Post-Quantum Cryptography: Lattice-based encryption

Author: Eduard Sanou

Advisor: Javier Herranz,
Mathematics Applied to Cryptography (MAK)

Tutor: Fernando Martínez,
Department of Mathematics

Master in Innovation and Research in Informatics
Advanced Computing specialization
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

July 8, 2016

Abstract

Post-Quantum Cryptography has gained attention in the recent years from the research community due to the disastrous consequences that efficient quantum computers would have on current public-key cryptosystems. In this work we explore the main aspects of lattice theory, provide definitions for some of the most relevant lattice problems and explain how lattice theory can be used as a framework to build post-quantum cryptosystems. In particular, we detail and analyze a public-key encryption scheme that relies on the hardness of the Ring Learning With Errors (Ring-LWE) problem. We propose sets of parameters and provide an efficient implementation of the scheme written in C++ which allows us to provide benchmarking results. Finally, we propose an adaptation of the e-voting scheme employed by Scytl (also described in this work) to make use of the previously described encryption scheme, along with the necessary modifications to make it a post-quantum e-voting scheme.

Acknowledgements

I would like to thank Javier Herranz for his help, advice and interesting discussions on schemes proposals as well as the reviews he did to earlier versions of this thesis. I would also like to thank Paz Morillo for her help and reviews of earlier versions of this thesis as well as for the fun times spent learning lattice theory together.

Contents

1 Introduction

For the past three decades, public key cryptography has become an essential tool to protect the digital communications in our infrastructures. It is used in personal communications, business networks and governments to guarantee authenticity and confidentiality in the exchanged data.

The main cryptographic functionalities used nowadays are public key encryption, digital signatures and key exchange schemes. These primitives are primarily implemented using three main constructions: RSA (Rivest-Shamir-Adleman), Diffie-Hellman key exchange and elliptic curves, whose security relies on the difficulty of solving certain number theoretic problems like integer factoring and the discrete logarithm.

In 1994, Peter Shor proposed an algorithm designed to run in a quantum computer that could solve the integer factorization problem in polynomial time. Additionally, this algorithm could also be used to solve the discrete logarithm problem in polynomial time, rendering most of the currently used public key cryptography schemes insecure.

This situation has led to a substantial amount of research on quantum computers (machines that take advantage of quantum mechanical phenomena to solve problems that are hard on classical computers). It is still not clear if efficient and scalable quantum computers would ever be created, but if this happened to be the case, the confidentiality and integrity of digital communications would be seriously compromised because current public-key cryptosystems would be broken.

To address this issue, the scientific community has been working on two main solutions: *quantum cryptography* and *post-quantum cryptography*.

Quantum cryptography consists in building cryptographic schemes that take advantage of quantum mechanic phenomena like the *BB84* quantum key distribution scheme [BB84]. It has been argued that this solution would be quite expensive due to the hardware requirements as well as quite impractical at scale, due to restrictions imposed by quantum physics [Ber09].

The other solution, which we study in this thesis, is post-quantum cryptography and consists in building cryptographic schemes that can run on classical computers and work with existing networks that are able to resist attacks on both classical and quantum computers[1].

[1] In practice, post-quantum resistance can't be proved. We can only rely on the fact that after many years studying specific computational problems, no one has found an efficient algorithm to solve them using a quantum computer.

To rise awareness of the importance of finding post-quantum replacements for current public-key cryptosystems, we refer the reader to the announcement that the National Security Agency of the USA (NSA) published in August 2015 about its plan to transition to a new cipher suite that is resistant to quantum attacks in the not too distant future; advising institutions that haven't made the transition to the current recommended algorithms to wait for the upcoming quantum resistant algorithms[2].

Currently there are four main proposals for post-quantum public-key cryptosystems: hash-based cryptography, code-based cryptography, lattice-based cryptography and multivariate-quadratic-equations cryptography. In this thesis we will only explain lattice-based cryptography. To learn about the other problems, the reader can find introductions for each one in the Post-Quantum Cryptography book [BBD09].

To avoid scaring the reader, we want to clarify that currently used secret-key cryptography (like AES) is quantum resistant, because it doesn't rely on integer factoring nor solving the discrete logarithm problem. Nevertheless, in 1996, Lov Grover proposed a quantum algorithm that can be used to break secret-key algorithms by brute-forcing a key space of $O(N^{1/2})$, where N is the length of the key. This means that to maintain the same level of security in secret-key cryptography in a post-quantum situation, the number of bits in the key should be doubled.

In this work we will focus on a post-quantum asymmetric encryption scheme. We argue that having post-quantum encryption schemes is more urgent than having post-quantum signature schemes. As long as efficient quantum computers aren't constructed, both pre-quantum encryption and signature schemes will be secure: no authorized entity will be able to compromise the confidentiality and integrity of the encrypted messages, and no entity will be able to forge signatures and so they will be verified without tampering. Once efficient quantum computers appear, we could easily stop trusting pre-quantum signatures, but any encrypted message that was intercepted at any time could be decrypted, thus breaking confidentiality. This means that we should be using post-quantum encryption algorithms long before efficient quantum computers appear, whereas there is less hurry for post-quantum signatures.

As mentioned before, we focus on lattice-based cryptography in this work. This field has become popular in the recent years not only because of the potential cryptographic applications in a post-quantum world, but also because of the

[2] https://web.archive.org/web/20160420221628/https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml/post

average case to worst case complexity reductions that some lattice problems have, which is something not found in previous cryptosystems.

Finally, we study how the current e-voting scheme employed by the electronic voting company Scytl works and we take the challenge of proposing an adaptation on the required parts to make it a post-quantum e-voting scheme.

Outline The remainder of this article is organized as follows. In Section 2 we give some basic definitions of cryptography concepts needed to understand the schemes and properties discussed in this work. In Section 3 we give the definitions of the main lattice concepts and we provide a novel lemma followed by a proof to help understand the geometry of lattices defined in a particular structure commonly used in cryptographic schemes. We also give the definitions of some relevant hard lattice problems. In Section 4 we focus on the Learning With Errors problem, giving its definition and detailing the process of cryptanalysis. In Section 5 we introduce the concept of ring lattices, detailing their advantages as well as adapting the previously defined Learning With Errors problem to these kind of lattices. Section 6 gives an introduction of the current state of two main public-key cryptography primitives in the lattice world. In Section 7 we focus on a public-key encryption scheme based on ring lattices, detailing it and analyzing its theoretical properties. In Section 8 we discuss our implementation of the previous scheme with specific parameter proposals, followed by an extensive analysis and benchmarking showing our practical results. Finally, in Section 9 we give an introduction to e-voting, followed by a detailed description of the current e-voting scheme used by Scytl which we later adapt as necessary to make it quantum resistant. Finally, Section 10 gives the conclusions of this work.

2 Basic cryptography concepts

2.1 Public-key encryption

An asymmetric, or public-key, encryption schemes is a triple of randomized algorithms having the following interfaces

- The *key generator* (given the security parameter) outputs a public key and a secret key.
- The *encryption* algorithm takes a public key and a valid message, and outputs a ciphertext.
- The *decryption* algorithm takes a secret key and a ciphertext, and outputs a message or a distinguished "failure" symbol.

The scheme is said to be *correct* if generating a public/secret key pair, then encrypting a message using the public key, then decrypting the resulting ciphertext using the secret key yields the original message (maybe with all but negligible probability).

