectively mutes any relevant information of $\mathbf{e}$ and in consequence of $\mathbf{s}$.

Basing ourselves on this cryptosystem is easy to see that it will also work, with slight adjustments, against a semi-honest adversary corrupting up to $t < \frac{u}{2}$ players (there will always be $t+1$ non-stopped players to decrypt), and against an active adversary corrupting up to $t < \frac{u}{3}$ players using the same argument as in [4] (it will always be verifiable who is saying the truth, because more than half of the players not in the protocol will not be corrupt).

Note that even if this protocol is correct and secure, it is highly inefficient, given that the number of K_H grows exponentially with u if t is a constant fraction of u. Therefore this protocol is only feasible with a small number of players.

4.3 Distributed key generation

The given protocol relies in a TTP to generate the keys, so it would be improved with a protocol to do the distributed key generation. To give it we will need to introduce a new secret sharing protocol and a new functionality.

Technique 4.16. [5] *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 $a \in \mathbb{Z}_q$ and a pseudorandom function $\phi.(\cdot)$ that given a seed and a outputs values in the interval $I = [a, b]$. 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 pseudorandom value the players share $r = \sum_H \phi_{K_H}(a)$, 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.

Remark. We would like to note a few things about NIVSS. Firstly, the verifiability of this protocol resides in the fact that every player can check the values other players output for the noise generated by the pseudorandom function if they both have the same key. Secondly, the pseudorandom function used in NIVSS is different than the one we have used in PRSS before, since it has different domain and range. And finally, even if the dealer D sends a random value y instead of $s - r$, the protocol would still provide the rest of the players with a valid share of some value, in particular of $y + r$.

We will now define the protocol for distributed key generation, for which we will need to assume private communication channels between players, and then define two functionalities we will use to prove its security. However, we will also need to prove that adding the distributed key generation does not render the decryption protocol insecure, so we will define another functionality to prove the protocol's security against an active, static adversary corrupting $t < \frac{u}{3}$ players. Unlike before, the "naïve" protocol one could think against a passive adversary would not work against an active adversary, therefore we will prove this protocol is secure against an active, static adversary corrupting $t < \frac{u}{3}$ players and we will after see that it also works against a passive adversary corrupting $t \leq u - 1$ players and a semi-honest adversary corrupting $t < \frac{u}{2}$ players, all of it based on [2].

Protocol 4.17. The *key generation protocol* works as follows if we take $I = [-\sqrt[3]{q}, \sqrt[3]{q}]$ the image interval of $\phi_{\cdot}(\cdot)$:

1. For the secret key $\mathbf{s} \in \mathbb{Z}_q^n$ each player P_j chooses uniformly at random $(s_1^j, \dots, s_n^j) \in \mathbb{Z}_q^n$ their contribution on the secret key and shares it with all the players using Shamir secret sharing, first using a commitment scheme to commit every s_i^j. Then the players will have, by adding all the shares received by other players Shamir shares of $\mathbf{s} = \left(\sum_j s_1^j, \dots, \sum_j s_n^j\right)$ because of Proposition 4.4.

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. Then the players will have, by adding all the shares received by other players Shamir shares of $K_H = \sum_j K_H^j$ because of Proposition 4.4. 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 error contributions each player P_j chooses its error contribution e_i^j for every row i of the matrix according to the distribution $\overline{\Psi}_\beta$. Then they act as the dealer in a NIVSS to share e_i^j to all players. All players verify the value broadcast when doing the NIVSS $e_i^j - \sum_H \phi_{K_H}(a)$ is in the interval $[-\binom{u}{t}\sqrt[3]{q}, \binom{u}{t}\sqrt[3]{q}]$. Now all players have shares of every e_i^j and by their lineality also of $e_i = \sum_j e_i^j$.

4. For $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ we proceed as in generating the secret key $\mathbf{s}$ m times, and finally all players send to all players their share on the public key so that everyone can recover it.

5. Every player computes locally their Shamir share on $b_i = \langle \mathbf{A}_i, \mathbf{s} \rangle + e_i$ by performing these same operations with the shares they have on $\mathbf{s}$ and e_i (having previously converted the shares to Shamir using Technique 4.9).

Remark. We would like to note some facts about the protocol. Firstly, given how the shares in PRSS, and extensively NIVSS, work, the shares every player has on b_i will vary depending on which players are contributing to reconstruct it. Secondly, by verifying whether the value broadcast when doing every NIVSS is in the interval $[-\binom{u}{t}\sqrt[3]{q}, \binom{u}{t}\sqrt[3]{q}]$, what we are really are verifying is if e_i^j is in the interval $[-\sqrt[3.5]{q}, \sqrt[3.5]{q}]$, which is needed for the correctness of the decryption protocol, and we know that a value following $\overline{\Psi}_\beta$ will be outside of the interval $[-\sqrt[3.5]{q}, \sqrt[3.5]{q}]$ only with negligible probability because of Lemma 3.9. Finally, this distributed key generation is, once again, highly inefficient given the amount of Shamir shares and application of NIVSS required.

Functionality 4.18. We define the functionality $\mathcal{F}_{Rand}$, which has four different commands and works as follows:

- **"Shared value from D"** where D is a player. The player D sends to $\mathcal{F}_{Rand}$ a value chosen uniformly at random in $\mathbb{Z}_q$ and the adversary sends to $\mathcal{F}_{Rand}$ a set of shares for the corrupt players. Then the functionality uses Lagrange interpolation to construct consistent Shamir shares of the value sent by D for the honest players and finally sends the shares to each player.
- **"Random shared value"**. $\mathcal{F}_{Rand}$ calls "Shared value from P_i" for all players, and then each player locally adds the shares of all the random numbers, giving shares of $s = \sum_i s_i$.
- **"Random value to B"** where B is a set of players. $\mathcal{F}_{Rand}$ calls "Random shared value" and all the players send their shares to the players in B.
- **"Constrained value from D"** where D is a player. Player D performs a NIVSS of its secret s, with the functionality verifying afterwards whether the broadcast value is in $[-\binom{u}{t}\sqrt[3]{q}, \binom{u}{t}\sqrt[3]{q}]$, and returning "Fail" if it is not.

Functionality 4.19. Given access to the functionality $\mathcal{F}_{Rand}$ we define the functionality $\mathcal{F}_{KeyGen'}$ that works as follows:

1. When receiving "start" from all honest players call "Random shared value" n times to generate and share the secret key $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$.
2. For each subset H of t players call "Random value to A" where A is the complement of H to generate the keys K_H used for the PRSS in the decryption.
3. Call "Random value to P" with P the set of all players nm times to generate $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.
4. Receive the broadcast value from the NIVSS done for each e_i^j from player P_j and call "Constrained value from P_j".
5. Broadcast the public key given by $(\mathbf{A}_i, b_i = \langle \mathbf{A}_i, \mathbf{s} \rangle + e_i)_{i=1}^m$ and every player's share of b_i.

Remark. Note that here again as in Functionality 4.12 we give the adversary the ability to choose the shares of the corrupted players. The reason is the same as in Functionality 4.12.

Functionality 4.20. We define the functionality $\mathcal{F}_{SimGen}$ as follows:

1. Receive the shares of the corrupted players of s_i^j, $i = 1, \dots, n$ and for every player P_j from the adversary and generate a random vector $\mathbf{s} \in \mathbb{Z}_q^n$. Then send the shares back to the adversary.

2. Receive the shares of the corrupted players of K_H^j for every player P_j and every subset H of t players from the adversary, and generate a random number K_H for every H. Then send the shares back to the adversary.

3. Receive the shares of the corrupted players of $\mathrm{A}_{i,k}^j$ for $i = 1, \dots, m$; $k = 1, \dots, n$ and every player P_j from the adversary and generate random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$. Then send the shares back to the adversary.

4. To simulate the sharing of the values e_i^j receive the broadcast values for every corrupted player and return "Fail" if they are not in the interval $[-\binom{u}{t}\sqrt[3]{q}, \binom{u}{t}\sqrt[3]{q}]$. Then simulate the NIVSS for every honest player taking random numbers as keys and a random noise contribution e_i^j in $[-\sqrt[3.5]{q}, \sqrt[3.5]{q}]$.

5. Reconstruct the public key $(\mathbf{A}_i, b_i = \langle \mathbf{A}_i, \mathbf{s} \rangle + e_i)_{i=1}^m$ and since $\mathcal{F}_{SimGen}$ knows the shares of the corrupted players, construct shares of b_i consistent with the adversary for all honest players with Lagrange interpolation.

6. Broadcast $(\mathbf{A}, b)$ and the shares of the honest players of b_i.

Theorem 4.21. *The Key Generation protocol securely implements the functionality $\mathcal{F}_{KeyGen'}$. The adversary is assumed to be active and static, corrupting up to $t < \frac{u}{3}$ of the players.*

Proof. As in Theorem 4.15, we have defined a functionality $\mathcal{F}_{SimGen}$ that simulates what $\mathcal{F}_{KeyGen'}$ does but using random values, and now we will prove that it is indistinguishable to the adversary which functionality it is interacting with.

Since every step is done exactly in the same order and are distributed exactly the same, it is easy to see that $\mathcal{F}_{KeyGen'}$ and $\mathcal{F}_{Sim}$ are indistinguishable to the adversary. □

Remark. Note that what we are proving here is slightly different that what we were proving on Theorem 4.15. What we want to know is that when distributing the generation of the keys the players don't know any information on the individual values chosen by each other of the players, which is what we have proven by seeing it is indistinguishable from a functionality choosing all the values at random.

Now we only need to prove that with the distributed key generation the decryption protocol is still secure. We will define another functionality and then prove the security, also based on [2].

Functionality 4.22. We define the functionality $\mathcal{F}_{KeyGen'-and-Decrypt}$ or $\mathcal{F}_{KG'-D}$ as Functionality 4.13, $\mathcal{F}_{KG-D}$ but using $\mathcal{F}_{KeyGen'}$ instead of $\mathcal{F}_{KeyGen}$.