2.2 Security

There are two main notions of security:

- Semantic Security, or more formally, indistinguishablity under chosen-plaintext attack (IND-CPA). It considers the following experiment, which is parametrized by a bit $b \in \{0, 1\}$: generate a public/secret key pair, and give the public key to the attacker, who must reply with two valid messages m_0, m_1; encrypt m_b using the public key and give the resulting "challenge ciphertext" to the attacker. The scheme is said to be semantically (or IND-CPA) secure if it is infeasible for an attacker to distinguish between the two cases $b = 0$ and $b = 1$, i.e., its probabilities of accepting in the two cases should differ by only a negligible amount.
- Active security, or more formally, indistinguishablity under chosen-ciphertext attack (IND-CCA). It augments the above experiment by giving the attacker access to a decryption oracle, i.e., one that runs the decryption algorithm (with the secret key) on any ciphertext the attacker may query, except for

the challenge ciphertext. (This restriction is of course necessary because otherwise the attacker could just request to decrypt the challenge ciphertext and thereby learn the value of b). The scheme is said to be actively (or IND-CCA) secure if it is infeasible for an attacker to distinguish between the two cases $b = 0$ and $b = 1$.

2.3 Homomorphic encryption

Homomorphic encryption is a special property of some encryption schemes such that certain operations can be carried out on some ciphertexts, generating a new ciphertext that when decrypted, the obtained plaintext corresponds to the result of operations performed on the plaintexts underlying the original ciphertexts. For instance, homomorphic addition would allow us to add two ciphertexts, where the decryption of the result would match the addition of the corresponding plaintexts of the original two ciphertexts.

This can be a desirable feature in specific systems, as it allows an untrusted party to perform computations on the encrypted data (without the need of revealing the content of such data), and so it ensures confidentiality of processed data. For example, in an e-voting context it could be desirable to sum encrypted ballots into one encrypted result so that only one decryption is performed (whose result will match the addition of all the ballots' plaintext) thus making it harder to link the ballots content to each voter. A more detailed analysis of this example can be found in Section 9

Homomorphic encryption schemes are malleable by design and so don't provide active security (see Section 2.1): an attacker could apply a known operation to the "challenge ciphertext", send the result to the oracle, and apply the inverse of the operation to reveal the plaintext selected by the challenger.

A partial homomorphic cryptosystem only allows a specific kind of operation on the encrypted data. There are several cryptosystems that have this feature such as *RSA* and *ElGamal* which are multiplicative homomorphic for some moduli. Other schemes allow addition operations (see Section 7.4 for a detailed example).

2.3.1 Fully homomorphic encryption

A cryptosystem that supports computing arbitrary functions (practically achieved by supporting addition and multiplication) on the ciphertexts is known as fully homomorphic encryption (FHE) and provides much more flexibility. These schemes allow the construction of arbitrary programs that can take as input encrypted data

and produce an encryption of the result. This theoretical problem had been proposed more than 30 years ago, without knowing if it was even possible to solve. In 2009 the first fully homomorphic encryption scheme was presented by Craig Gentry using lattice-based cryptography [Gen09], which was soon followed by efficiency improvements by Marten van Dijk, Shai Halevi, Vinod Vaikuntanathan and Craig Gentry himself [DGHV09]. After that, many advances have been made on the topic (with some new schemes not based on lattices) that keep improving the efficiency of the operations on the encrypted data, making it a very active research topic nowadays.

Unfortunately, despite the many advances, the best implementations are still quite inefficient: using the fastest implementation, in [GHS12] the authors achieve an homomorphic evaluation of a single AES-128 encryption operation in 4 minutes using a modern computer, with the possible parallelization improvement using CPU optimizations to evaluate the encryption of 120 AES-128 blocks in the same amount of time (leading to an amortized rate of 2 seconds per block). As a comparison, a modern computer is capable of encrypting a single normal AES-128 block in 20 μs, that is, 8 orders of magnitude faster.

This thesis only provides an introduction to lattice based cryptography, and as such, fully homomorphic schemes are not included in the contents.

3 Basic lattice concepts

In the following sections, standard notation will be used. Scalar integers will be represented in the lowercase roman alphabet (such as k and n). Vectors will be represented by boldface lowercase roman letters (such as $\mathbf{v}$ and $\mathbf{w}$). Matrices will be represented in boldface uppercase roman letters (such as $\mathbf{B}$ and $\mathbf{H}$).

Most of the definitions in this section are taken from the publications [MR09] and [Pei16].

3.1 Lattice

Definition 1 (Lattice)**.** *A lattice is a set of points in an n-dimensional space with a periodic structure. More formally, an n-dimensional lattice $\mathcal{L}$ is any subset of $\mathbb{R}^n$ that is both:*

1. *an **additive subgroup**: $\mathbf{0} \in \mathcal{L}$, and $-\mathbf{x}, \mathbf{x}+\mathbf{y} \in \mathcal{L}$ for every $\mathbf{x}, \mathbf{y} \in \mathcal{L}$; and*
2. ***discrete**: every $\mathbf{x} \in \mathcal{L}$ has a neighborhood in $\mathbb{R}^n$ in which $\mathbf{x}$ is the only lattice point.*

Given k-linearly independent vectors $\mathbf{b_1}, \ldots \mathbf{b_k} \in \mathbb{R}^n$, the rank k lattice generated by them is the set of vectors:

$$\mathcal{L}(\mathbf{b_1}, \ldots \mathbf{b_n}) = \left\{ \sum_{i=1}^{k} z_i \mathbf{b_i} : z_i \in \mathbb{Z} \right\} = \{\mathbf{Bz} : \mathbf{z} \in \mathbb{Z}^n\}$$

Where $\mathbf{Bz}$ is the usual matrix-vector multiplication. Here we say that $\mathbf{b_1}, \ldots \mathbf{b_k}$ is a basis for the lattice $\mathcal{L} = \mathcal{L}(\mathbf{b_1}, \ldots \mathbf{b_k})$. We denote the basis matrix for $\mathbf{b_1}, \ldots \mathbf{b_k}$ as $\mathbf{B} = (\mathbf{b_1}, \ldots \mathbf{b_k})$, i.e., the matrix whose columns are $\mathbf{b_1}, \ldots \mathbf{b_k}$. For convenience, we will use the notation $\mathcal{L}(\mathbf{B}) = \mathbf{B} \cdot \mathbb{Z}^k$, where we note that $\mathcal{L}(\mathbf{B}) = \mathcal{L}$. See Figure 1 for an example of a two dimensional lattice.

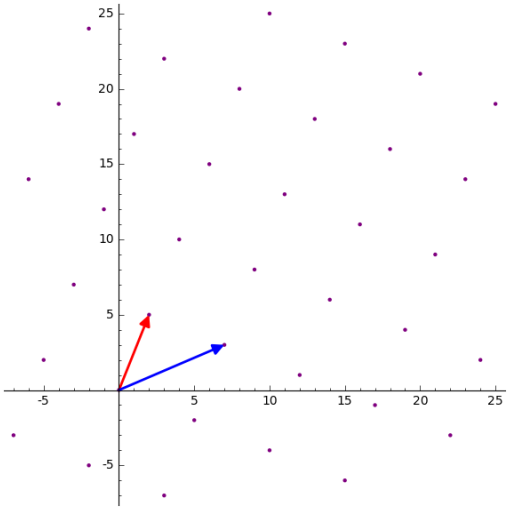


Figure 1: Two dimensional lattice generated from the basis $\mathbf{b}_1 = [2, 5], \mathbf{b}_2 = [7, 3]$

It is not difficult to see that if $\mathbf{U}$ is a unimodular matrix (i.e., an integer square matrix with determinant ± 1), the bases $\mathbf{B}$ and $\mathbf{BU}$ generate the same lattice. (In fact, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if there exists a unimodular matrix $\mathbf{U}$ such that $\mathbf{B}' = \mathbf{BU}$). In particular, any lattice admits an infinite number of bases, and this fact is at the core of many cryptographic applications.