Theorem 4.23. *Assuming we use $\mathcal{F}_{KeyGen'}$ to generate the public key $(\mathbf{A}, \mathbf{b})$ and the secret key $\mathbf{s}$, and that the number of players u satisfies $u\binom{u}{t} < \frac{\sqrt[10]{q}}{m}$. If* GAPSVP *is hard in the worst case, then encryption under* $\mathbf{s}$ *is semantically secure against any active polynomial time adversary corrupting $t < \frac{u}{3}$ players that interacts with $\mathcal{F}_{KeyGen'}$ during key generation. Furthermore, Protocol 4.10 securely implements the functionality $\mathcal{F}_{KG'-D}$ and in particular decryption under* $\mathbf{s}$ *is correct except with negligible probability.*

Proof. Firstly we will prove semantical security. To see that we will prove that given an element in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ the adversary cannot distinguish whether it is a key generated by $\mathcal{F}_{KeyGen'}$ or it is a random element. That would imply that since the adversary when decrypting cannot distinguish between a true encryption or a random element, which in its turn implies, as we have seen in Theorem 3.10, that the adversary cannot distinguish between an encryption of 0 and 1. Let us then see that if an adversary can distinguish between a key generated by $\mathcal{F}_{KeyGen'}$ from a random element in $\mathbb{Z}_q^n \times \mathbb{Z}_q$ then we can solve decision LWE. Let I be an instance of decision LWE, a vector we want to see whether it is random or part of $\mathcal{A}_{\mathbf{s},\chi}$. We then simulate to run $\mathcal{F}_{KeyGen'}$ so we get the noise contributions from the adversary and add them to the instance I before returning the vector to the adversary. Then if the instance I was uniformly distributed then the fake key will be uniformly distributed too and if I was in $\mathcal{A}_{\mathbf{s},\chi}$ then the key will follow the same distribution as if generated by $\mathcal{F}_{KeyGen'}$, allowing us to solve decision LWE. And as seen in Theorem 3.10 then if GAPSVP is hard in the worst case, then encryption under $\mathbf{s}$ is semantically secure.

Secondly we will prove correctness. Let $e + x$ be the reconstructed value after the decryption protocol, what we need to see is that e is small enough. Let e_i^j the contribution of player P_j to e_i. Then $e = \sum_{i=1}^m r_i e_i = \sum_{i=1}^m \sum_{j=1}^u r_i e_i^j$ with $|e_i^j| \leq \binom{u}{t}\sqrt[3]{q}$ (in the worst case scenario if all the shares in the NIVSS protocol were to be 0) so $|e| \leq um\binom{u}{t}\sqrt[3]{q}$. And we know from Theorem 5.11 that the decryption will be correct if the probability that $|e| > \sqrt{q}$ is negligible, which is fulfilled if $u\binom{u}{t} < \frac{\sqrt[6]{q}}{m}$ holds true, which it does.

Finally we will prove that the simulation still holds. The proof of the simulation works as in Theorem 5.15, however we need to see that indeed the interval where e resides is exponentially smaller than $[-\sqrt{q}, \sqrt{q}]$. Indeed, since $|e| \leq um\binom{u}{t}\sqrt[3]{q}$ and $u\binom{u}{t} < \frac{\sqrt[10]{q}}{m}$ we get that $|e| < \sqrt[10]{q} \cdot \sqrt[3]{q} = q^{\frac{13}{30}}$ which is exponentially smaller than $\sqrt{q}$. $\square$

This proves the protocol secure against active, static adversaries corrupting $t < \frac{u}{3}$ players. It is obvious from the security of the secret sharing schemes that the protocol will be secure against a passive adversary corrupting $t \leq u - 1$ players and a semi-honest adversary corrupting $t < \frac{u}{2}$ players.

5. RLWE threshold cryptosystem

Now that we have seen how the threshold decryption and key generation works on a cryptosystem based on Regev's cryptosystem, we will proceed to adapt it to a cryptosystem based on the one proposed by Lyubashevsky, Peikert and Regev in [12], which is very similar to Regev's cryptosystem but using the ideal version of LWE, called Ring Learning With Errors (RLWE).

Definition 5.1. We define $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ polynomial ring. Note that we identify any polynomial in $\langle \mathbf{a} \rangle \in R_q$ with points in the lattice $\mathcal{L}(\mathbf{A})$, with $\mathbf{A} = [\mathbf{a}, \mathbf{Fa}, \dots, \mathbf{F}^{n-1}\mathbf{a}]$ and $\mathbf{F}$ transformation matrix of $\mathbf{a}$, as we have seen in Definition 2.12 and the following remark.

We will need to define the parameters for the LPR cryptosystem:

- $m \in \mathbb{Z}_{>0}$.
- $n \in \mathbb{Z}_{>0},\ n = 2^m$, the security parameter of the cryptosystem.
- $q \in \mathbb{Z}_{>0}$ such that $q = 2^{\Theta(n)}$.
- $l \in \mathbb{Z}_{>0}$ the number of samples of encrypted messages.
- $\overline{\Psi}_{\frac{\xi}{q}}$ with $\xi = \alpha \left(\frac{nl}{log(nl)}\right)^{\frac{1}{4}}$, $\alpha > 0$ discrete Gaussian in $\mathbb{Z}_q$ with standard deviation bounded by ξ.

Definition 5.2. Given the parameters above, *LPR cryptosystem's* tuple is as follows:

- $\mathcal{P} = \{0,1\}^n$. We can encrypt strings of bits, not bit a bit as in Regev's cryptosystem. For every message m of n bits we want to encrypt, we associate it with the polynomial in R_q with such coefficients.
- $\mathcal{C} = R_q \times R_q$.
- This is a public encryption scheme, we have $\mathcal{K}_s = R_q$ and $\mathcal{K}_p = R_q \times R_q$.
- The secret key will be $s \in \mathcal{K}_s$ chosen following the distribution $\overline{\Psi}_{\frac{\xi}{q}}$ and the public key will be (a_E, b_E) where $a_E \in R_q$ is chosen uniformly at random and $b_E = a_E \cdot s + e$ with $e \in R_q$ is chosen following the distribution $\overline{\Psi}_{\frac{\xi}{q}}$. When an element in R_q is said to be chosen following the distribution $\overline{\Psi}_{\frac{\xi}{q}}$ it means that every coefficient follows such distribution in $\mathbb{Z}_q$.
- $\mathcal{E} = \{\mathrm{E}_k : k \in \mathcal{K}_p\}$ such that given a message $m \in \mathcal{P}$:

$$\begin{aligned} \mathrm{E}_k : \mathcal{P} &\to \mathcal{C} \\ m &\mapsto (u, v) \end{aligned}$$

  Where $(u, v) = (a_E \cdot r_E + e_u,\ b_E \cdot r_E + e_v + m \cdot \lfloor \frac{q}{2} \rfloor)$ with $k = (a_E, b_E)$ and $r_E, e_u, e_v \sim \overline{\Psi}_{\frac{\xi}{q}}$ and therefore their coefficients are small elements in R_q.
- $\mathcal{D} = \{\mathrm{D}_s : s \in \mathcal{K}_s\}$ such that given a cyphertext $(u, v) \in \mathcal{C}$:

$$\begin{aligned} \mathrm{D}_s : \mathcal{C} &\to \mathcal{P} \\ (u, v) &\mapsto m \end{aligned}$$

Where we will recover every bit of m by rounding every coefficient of $v - s \cdot u$ to 0 or $\lfloor \frac{q}{2} \rfloor \pmod q$ and then mapping 0 to 0 and $\lfloor \frac{q}{2} \rfloor$ to 1.

Remark. Note that the product here is as a product of polynomials in R_q. If we want to translate it to the lattice equivalent it will be given by the product of the anticyclic matrix generated by one vector with the other. In other words, let $a(x), b(x) \in R_q$, then:

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

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

5.1 Threshold decryption

Now we can present the threshold decryption protocol for RLWE encryption. To do so we will need to give a couple more parameters. Let $\Phi^R_\cdot(\cdot)$ a pseudorandom function that given a key in $\mathbb{Z}_q$ and a cyphertext outputs a pseudorandom value in I^n, where $I = [a, b]$ is an interval.

Protocol 5.3. The *decryption protocol* works as follows:

1. A Trusted Third Party generates the keys $K_H \in \mathbb{Z}_q$ for every subset H of players of size t and distributes them according to PRSS (Technique 4.8). It also generates the secret key $s \sim \overline{\Psi}_{\frac{\xi}{q}}$ and the public key $(a_E, b_E) \in R_q \times R_q$ as in LPR cryptosystem. Then the TTP sends to the players (a_E, b_E) and Shamir shares of s. We call s^j the Shamir share of s of player P_j, understood as a Shamir share on the vector of coefficients of s.

2. Client receives cyphertext (u, v), decides which $t + 1$ players will decypher the text and sends them (u, v) and a list of the players participating in the decryption.

3. Each player P_j computes $\tilde{e}^j = v - s^j \cdot u$ that is a Shamir share of $\tilde{e} = e \cdot r_E + e_v - s \cdot e_u + \lfloor \frac{q}{2} \rfloor \cdot m$ with $e, r_E, e_v, s, e_u \sim \overline{\Psi}_{\frac{\xi}{q}}$ because of Proposition 4.4 and Proposition 4.5.

4. Each player P_j computes its additive share of $\mathbf{x} := \sum_H \Phi^R_{K_H}(c)$ in the following way: in order, the additive share $\mathbf{x}^j$ is the sum of all $\Phi^R_{K_H}(c)$ no player before has, but P_j does. Then P_j converts this share to $\tilde{\mathbf{x}}^j$ Shamir share of $\mathbf{x}$ using Technique 4.9.

5. Each player P_j computes $\tilde{\mathbf{x}}^j + \tilde{e}^j$ Shamir share of the vector of coefficients of $\mathbf{x} + \tilde{e} \in R_q$, where $\mathbf{x} + \tilde{e} \in R_q$ is understood as the polynomial in R_q with vector of coefficients $\mathbf{x} + \tilde{e}$.

6. Client reconstructs $\mathbf{x} + \tilde{e}$ using Lagrange interpolation, then returns 0 if $\mathbf{x} + \tilde{e}$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ and returns 1 otherwise for every bit.

Before proving correctness and security we will state a result we will use in the proof taken from [13].