3.2 Rank

As seen in the previous section, the *rank* of a lattice is the number of linearly independent vectors in any [equivalent] basis of that lattice. A lattice will be *full-rank* when the number of linearly independent vectors in any basis of this lattice is equal to the number of dimensions in which the lattice is embedded. In such instances, it is clear that any basis for such a lattice will be formed by a set of n vectors, each of n dimensions, allowing us to describe the basis as a square matrix.

3.3 Norm

Many problems in lattice theory involve distances between points. While all the lattice problems can be defined with respect to any norm obtaining equivalent hardness [KS99], the most commonly used one is the *Euclidean norm*. We will operate exclusively with the Euclidean norm in this work.

Definition 2 (Euclidean norm)**.** *Let* $\mathbf{w} \in \mathbb{R}^n$*. The Euclidean norm is the function* $\|.\|_2$ *defined by*

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{n} |\mathbf{w}_i|^2}$$

3.4 Fundamental parallelepiped

Definition 3 (Fundamental parallelepiped)**.** *For a set of linearly independent vectors* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$*, its fundamental parallelepiped is*

$$\mathcal{P}_{1/2}(\mathbf{B}) := \mathbf{B} \cdot \left[-\tfrac{1}{2}, \tfrac{1}{2}\right)^k = \left\{ \sum_{i \in [k]} c_i \cdot \mathbf{b}_i : c_i \in \left[-\tfrac{1}{2}, \tfrac{1}{2}\right) \right\}$$

Figure 2 shows the fundamental parallelepiped of the lattice shown at Figure 1.

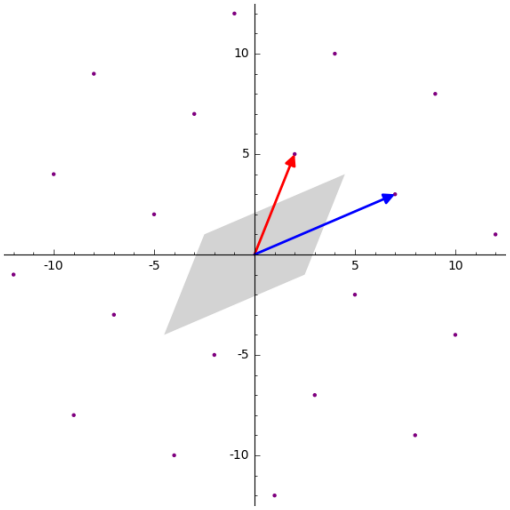


Figure 2: In gray, the fundamental parallelepiped of the two-dimensional basis $\mathbf{b}_1 = [2, 5], \mathbf{b}_2 = [7, 3]$

3.5 Determinant

The determinant of a lattice geometrically corresponds to the inverse of the density of the lattice points in $\mathbb{R}^n$, or equivalently, to the n-dimensional volume of the fundamental parallelepiped defined by the lattice basis $\mathbf{B}$.

For the case when the lattice is full-rank, we can compute the determinant of a lattice as the absolute value of the determinant of the basis matrix

$$det(\mathcal{L}(\mathbf{B})) = |det(\mathbf{B})|$$

Notice that the value of the determinant is independent of the choice of the basis: lattice equivalence is shown by means of a transformation by an unimodular matrix $\mathbf{U}$ applied to a basis from the lattice, and so we have

$$det(\mathbf{U}) = \pm 1$$
$$|det(\mathbf{BU})| = |det(\mathbf{B})| \cdot |det(\mathbf{U})| = |det(\mathbf{B})|$$

And so the choice of basis for a particular lattice has no effect on the volume of the fundamental parallelepiped.

To compute the determinant of a lattice that is not full-rank (i.e., it has rank k with $k < n$), the process is a bit more involved. We first will need to find an orthonormal basis $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_k \in \mathbb{R}^n$ of the space in which the lattice is embedded, and then transform the lattice basis $\mathbf{B}$ by expressing it in this orthonormal basis, which will yield a square matrix $\mathbf{B}' \in \mathbb{R}^{k \times k}$. We can then find the determinant of the lattice by computing the absolute value of the determinant of the matrix $\mathbf{B}'$. Note that an orthonormal basis of the space in which the lattice is embedded can be computed by applying the Gram-Schmidt process (see Section 4.2.1) to the lattice basis $\mathbf{B}$, and then normalizing the obtained vectors.

3.6 Minima

The minimum $\lambda_i(\mathcal{L})$ of a lattice $\mathcal{L}$ is defined as the radius of the smallest hypersphere centered at the origin, containing i linearly independent lattice vectors. Notice that the minimum distance between two points belonging to a given lattice will be $\lambda_1(\mathcal{L})$, which is equivalent to the length of a shortest nonzero lattice vector[3]:

$$\lambda_1(\mathcal{L}) := \min_{\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$$

3.7 Orthogonality defect

Many lattice problems have computational hardness proportional to the orthogonality of the basis used to define the lattice. As mentioned before, a lattice can be expressed with infinite many equivalent bases; but not all of them will have the same orthogonality. For a specific lattice, we informally say that a basis is good if it has high orthogonality, and that it is bad if it has low orthogonality. Moreover, a lattice generated with random coefficients drawn from a uniform distribution will have low orthogonality with high probability.

To compare the orthogonality of lattice bases, we define the *orthogonality defect* as

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

The orthogonality defect is equal to 1 if and only if the basis $\mathbf{B}$ is completely orthogonal. As orthogonality decreases, the orthogonality defect increases.

[3]Lattices have multiple shortest vectors.

It is common to normalize this metric by taking the n'th root (where n is the rank of the lattice), so that if the vectors are multiplied by some constant factor, the orthogonality defect is scaled by the same factor.

A related metric can be found in the research literature to also measure the quality of a basis: the Hermite factor δ_0^n

$$\delta_0^n = \frac{\|\mathbf{b}_0\|}{\sqrt[n]{det(\mathcal{L}(\mathbf{B}))}}$$

Where $\mathbf{b}_0$ is the shortest non-zero vector from the lattice basis $\mathbf{B}$. Similarly to the orthogonality defect, we can also refer to δ_0 itself, and call it the root-Hermite factor.

3.8 Lattice duality

Definition 4 (The dual lattice)**.** *The dual of a lattice $\mathcal{L} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{L}^* := \{\mathbf{w} : \langle \mathbf{w}, \mathcal{L} \rangle \subseteq \mathbb{Z}\}$$

That is, the set of points whose inner products with the vectors in $\mathcal{L}$ are all integers.

3.9 q-ary lattices

Definition 5 (q-ary lattice)**.** *A q-ary lattice is a lattice $\mathcal{L}$ satisfying $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ for some (probably prime) integer q. In other words, the membership of a vector $\mathbf{x}$ in $\mathcal{L}$ is determined by $\mathbf{x}$ mod q.*

Most lattice-based cryptographic constructions use problems defined on q-ary lattices (where the q is a known parameter) as their hard on average problem. Note that any integer lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ is a q-ary lattice for some q, e.g., whenever q is an integer multiple of the determinant $det(\mathcal{L})$.

For practical lattice-based cryptographic constructions, there are two common forms of expressing q-ary lattices, which allows easy operations in the implementations. They are the following, where q is usually prime and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$:

$$\Lambda_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^n : \mathbf{y} = \mathbf{A}\mathbf{z} \bmod q : \mathbf{z} \in \mathbb{Z}^m\}$$

We will call this the Λ_q form.

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

This second lattice contains all the vectors that are orthogonal modulo q to the columns of $\mathbf{A}$. We will call this the orthogonal Λ_q form.