Lemma 5.4. *[13] Let $\overline{\Psi}_{\frac{\sigma}{q}}$ be a discrete Gaussian with standard deviation bounded by σ. Then $\forall c > 0$:*

$$\Pr(|\overline{\Psi}_{\frac{\sigma}{q}}| > c\sigma) < 2e^{-\frac{c^2}{2}}$$

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

Proof. Let $\hat{e} = e \cdot r_E + e_v - s \cdot e_u$. What we want to see is that $\Pr((|\mathbf{x} + \hat{e}|)_i > \frac{q}{4})$ is negligible $\forall i$, where $(.)_i$ notes the coefficient i on the polynomial.

Since the product in R_q is done through the anticyclic matrix, we know we have that:

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

and therefore, since $e, r_E, e_v, s, e_u \sim \overline{\Psi}_\xi$, if we have that $\Pr(|\overline{\Psi}_\xi| > k) < \lambda$ for some k,then $\Pr((|\hat{e}|)_i > 2nk^2 + k) < \lambda$.

From Lemma 6.4 we know that $\Pr\left(|\overline{\Psi}_\xi| > c\xi\right) \leq 2e^{-\frac{c^2}{2}}$, so we have that $k = c\xi$. And given that $\xi, c > 1$ (if we take $\alpha > 1$) we have that $k > 1$ and therefore $2nk^2 + k < k^2(2n+1)$, so:

$$\Pr\left((|\hat{e}|)_i > k^2(2n+1)\right) < \Pr\left((|\hat{e}|)_i > 2nk^2 + k\right) < 2e^{-\frac{c^2}{2}}$$

We also know that by construction:

$$|x_i| \leq \binom{u}{t}(c\xi)^2(2n+1)q^d$$

And by definition:

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

Therefore giving us:

$$\Pr\left((|\mathbf{x} + \hat{e}|)_i > \frac{q}{4}\right) < \Pr\left((|\mathbf{x} + \hat{e}|)_i > k^2(2n+1)\left(\binom{u}{t}q^d+1\right)\right) < 2e^{-\frac{c^2}{2}}$$

And given that $c = \Omega(\sqrt{n})$ the probability is negligible. □

Now we will prove security. We know that the difficulty of distinguishing between two cyphertexts is directly related to $\mathrm{RLWE}_{q,n,\Psi_{\frac{\xi}{q}}}$, and given our set of parameters, there exists a reduction from $\mathrm{RLWE}_{q,n,\overline{\Psi}_{\frac{\xi}{q}}}$ to $\mathrm{K-DGS}_\gamma$ in [17], where $\mathrm{K-DGS}_\gamma$ is an ideal lattice problem usually hard to solve and:

$$\gamma = \max\left\{\eta(I)\frac{\sqrt{2}}{\xi}\left(\frac{nl}{log(nl)}\right)^{\frac{1}{4}}\Omega(\sqrt{log(n)}),\ \frac{\sqrt{2n}}{\lambda_1(I^*)}\right\}$$

This means that breaking the security of the cryptosystem is as hard as solving $\mathrm{K-DGS}_\gamma$. Therefore we only need to prove that no information is leaked when doing the threshold decryption (as in the previous chapter).

As the chapter before, we now need to prove that no relevant information is leaked while sharing the decryption. And as before we will define three functionalities to help with the proof.

Functionality 5.6. We define the functionality $\mathcal{F}_{RKeyGen}$ that works as following:

1. When given"start" by all players, $\mathcal{F}_{RKeyGen}$ chooses a secret key $s \in R_q$ uniformly at random and constructs the public key $(a_E, b_E) \in R_q \times R_q$, as in LPR cryptosystem.
2. For each subset H of size t of players $\mathcal{F}_{RKeyGen}$ chooses key $K_H \in \mathbb{Z}_q$ uniformly at random.
3. $\mathcal{F}_{RKeyGen}$ receives from the adversary, for each corrupted player P_j, the share $s^j \in R_q$.
4. Using Lagrange interpolation and the shares s^j received from the adversary, $\mathcal{F}_{RKeyGen}$ computes a polynomial $f(x_1, \ldots, x_n) = (f_1(x_1), \ldots, f_n(x_n))$, $f_i(x_i) \in \mathbb{Z}_q[x]$ of degree t that goes through the shares and such that $f(0) = (s_1, \ldots, s_n)$ with $(s_1, \ldots, s_n)$ is the vector of coefficients of s, and then computes the Shamir shares for every player.
5. $\mathcal{F}_{RKeyGen}$ sends privately to every player P_j their share s^j of the secret key and K_H for all H such that $P_j \notin H$.
6. $\mathcal{F}_{RKeyGen}$ sends public key to all players and adversary.

Functionality 5.7. We define the functionality $\mathcal{F}_{RKeyGen-and-Decrypt}$ or $\mathcal{F}_{RKG-D}$ that implements the same steps as Protocol 6.3:

1. When receiving "start" of all players send "start" to $\mathcal{F}_{RKeyGen}$, receives all the shares of s and all the K_H and forwards it to every player.
2. When receiving "decrypt (u, v)" from the client, send "decrypt (u, v)" to all players participating in the decryption and the adversary, and then reconstruct the decrypted message m.
3. In the next round, send m and the shares of every player to the client and adversary.

Functionality 5.8. We define the new functionality $\mathcal{F}_{SimRDecrypt}$ that works as follows, assuming B the set of corrupted players:

1. When given "start" by all players in B $\mathcal{F}_{SimRDecrypt}$ gives "start" to $\mathcal{F}_{RKeyGen}$ and receives all the shares of s and the keys K_H.
2. $\mathcal{F}_{SimRDecrypt}$ sends the shares of s and their respective K_H to the players in B.
3. When receiving "decrypt (u, v)" from the client, $\mathcal{F}_{SimRDecrypt}$ sends "decrypt (u, v)" to all players in B, receives their shares of the decryption and decrypts the message m with these shares, the shares of the secret key of the honest players and the decryption protocol.
4. For the honest players $\mathcal{F}_{SimRDecrypt}$ must simulate shares to broadcast in the next round, so for every K_H the adversary does not know (i.e. $B \subseteq H$) $\mathcal{F}_{SimRDecrypt}$ generates a random vector of integers $r_H \in [-\frac{1}{2}(c\xi)^2(2n+1)q^d, \ \frac{1}{2}(c\xi)^2(2n+1)q^d]^n$, and we define $y \in R_q$ as the polynomial such that its coefficients are $\sum_{B \not\subseteq H} \Phi^R_{K_H}(c) + \sum_{B \subseteq H} r_H$. Now using Lagrange interpolation $\mathcal{F}_{SimRDecrypt}$ computes shares of $y + m\lfloor \frac{q}{2} \rfloor$ using Lagrange interpolation, the shares of the decryption received by the players in B in step 3 and m calculated in step 3.
5. In the next round send the message m corresponding to cyphertext (u, v) and the shares of $y + m\lfloor \frac{q}{2} \rfloor$.

Theorem 5.9. *When given access to the functionality $\mathcal{F}_{RKeyGen}$ and assuming that $\Phi^R_\cdot(\cdot)$ is a pseudo-random function, the decryption protocol is secure. The adversary is assumed to be passive and static, corrupting up to $t = u - 1$ players.*

Proof. This proof is analogous to the proof of theorem 4.15, changing the functionalities for their analogous in RLWE (adding an R in front), $\Phi_\cdot(\cdot)$ for $\Phi^R_\cdot(\cdot)$ and the length of the intervals, from $2\sqrt{q}$ to $2(c\xi)^2(2n+1)q^d$ and from $2\sqrt[3]{q}$ to $2(c\xi)^2(2n+1)$ which is obviously exponentially smaller than the first since $0 < d < 1$. □

As in the chapter before, we have only proven security against a passive adversary corrupting $t \leq u-1$ players, but it is very easy 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.

5.2 Distributed key generation

Again, as before, we will proceed to define the protocol for distributed key generation based on the LPR cryptosystem, for which we will need to assume private communication channels between players, and then define two functionalities we will use to prove its security. However, we will also need to prove that adding the distributed key generation does not render the decryption protocol insecure, so we will define another functionality to prove the protocol's security against an active, static adversary corrupting $t < \frac{u}{3}$ players as in the previous chapter.

Protocol 5.10. The *key generation protocol* works as follows, given $\phi^R_\cdot(\cdot)$ a pseudorandom function that given a seed and an integer returns a pseudorandom value in $[-(c\xi)^2(2n+1)q^{\tilde{d}},\ (c\xi)^2(2n+1)q^{\tilde{d}}]^n$ with the same parameters as in the previous subsection and having $0 < \tilde{d} < d$:

1. For the secret key $s \in R_q$, each player P_j chooses its contribution $(s^j_1, \dots, s^j_1)$ with $s^j_i \sim \overline{\Psi}_{\frac{\xi}{q\sqrt{n}}}$. Then they act as the dealer in a NIVSS to share every s^j_i to all players. All players verify the value broadcast when doing the NIVSS $s^j_i - \sum_H \phi^R_{K_H}(0)$ is in the interval $[-\binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}, \binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}]$. Now all players have shares of every s^j_i and by their lineality also of $s_i = \sum_j s^j_i$. Then s is the polynomial in R_q with coefficients $(s_1, \dots, s_n)$.

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. Then the players will have, by adding all the shares received by other players Shamir shares of $K_H = \sum_j K_{H_j}$ because of Proposition 4.4. 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 $e \in R_q$ proceed identically to when generating s.

4. For $a_E \in R_q$ every player P_j chooses its share $(a^j_{E,1}, \dots, a^j_{E,n})$ randomly in Z^n_q and does a Shamir share of it. Then all players send to all players their share on all the $(a^j_{E,1}, \dots, a^j_{E,n})$ so every player can recover (by adding the shares) $\left(\sum_j a^j_{E,1}, \dots, \sum_j a^j_{E,n}\right)$. a_E will be the polynomial in R_q with coefficients $\left(\sum_j a^j_{E,1}, \dots, \sum_j a^j_{E,n}\right)$.

5. Every player computes locally their Shamir shares on $b_E = a_E \cdot s + e$ by performing these same operations with the shares they have on s and e (having previously converted the shares to Shamir using Technique 4.9).