It can be seen that these lattices are dual to each other, up to normalization; namely, $\Lambda_q^{\perp}(\mathbf{A}) = q\Lambda_q(\mathbf{A})^*$ and $\Lambda_q(\mathbf{A}) = q\Lambda_q^{\perp}(\mathbf{A})^*$

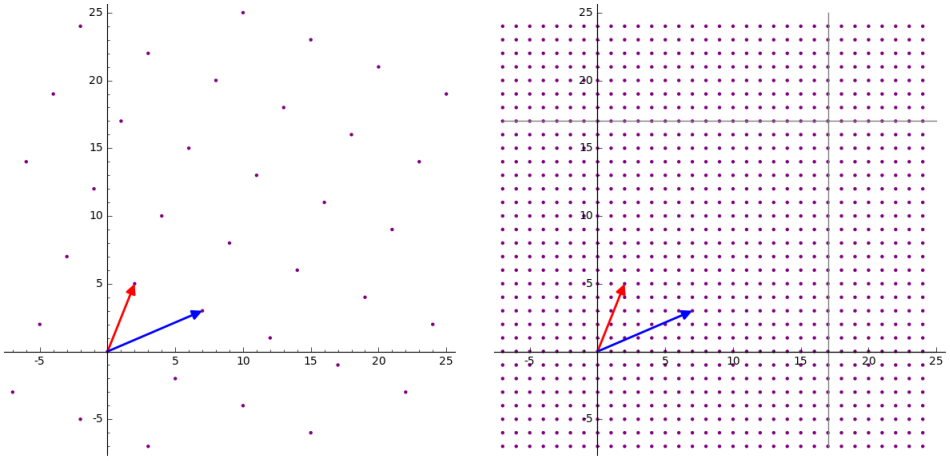
As we will see later, finding the lattice point given a perturbation of it using the Λ_q form is related to solving the LWE problem, whereas finding short vectors using the orthogonal Λ_q form is related to solving the SIS problem.

One interesting observation we can make is that in the Λ_q form, $\mathbf{A}$ is usually not a basis of the lattice. We can see this for example with the fact that usually, no integer linear combination of the columns of $\mathbf{A}$ can generate all the points of the form $q\mathbb{Z}^n$. Consider for example the two dimensional q-ary lattice defined with $\mathbf{b}_1 = [2, 5], \mathbf{b}_2 = [7, 3]$ (the same vectors used in figure 1) and $q = 17$. There is no integer solution to the following equation, which would show how to construct the point $[q, 0]$

$$17 = 2z_1 + 7z_2$$

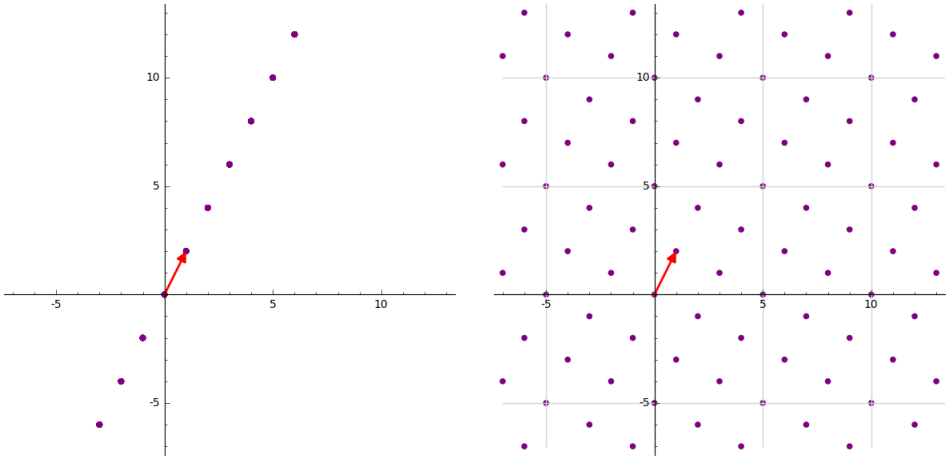
$$0 = 5z_1 + 3z_2$$

Another observation we draw is that any q-ary lattice has full rank because it contains $q\mathbb{Z}^n$. From this observation, it is easy to see that with high probability, a q-ary lattice defined with n independent vectors in $\mathbb{Z}_q^n$ with q prime (that is, the case where we have $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ in the $\Lambda_q(\mathbf{A})$ expression) is the same as $\mathbb{Z}^{n\times n}$. This situation can be seen in the example shown in Figure 3. It will not be the case whenever $det(\mathbf{A}) = 0 \bmod q$. The explanation for this behavior can be seen in a more general case in Lemma 1.



(a) Using $\mathcal{L}$ form. (b) Using Λ_q form.

Figure 3: The two lattices have been generated with the same pair of vectors, but the one on the right has been generated using the Λ_q form with $q = 17$



(a) Using $\mathcal{L}$ form. (b) Using Λ_q form.

Figure 4: The two lattices have been generated with the same vector, but the one on the right has been generated using the Λ_q form with $q = 5$

To get a better intuition on the geometry of a lattice defined in the Λ_q form, we propose the following lemma, for which we have developed a more general proof as part of this thesis. See Figures 3 and 4 for two examples where the lemma applies.

Lemma 1. *For any $k \leq n$ and any q with its prime factors having multiplicity 1; any q-ary lattice with q^k points in the space defined by the hypercube $[0, q)^n$ can be expressed with k vectors in $\mathbb{Z}_q^n$ in the Λ_q form.*

Proof. We analyze this lemma by finding the cardinality of the set generated by the q-ary lattice expressed in Λ_q form with matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$

$$c = \#\{\mathbf{y} \in \mathbb{Z}^n : \mathbf{y} = \mathbf{A}\mathbf{z} \bmod q : \mathbf{z} \in \mathbb{Z}^m\}$$

We factor q into k primes, assuming that there are no powers of primes in the factorization of q

$$q = \prod_{i=1}^{l} p_i$$

We want to find the number of unique $\mathbf{y}$ that can be generated from the following expression with $\mathbf{z} \in \mathbb{Z}^m$

$$\mathbf{A}\mathbf{z} \equiv \mathbf{y} \bmod q$$

And from the Chinese Reminder Theorem, we know that we can get the following bijection

$$\begin{aligned} \mathbf{A}\mathbf{z}^{(1)} &\equiv \mathbf{y}^{(1)} \bmod p_1 \\ \mathbf{A}\mathbf{z}^{(2)} &\equiv \mathbf{y}^{(2)} \bmod p_2 \\ &\vdots \\ \mathbf{A}\mathbf{z}^{(l)} &\equiv \mathbf{y}^{(l)} \bmod p_l \end{aligned}$$

Where we can easily find, for each $i \in \{1, \ldots, l\}$, the number of unique $\mathbf{y}^{(i)}$ generated by calculating $p_i^{\text{rank } \mathbf{A} \bmod p_i}$, where the rank of a matrix modulo p_i is defined as the size of the largest non-vanishing (non-zero) minor modulo p_i.