Remark. All the remarks previously made about the NIVSS protocol still hold true this time, but now we would also like to remark that taking s and e as we have, indeed follows the parameters of the previous subsection, since:

$$\sum_{i=1}^{n} \overline{\Psi}_{\frac{\xi}{q\sqrt{n}}} \sim \sum_{i=1}^{n} \lfloor q\Psi_{\frac{\xi}{q\sqrt{n}}} \rceil \pmod q \sim \lfloor q \sum_{i=1}^{n} \Psi_{\frac{\xi}{q\sqrt{n}}} \rceil \pmod q \sim \lfloor q\Psi_{\sqrt{n}\frac{\xi}{q\sqrt{n}}} \rceil \pmod q \sim \overline{\Psi}_{\frac{\xi}{q}}$$

Where we have used that Ψ_β is the reduction to $\mathbb{T}$ of a $\mathrm{N}\left(0, \frac{\beta}{2\pi}\right)$ and therefore given n independent $\mathrm{N}\left(0, \frac{\beta}{2\pi}\right)$ we have:

$$\sum_{i=1}^{n} \mathrm{N}(0, \frac{\beta}{2\pi}) \sim \mathrm{N}\left(0, \sqrt{\sum_{i=1}^{n}\left(\frac{\beta}{2\pi}\right)^2}\right) \sim \mathrm{N}\left(0, \sqrt{n}\frac{\beta}{2\pi}\right)$$

We will now proceed to define the three functionalities needed to prove security of the distributed key generation scheme.

Functionality 5.11. We define the functionality $\mathcal{F}_{RRand}$, which has four different commands and works as follows:

- **"Shared value from D"** where D is a player. The player D sends to $\mathcal{F}_{Rand}$ a value chosen uniformly at random in $\mathbb{Z}_q$ and the adversary sends to $\mathcal{F}_{Rand}$ a set of shares for the corrupt players. Then the functionality uses Lagrange interpolation to construct consistent Shamir shares of the value sent by D for the honest players and finally sends the shares to each player.
- **"Random shared value"**. $\mathcal{F}_{Rand}$ calls "Shared value from P_i" for all players, and then each player locally adds the shares of all the random numbers, giving shares of $s = \sum_i s_i$.
- **"Random value to B"** where B is a set of players. $\mathcal{F}_{Rand}$ calls "Random shared value" and all the players send their shares to the players in B.
- **"Constrained value from D"** where D is a player. Player D performs a NIVSS of its secret s, with the functionality verifying afterwards whether the broadcast value is in $[-\binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}, \binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}]$, and returning "Fail" if it is not.

Functionality 5.12. Given access to the functionality $\mathcal{F}_{RRand}$ we define the functionality $\mathcal{F}_{RKeyGen'}$ that works as follows:

1. When receiving "start" from all honest players receive the broadcast value from the NIVSS done for each s_i^j from player P_j and call "Constrained value from P_j".
2. For each subset H of t players call "Random value to A" where A is the complement of H to generate the keys K_H used for the PRSS in the decryption.
3. Call "Random value to P" with P the set of all players n times to generate the coefficients of $a_E \in R_q$.

4. Receive the broadcast value from the NIVSS done for each e_i^j from player P_j and call "Constrained value from P_j".
5. Broadcast the public key given by $(a_E,\ b_E = a_E \cdot s + e)$ and every player's share of b_i.

Functionality 5.13. We define the functionality $\mathcal{F}_{RSimGen}$ as follows:

1. Receive the shares of the corrupted players of $s_{i,j},\ i = 1, \dots, n$ and for every player P_j from the adversary and generate a random vector $(s_1, \dots, s_n)$ with $s_i \sim \Psi_{\frac{\xi}{q}}$. Then send the shares back to the adversary.
2. Receive the shares of the corrupted players of K_{H_j} for every player P_j and every subset H of t players from the adversary, and generate a random number K_H for every H, Then send the shares back to the adversary.
3. Receive the shares of the corrupted players of the coefficients of a_E for every player P_j from the adversary and generate randomly $a_E \in R_q$. Then send the shares back to the adversary.
4. To simulate the sharing of the values e_i^j receive the broadcast values for every corrupted player and return "Fail" if they are not in the interval $[-\binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}, \binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}]$. Then simulate the NIVSS for every honest player taking random numbers as keys and a random noise contribution e_i^j in $[-\binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}, \binom{u}{t}(c\xi)^2(2n+1)q^{\tilde{d}}]$.
5. Reconstruct the public key $(a_E,\ b_E = a_E \cdot s + e)$ and since $\mathcal{F}_{RSimGen}$ knows the shares of the corrupted players, construct shares of b_E consistent with the adversary for all honest players with Lagrange interpolation.
6. Broadcast (a_E, b_E) and the shares of the honest players of b_E.

Theorem 5.14. *The Key Generation protocol securely implements the functionality $\mathcal{F}_{RKeyGen'}$. The adversary is assumed to be active and static, corrupting up to $t < \frac{u}{3}$ of the players.*

Proof. The proof is analogous to Theorem 4.21. □

Now we only need to prove that with the distributed key generation the decryption protocol is still secure. We will define another functionality and then prove the security.

Functionality 5.15. We define the functionality $\mathcal{F}_{RKeyGen'-and-Decrypt}$ or $\mathcal{F}_{RKG'-D}$ as Functionality 5.7, $\mathcal{F}_{RKG-D}$ but using $\mathcal{F}_{RKeyGen'}$ instead of $\mathcal{F}_{RKeyGen}$.