So, we conclude that the cardinality of Λ_q with matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is:

$$c = \prod_{i=1}^{l} p_i^{\text{rank } A \bmod p_i}$$

Now, if the cardinality of a given Λ_q is q^k with $k \leq n$, we can factor q^k:

$$q^k = \prod_{i=1}^{l} p_i^k$$

So we know that in the Chinese Reminder Theorem, for each $i \in \{1, \ldots, l\}$ it will be required that rank $\mathbf{A}'$ mod $p_i = k$, and so there will exist some $\mathbf{A}' \in \mathbb{Z}_q^{n \times k}$ for which the given $\Lambda_q = \Lambda_q(\mathbf{A}')$. □

Here we show a possible Λ_q expression of the examples shown in Figure 3 ($\mathbf{A}_1$) and Figure 4 ($\mathbf{A}_2$) where the relation between the lattice cardinality and the number of vectors in the Λ_q expression follows Lemma 1.

$$\mathbf{A_1} = \begin{pmatrix} 2 & 7 \\ 5 & 3 \end{pmatrix} \qquad q = 17$$

$$\text{rank } \mathbf{A_1} \bmod 17 = \text{rank } \begin{pmatrix} 2 & 7 \\ 5 & 3 \end{pmatrix} \bmod 17 = 2$$

$$\#\Lambda_q(\mathbf{A_1}) = 17^2 = 289$$

$$\mathbf{A_2} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \qquad q = 5$$

$$\text{rank } \mathbf{A_2} \bmod 5 = \text{rank } \begin{pmatrix} 1 \\ 3 \end{pmatrix} \bmod 5 = 1$$

$$\#\Lambda_q(\mathbf{A_2}) = 5^1 = 5$$

Here we show an example of calculating the cardinality of two random Λ_q lattices. In Figure 5 we show a visual representation of the two lattices from the example.

$$\mathbf{A_3} = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \qquad q = 6 = 2 \cdot 3$$

$$\text{rank } \mathbf{A_3} \bmod 2 = \text{rank } \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \bmod 2 = 2$$

$$\text{rank } \mathbf{A_3} \bmod 3 = \text{rank } \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \bmod 3 = 1$$

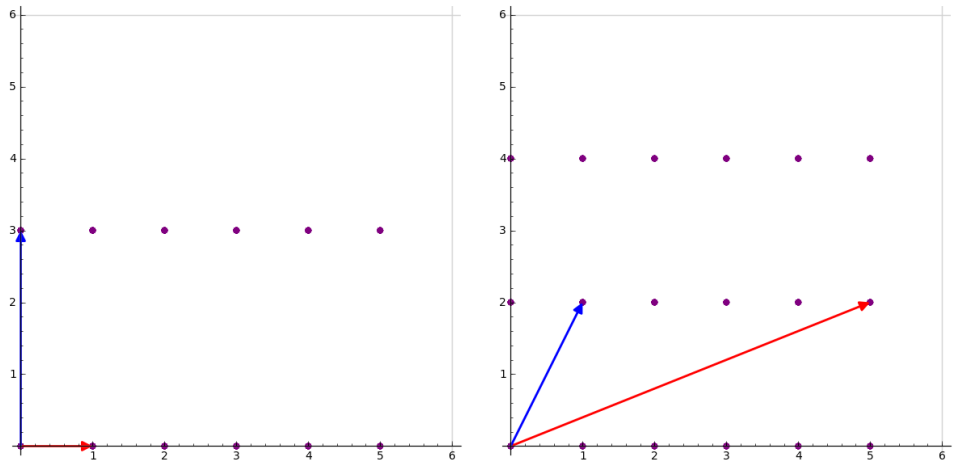
$$\#\Lambda_q(\mathbf{A_3}) = 2^2 \cdot 3^1 = 12$$

$$\mathbf{A_4} = \begin{pmatrix} 5 & 1 \\ 2 & 2 \end{pmatrix} \qquad q = 6 = 2 \cdot 3$$

$$\text{rank } \mathbf{A_4} \bmod 2 = \text{rank } \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \bmod 2 = 1$$

$$\text{rank } \mathbf{A_4} \bmod 3 = \text{rank } \begin{pmatrix} 2 & 1 \\ 2 & 2 \end{pmatrix} \bmod 3 = 2$$

$$\#\Lambda_q(\mathbf{A_4}) = 2^1 \cdot 3^2 = 18$$



(a) $\Lambda_q(\mathbf{A}_3)$. (b) $\Lambda_q(\mathbf{A}_4)$.

Figure 5: Cardinality of Λ_q lattices for two different set of q and matrices $\mathbf{A}_3$ and $\mathbf{A}_4$.

3.10 Lattice problems

Many lattice problems have been studied for a long time and have been conjectured or proved to be hard on the average case, which contrast with other problems on which cryptographic schemes have been based that are only hard on the worst case (notice that even most NP-hard problems are probably only worst case hard). Moreover, the simplicity and flexibility of some of these average-case hard problems (like LWE) combined with proofs that show that they are probably as hard as certain lattice problems in the worst case, and the fact that they appear to require time exponential in the main security parameter to be solved, makes them excellent candidates to base cryptographic schemes on.

Most of the lattice problems exist in two versions: the exact and the approximation problem; the later one being a generalization of the exact problem and defined with an approximation factor $\gamma(n)$ in terms of the lattice dimension. Conversely, the exact version is a particular instance of the approximation version with $\gamma(n) = 1$. It has been proved that the approximation versions up to a polynomial factor of n are hard problems, and those are the ones used in cryptographic schemes. On the other hand, when the approximation factor reaches an exponential factor of n, these problems become easy to solve.

Some of the main lattice problems are defined below (the Learning With Errors problem definition can be found in its own at Section 4.1).

Definition 6 (Approximate Shortest Vector Problem (γ-SVP))**.** *Given a lattice basis* $\mathbf{B}$*, find a non-zero vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.

Definition 7 (Approximate Closest Vector Problem (γ-CVP))**.** *Given a lattice basis* $\mathbf{B}$ *and a target vector* $\mathbf{t}$ *(not necessarily in the lattice), find a lattice point* $\mathbf{u} \in \mathcal{L}(\mathbf{B})$ *such that* $\mathbf{v} = \arg\min_{\mathbf{v} \in \mathcal{L}} \|\mathbf{t} - \mathbf{v}\|$, $\|\mathbf{u} - \mathbf{v}\| \leq \gamma \|\mathbf{t} - \mathbf{v}\|$.

Definition 8 (Shortest Independent Vector Problem (SIVP))**.** *Given a lattice basis* $\mathbf{B}$ *and a parameter* $q \in \mathbb{Z}$*, find a set of shortest* q *linearly independent lattice vectors (i.e., a set of linearly independent vectors* $\mathbf{s}_1, \ldots, \mathbf{s}_q \in \mathcal{L}(\mathbf{B})$ *such that* $\mathbf{s}_1, \ldots, \mathbf{s}_q \leq \lambda_q(\mathbf{B})$*).*

Definition 9 (Bounded Distance Decoding Problem (BDD_γ))**.** *Given a basis* $\mathbf{B}$ *of an* n*-dimensional lattice* $\mathcal{L} = \mathcal{L}(B)$ *and a target point* $\mathbf{t} \in \mathbb{R}^n$ *with the guarantee that* $dist(\mathbf{t}, \mathcal{L}) < d = \lambda_1(\mathcal{L})/(2\gamma(n))$*, find the unique lattice vector* $\mathbf{v} \in \mathcal{L}$ *such that* $\|\mathbf{t} - \mathbf{v}\| < d$*. This is a special case of the CVP problem where the vector given is guaranteed to be close to a lattice point.*

Where we define the distance $dist(\mathbf{t}, \mathcal{L})$ with $\mathbf{t} \in \mathbb{R}^n$ as the distance from the point $\mathbf{t}$ to the closest point in the lattice $\mathcal{L}$, and where $\gamma(n)$ is a function used to bound the maximum possible distance in the problem.

The definition of the Learning With Errors problem can be found in the next Section, along with some related theorems.

4 Learning With Errors (LWE)

4.1 Problem definition

Definition 10 (Learning With Errors (LWE) - [Reg09])**.** *Let n, q (possibly prime) be positive integers, $\mathcal{X}$ be a discrete probability distribution on $\mathbb{Z}$ called the error distribution (possibly the discrete Gaussian distribution) and $\mathbf{s}$ a secret vector in $\mathbb{Z}_q^n$. We denote by $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}$ according to $\mathcal{X}$ and considering it in $\mathbb{Z}_q$, and returning $(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Decision-LWE *is the problem of deciding whether pairs $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Search-LWE *is the problem of recovering* $\mathbf{s}$ *from $(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $\mathcal{L}_{\mathbf{s},\mathcal{X}}$.*

For either version of the problem, the number m of LWE samples available is a polynomial of n.

Notice that we can use the fact that Decision-LWE is hard to treat LWE samples as pseudorandom elements, considering that they can't be easily distinguished from uniform samples.

Solving LWE instances where the coefficients of s are taken from the $\mathcal{X}$ error distribution is at least as hard as solving LWE instances where s is taken uniformly from $\mathbb{Z}_q^n$. To prove this we use the following lemma:

Lemma 2 ([ACPS09])**.** *Given access to an oracle $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ returning samples of the form $(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $e \leftarrow \mathcal{X}$ and $\mathbf{s} \in \mathbb{Z}_q^n$, we can construct samples of the form $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{e} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $e \leftarrow \mathcal{X}$ and $\mathbf{e} \leftarrow \mathcal{X}(\mathbb{Z}^n)$ at the loss of n samples overall. This is also called the normal form in [Pei16].*

Proof. Given an instance

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

Here $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times n}$ is invertible and $\mathbf{b}_1 \in \mathbb{Z}_q^n$, we transform it to the instance

$$\bar{\mathbf{A}} = -\mathbf{A}_2 \cdot \mathbf{A}_1^{-1}, \ \bar{\mathbf{b}} = \bar{\mathbf{A}}\mathbf{b}_1 + \mathbf{A}_2$$

As before, $\bar{\mathbf{A}}$ is uniformly random, because $\mathbf{A}_2$ is independent of $\mathbf{A}_1$. Now observe that when the input comes from an LWE distribution, each $\mathbf{b}_i = \mathbf{A}_i\mathbf{s} + \mathbf{e}_i$ for some $\mathbf{s} \in \mathbb{Z}_q^n$, where the entries of each $\mathbf{e}_i$ are independently drawn from $\mathcal{X}$, so we have

$$\bar{\mathbf{b}} = \bar{\mathbf{A}} \cdot \mathbf{A}_1\mathbf{s} + \bar{\mathbf{A}}\mathbf{e}_1 + \mathbf{A}_2\mathbf{s} + \mathbf{e}_2 = \bar{\mathbf{A}}\mathbf{e}_1 + \mathbf{e}_2.$$

That is, the instance $\bar{\mathbf{A}}$, $\bar{\mathbf{b}}$ comes from the LWE distribution with secret $\mathbf{e}_1$, and finding $\mathbf{e}_1$ yields the original secret $\mathbf{s} = \mathbf{A}_1^{-1}(\mathbf{b}_1 - \mathbf{e}_1)$. □

Certain instantiations of LWE are computationally hard and have worst-case hardness theorems to support this belief. For a discrete Gaussian error distribution $\psi = D_s$ with $s \geq 2\sqrt{n}$, solving search-LWE$_{n,q,\psi}$ is at least as hard as quantumly approximating the shortest vector problem (γ-SVP) on any n-dimensional lattice to within $\gamma(n) = O(n \cdot q/s)$ approximation factors, i.e., there exists a quantum reduction from approximate SVP (in the worst case) to search-LWE$_{n,q,\psi}$ [Reg05]. Moreover, for $q \geq 2^{n/2}$ there is a classical reduction from approximate SVP for the same approximation factors [Pei09].

Decision-LWE and Search-LWE are equivalent problems under mild conditions on the modulus q and the Gaussian parameter s. The search to decision reduction is trivial: if search is solved, $\mathbf{s}$ is obtained, which can be used to verify that $\mathbf{b}$ are LWE samples by checking that $\mathbf{e} = \mathbf{b} - \mathbf{A}\mathbf{s}$ is small.

The other direction is shown in the proof below.

Lemma 3 ([Reg09])**.** *Let $n \geq 1$ be some integer, $2 \leq q \leq \text{poly}(n)$ be a prime, and $\mathcal{X}$ be some distribution on $\mathbb{Z}_q$. Assume that we have access to a procedure W that, for all $\mathbf{s}$, accepts with probability exponentially close to 1 on inputs from $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ and rejects with probability exponentially close to 1 on uniformly random inputs. Then, there exists an efficient algorithm W' that, given samples from $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ for some $\mathbf{s}$, outputs $\mathbf{s}$ with probability exponentially close to 1.*

Proof. We show how W' finds the first component $\mathbf{s}_{(0)}$ of $\mathbf{s}$; finding the other components is similar. For any $k \in \mathbb{Z}_q$ consider the following transformation. Given a pair $(\mathbf{a}, c)$ as input to W', let it output the pair $(\mathbf{a} + (l, 0, \ldots, 0), c + lk)$ where $l \in \mathbb{Z}_q$ is chosen uniformly at random. It is easy to see that this transformation takes the uniform distribution to itself. On the other hand suppose the input pair $(\mathbf{a}, c)$ is sampled from $\mathcal{L}_{\mathbf{s},\mathcal{X}}$. If $k = \mathbf{s}_{(0)}$ then this transformation takes $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ into itself. If $k \neq \mathbf{s}_{(0)}$ then this transformation takes $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ to the uniform distribution. There are only polynomially many (namely q) possibilities for $\mathbf{s}_{(0)}$, so we can try all of them as possible k values. For each k value, let the output of W' be the input to W. Then as W can distinguish $\mathcal{L}_{\mathbf{s},\mathcal{X}}$ from uniform, it can tell whether $k = \mathbf{s}_{(0)}$. □

4.2 Cryptanalysis

There are two main strategies to attack cryptosystems based on the LWE problem (see Section 7 for an example of such cryptosystem): one is referred as a distinguishing attack while the other is referred as a decoding attack.

As the name suggests, the distinguishing attack tackles the decision version of LWE: the adversary distinguishes an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$ from uniformly random which is typically enough to break an LWE problem as seen in Lemma 3. To do so, the adversary finds a short non-zero integer vector $\mathbf{v}$ such that $\mathbf{A}\mathbf{v} = \mathbf{0} \bmod q$, which can be seen as a short vector in the dual of the LWE lattice $\Lambda_q(\mathbf{A})$, that is, $\Lambda_q^{\perp}(\mathbf{A})$. Notice that this involves solving the approximate SVP. For further details on how the attack works, see [MR09].

On the other hand, the decoding attack, which we explain in detail here, tackles the search version of LWE, and thus, is able to recover the secret $\mathbf{s}$ from an LWE instance. The attack can be seen as solving the LWE problem by considering it as a bounded-distance decoding problem with the lattice $\Lambda_q(\mathbf{A})$; which implies, in fact, that it will only have a chance of working if the error is small enough.

To solve the BDD problem on lattices, the standard method is to apply Babai's Nearest Plane algorithm [Bab85], which given a lattice basis and a target point, returns a lattice point that is 'relatively close' to the target point. The success probability of the Nearest Plane algorithm to return the closest lattice point depends on the quality of the input basis. This decoding algorithm uses the Gram-Schmidt orthogonalization process, which is detailed in Section 4.2.1.

To improve the success probability of the Nearest Planes algorithm we can first apply a lattice reduction algorithm on the input basis as explained in Section 4.2.2, which will return a better basis for the lattice, i.e., a more orthogonal basis.