Theorem 5.16. *Assuming we use $\mathcal{F}_{RKeyGen'}$ to generate the public key (a_E, b_E) and the secret key s, and the conditions of Theorem 5.5 are fulfilled. If $\mathrm{K}-\mathrm{DGS}_\gamma$ is hard, then encryption under s is semantically secure against any active polynomial time adversary corrupting $t < \frac{u}{3}$ players that interacts with $\mathcal{F}_{RKeyGen'}$ during key generation. Furthermore, Protocol 5.10 securely implements the functionality $\mathcal{F}_{RKG'-D}$ and in particular decryption under s is correct except with negligible probability.*

Proof. The proof is analogous to the proof of Theorem 4.23 with the decision RLWE problem, adding the bounds for $\frac{q}{4}$ and $(|x + \hat{e}|_i)$ from Theorem 5.5. □

This proves the protocol secure against active, static adversaries corrupting $t < \frac{u}{3}$ players. It is obvious from the security of the secret sharing schemes that the protocol will be secure against a passive adversary corrupting $t \leq u - 1$ players and a semi-honest adversary corrupting $t < \frac{u}{2}$ players.

6. Conclusion

As seen in the previous section, the objective of developing a new threshold cryptosystem with distributed key generation based on RLWE has been accomplished, since we have defined it and proven correctness and security for certain choices of parameters. However, to make the cryptosystem somewhat operative we still need to have in mind two things.

Firstly and most importantly, the security of this cryptosystem has only been proven to be asymptotically as difficult as $\mathsf{K}-\mathsf{DGS}_\gamma$, with γ depending on the choice of parameters. Therefore, to have security to be as hard as one desires, we should fine-tune all the parameters of the cryptosystem $(n, q, l, c, d$ and $\tilde{d})$ to obtain such security.

Secondly, and a little bit more complicated to solve, our cryptosystem suffers in a similar way as Bendlin and Damgård's, in the sense that the shared generation of the value x, in our case a vector, through PRSS is highly inefficient due to the amount of keys necessary (as it has been stated before). This could be solved by using another secret sharing scheme, or even instead of using the statistical difference measured with the traditional metric in $\mathbb{R}$, we could try to use another metric or a *divergence*, a notion weaker than distance sometimes used in cryptography proofs.

All this, however, is left as future work and has not been explored in this bachelor's degree thesis.

7. Bibliography

References

[1] AJTAI, Miklós. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998. p. 10-19.

[2] BENDLIN, Rikke; DAMGÅRD, Ivan. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *Theory of Cryptography Conference*. Springer, Berlin, Heidelberg, 2010, p. 201-218.

[3] CATALANO, Dario. Efficient distributed computation modulo a shared secret. In *Contemporary Cryptology*. Birkhäuser Basel, 2005. p. 1-39.

[4] CHAUM, David; CRÉPEAU, Claude; DAMGÅRD, Ivan. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988. p. 11-19.

[5] CRAMER, Ronald; DAMGÅRD, Ivan; ISHAI, Yuval. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography Conference*. Springer, Berlin, Heidelberg, 2005. p. 342-362.

[6] GOLDWASSER, Shafi; MICALI, Silvio. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM Symposium: Theory of Computing*, 1982, p. 365-377.

[7] GOLDWASSER, Shafi; MICALI, Silvio. Probabilistic encryption. Journal of computer and system sciences, 1984, vol. 28, no 2, p. 270-299.

[8] GAMA, Nicolas; NGUYEN, Phong Q. Finding short lattice vectors within mordell's inequality. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008. p. 207-216.

[9] GOLDREICH, Oded. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

[10] KATZ, Jonathan; LINDELL, Yehuda. *Introduction to modern cryptography*. CRC press, 2014.

[11] LENSTRA, Hendrik Willem; LENSTRA, Arjen Klaas; LOVÁSZ, László. Factoring polynomials with rational coeficients. 1982.

[12] LYUBASHEVSKY, Vadim; PEIKERT, Chris; REGEV, Oded. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 2013, vol. 60, no 6, p. 1-35.

[13] LYUBASHEVSKY, Vadim. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2012. p. 738-755.

[14] MENEZES, Alfred J., et al. *Handbook of applied cryptography*. CRC press, 1996.

[15] PEIKERT, Chris. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, p. 333-342.

[16] PEIKERT, Chris, et al. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 2016, vol. 10, no 4, p. 283-424.

[17] PEIKERT, Chris; REGEV, Oded; STEPHENS-DAVIDOWITZ, Noah. Pseudorandomness of ring-LWE for any ring and modulus. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017. p. 461-473.

[18] REGEV, Oded. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 2009, vol. 56, no 6, p. 1-40.

[19] SCHNORR, Claus-Peter; EUCHNER, Martin. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 1994, vol.66, no 1-3, p. 181-199.

[20] SHAMIR, Adi. How to share a secret. *Communications of the ACM*, 1979, vol. 22, no 11, p. 612-613.

[21] SHOR, Peter W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 1999, vol. 41, no 2, p. 303-332.

[22] YI, Xun; PAULET, Russell; BERTINO, Elisa. Private information retrieval. *Synthesis Lectures on Information Security, Privacy, and Trust*, 2013, vol. 4, no 2, p. 1-114.