4.2.1 Gram-Schmidt orthogonalization

The *Gram-Schmidt orthogonalization* is the process of, given an ordered set of linearly independent vectors (which define a lattice) $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^n$, generating a set of orthogonal vectors $\tilde{\mathbf{B}}$ defined iteratively as $\tilde{\mathbf{b}}_1 = \mathbf{b}_1$ and $\tilde{\mathbf{b}}_i$ as the component of $\mathbf{b}_i$ orthogonal to the $span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ for $i = 2, \ldots, k$:

$$\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=1}^{1-i} \mu_{ji} \tilde{\mathbf{b}}_j \qquad \text{where } \mu_{ji} = \frac{\langle \tilde{\mathbf{b}}_j, \mathbf{b}_i \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle}$$

As seen from the definition, the Gram-Schmidt orthogonalization works by projecting each vector on the space orthogonal to the span of the previous vectors, and as such, the last several vectors are typically very short. Notice also that the order of the vectors is important in the orthogonalization process: changing the order of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ will generally change the output vectors. Furthermore, the vectors $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_k$ are generally not a vectors of the lattice $\mathcal{L}(\mathbf{B})$.

Figure 6 and Figure 7 show the Gram-Schmidt orthogonalization of two different basis for the same lattice: one good (low orthogonality defect) and the other one bad (high orthogonality defect).

4.2.2 Lattice reduction algorithm - LLL and BKZ

The last several Gram-Schmidt vectors of $\mathbf{B}$ are typically very short, which will make the parallelepiped $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ very 'long and skinny', and so the Gaussian error vector $\mathbf{e}$ is very unlikely to land in it, causing the Nearest Plane to return an incorrect answer.

As mentioned before, to improve this situation we can apply a reduction algorithm on the lattice basis, which will make the parallelepiped $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ more evenly spread in every direction. Gamma and Nguyen, in their work to analyze lattice reduction algorithms [GN08], identified the *Hermite factor* (see Section 3.7 for the definition) of the reduced basis as the main parameter to evaluate the runtime of the algorithm as well as the quality of the reduced basis. As the basis gets reduced, not only the vectors become more 'orthogonal', they also get smaller, with the optimal case where a shortest vector will be in the basis.

The general reduction algorithm is BKZ [SE94] which is parametrized by the block size k. For $k = 2$ the algorithm runs in polynomial time and it behaves as the LLL algorithm [LLL82], but it will only contain a short vector within exponential factors of a shortest vector. When $k = n$, i.e., the full size of the basis, then the quality of the output basis is optimal, but requires at least exponential runtime.

The LLL algorithm works by taking pairs of vectors with decreasing norm and subtracting multiples of the smallest one from the bigger one. This process is done iteratively until no vector norm can be decreased.

On the other hand, the BKZ algorithm divides the vectors from the input basis in blocks of k vectors, and generally speaking, subtracts linear combinations of these vectors to reduce the norm of the other vectors. As in LLL, this process is done iteratively until no more improvements can be made on the norms of the vectors.

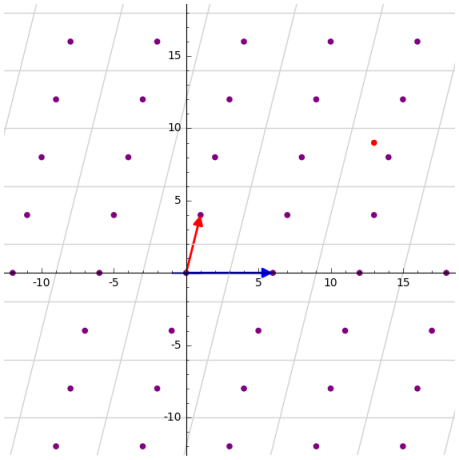
For further details on these lattice reduction algorithms, we refer the reader to the original papers [LLL82] and [SE94], or to the summary presented in [ACPS09].

4.2.3 Lattice decoding algorithm - Babai's Nearest Planes

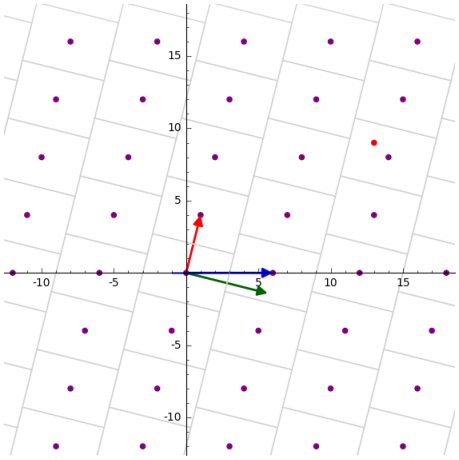
To break *LWE* by a decoding attack given $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$, the following condition must hold: $\|\mathbf{e}\| << \lambda_1(\mathcal{L}(\mathbf{A}))$; otherwise, $\mathbf{e}$ hides the original lattice point in an unrecoverable way ($\mathbf{b}$ would be closer to a different lattice point than the original). This way, as mentioned before, we can see an LWE instance as an instance of a BDD problem where $\mathbf{b}$ is a point which is bounded in distance from a lattice point $\mathbf{v} = \mathbf{A} \cdot \mathbf{s}$. In this case, we could bound the distance with high probability, for instance, with three times the standard deviation of the Gaussian distribution used in the error of the LWE problem.

Babai's Nearest Plane algorithm takes a lattice basis $\mathbf{B} = \mathbf{b_1}, \ldots, \mathbf{b_k}$ and a target point $\mathbf{t} \in \mathbb{R}^n$ as input, and returns a lattice point $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ that is 'relatively close' to $\mathbf{t}$. The success probability of the Nearest Plane algorithm to solve the BDD problem depends on the orthogonality defect of the input basis. Specifically, the Nearest Plane algorithm will recover the correct lattice point $\mathbf{v}$ if $\mathbf{t} \in \mathbf{v} + \mathcal{P}_{1/2}(\mathbf{B})$, that is, if the target vector lands in the parallelepiped generated by the Gram-Schmidt vectors of $\mathbf{B}$ centered at $\mathbf{v}$. Figure 6 shows how an instance of the BDD problem (the point in red) can be solved by the Nearest Plane algorithm when the basis is good, since the point lands inside the Gram-Schmidt parallelepiped centered in the original lattice point. On the other hand, in Figure 7 a bad basis is used, and the Nearest Plane is unable to decode the correct lattice point: the target point (in red) has landed outside the parallelepiped of interest.

The algorithm works by recursively computing the closest vector on the sublattice spanned by subsets of the Gram-Schmidt vectors $\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_k$. The procedure can be found in the Appendix B.2, written in Python for Sage.

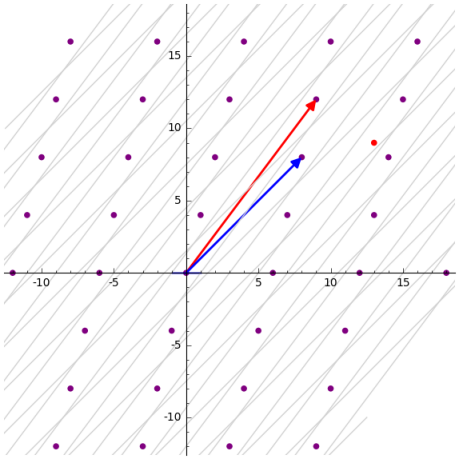


(a) Showing the fundamental parallelepiped of the lattice basis.

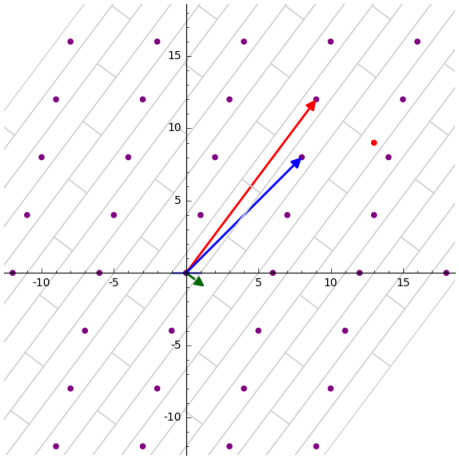


(b) Showing the fundamental parallelepiped from the *Gram-Schmidt* vectors.

Figure 6: Lattice representation with a good quality basis, with a perturbed lattice point in red. The basis vectors are shown in red and blue, whereas the Gram-Schmidt vectors are shown in red and green.



(a) Showing the fundamental parallelepiped of the lattice basis.



(b) Showing the fundamental parallelepiped from the *Gram-Schmidt* vectors.

Figure 7: Lattice representation with a bad quality basis, with a perturbed lattice point in red. The basis vectors are shown in red and blue, whereas the Gram-Schmidt vectors are shown in red and green.

Although this algorithm can be run in polynomial time, as stated before, the success rate of finding the right lattice point lies on the quality of the input basis, that is, how well reduced it is. Reducing a random basis such that the Nearest Planes algorithm finds the right lattice point with high probability takes exponential time, which is sound with our previous knowledge of the hardness of LWE.

Some authors have proposed modifications of the Nearest Plane algorithm [LP11] to reduce the computational cost of the lattice reduction step by generalizing it such that it admits a time/success trade-off. This modification consists in expanding the exploration space by trying multiple (d_i) distinct planes during each step i in the recursion. This has the effect of making the parallelepiped $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ wider in the direction of the desired vectors $\tilde{\mathbf{b}}_i$ by some factors d_i.

5 Ring lattices

Several schemes based on LWE have appeared in the literature through the years, although two main factors have made them impractical. The first difficulty arises from the size of the keys (which correspond to a lattice basis): being an $n \times n$ matrix they get easily big for a reasonable secure dimension n. The second issue has been the operation speed; although matrices multiplications can be parallelized, the operation overall is not very competitive with other classical schemes.

To solve these inconveniences, the usage of ring lattices (or ideal lattices) to design cryptographic schemes was proposed.

5.1 Definition

The basis $\mathbf{A} \in \mathbb{Z}^{n \times n}$ of an ideal lattice is constructed with a vector $\mathbf{a} \in \mathbb{Z}^n$ iteratively multiplied by a transformation matrix $\mathbf{F} \in \mathbb{Z}^{n \times n}$ defined from a vector $\mathbf{f} \in \mathbb{Z}^n$, as follows

$$\mathbf{F}^*\mathbf{a} = [\mathbf{a}, \mathbf{F}\mathbf{a}, \ldots, \mathbf{F}^{n-1}\mathbf{a}] \qquad \text{where } \mathbf{F} = \left[\begin{array}{c|c} \begin{array}{c} \mathbf{0}^T \\ \hline \ddots \\ \mathbf{I} \\ \ddots \end{array} & -\mathbf{f} \end{array}\right]$$

The lattices that follow this particular structure have been named ideal lattices because they can be equivalently characterized as ideals of the ring of modular polynomials $\mathbb{Z}[x]/\langle f(x) \rangle$ where $f(x) = x^n + f_n x^{n-1} + \cdots + f_1 \in \mathbb{Z}[x]$.

That means that working on the polynomial domain modulo $f(x)$ is equivalent to working on the ideal lattice domain characterized by $\mathbf{F}$, as seen in the following example for $n = 4$ and a specific $\mathbf{f}$.

We consider the coefficients for $\mathbf{a}$ and $\mathbf{s}$ to be

$$\mathbf{a}^T = (a_1, a_2, a_3, a_4) \quad \mathbf{s}^T = (s_1, s_2, s_3, s_4)$$

And use the following $\mathbf{f}$

$$\mathbf{f}^T = (1, 0, 0, 0)$$

First we do the operations working with lattice representation, i.e., with matrices and vectors

$$\mathbf{F} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{A} = \mathbf{F}^*\mathbf{a} = \begin{pmatrix} a_1 & -a_4 & -a_3 & -a_2 \\ a_2 & a_1 & -a_4 & -a_3 \\ a_3 & a_2 & a_1 & -a_4 \\ a_4 & a_3 & a_2 & a_1 \end{pmatrix}$$

$$\mathbf{As} = \begin{pmatrix} a_1 & -a_4 & -a_3 & -a_2 \\ a_2 & a_1 & -a_4 & -a_3 \\ a_3 & a_2 & a_1 & -a_4 \\ a_4 & a_3 & a_2 & a_1 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = \begin{pmatrix} a_1s_1 - a_4s_2 - a_3s_3 - a_2s_4 \\ a_2s_1 + a_1s_2 - a_4s_3 - a_3s_4 \\ a_3s_1 + a_2s_2 + a_1s_3 - a_4s_4 \\ a_4s_1 + a_3s_2 + a_2s_3 + a_1s_4 \end{pmatrix}$$

Now we perform the same operations working with polynomials modulo $f(x)$. $\mathbf{a}, \mathbf{s}$ and $\mathbf{f}$ will now be polynomials

$$a(x) = a_1 + a_2x + a_3x^2 + a_4x^3$$
$$s(x) = s_1 + s_2x + s_3x^2 + s_4x^3$$
$$f(x) = x^4 + 1$$

And so the product will be

$$a(x)s(x) \bmod f(x) =$$
$$(a_1 + a_2x + a_3x^2 + a_4x^3)(s_1 + s_2x + s_3x^2 + s_4x^3) \bmod (x^4 + 1) =$$

$$\begin{aligned}
&a_1s_1 + a_1s_2x + a_1s_3x^2 + a_1s_4x^3 \\
&+a_2s_1x + a_2s_2x^2 + a_2s_3x^3 + a_2s_4x^4 \\
&+a_3s_1x^2 + a_3s_2x^3 + a_3s_3x^4 + a_3s_4x^5 \\
&+a_4s_1x^3 + a_4s_2x^4 + a_4s_3x^5 + a_4s_4x^6 =
\end{aligned}$$

$$\begin{bmatrix} x^4 + 1 = 0 & \bmod (x^4 + 1) \\ x^4 = -1 & \bmod (x^4 + 1) \end{bmatrix}$$

$$\begin{aligned}
&= (a_1s_1 - a_4s_2 - a_3s_3 - a_2s_4) \\
&+(a_2s_1 + a_1s_2 - a_4s_3 - a_3s_4)x \\
&+(a_3s_1 + a_2s_2 + a_1s_3 - a_4s_4)x^2 \\
&+(a_4s_1 + a_3s_2 + a_2s_3 + a_1s_4)x^3 \bmod (x^4 + 1)
\end{aligned}$$

And we see that the coefficients of the vector from the multiplication result in the lattice representation are the same as the coefficients of the multiplication result in the polynomial representation.

Notice that using ideal lattices, we are able to express a rank n ideal lattice with only n values, rather than $n \times n$ as it was the case for general lattices, which allows a more compact representation that requires less space.

Moreover, working with the polynomial representation with certain modules allows a speedup in operations commonly used in lattice-based schemes: polynomial multiplication can be performed in $O(n \log n)$ scalar operations, and in parallel depth $O(\log n)$, using the Fast Fourier Transform (FFT), also called the Number Theoretic Transform in some contexts.

To get an intuition of how this works, notice that if two polynomials are evaluated at n different points, we obtain two n-dimensional vectors that can be multiplied coefficient wise, from which a polynomial can be recovered by means of

interpolation; and this is equivalent to the result obtained by the multiplication of the two polynomials. To be more specific, to do the evaluation and interpolation (polynomial recovery), the FFT and the inverse FFT are used respectively, which cost $O(n \log n)$ each. In the transformed space (evaluations), the multiplication's cost is $O(n)$.

Regarding the polynomial used in the modulo, which defines the ring, we are mainly interested in using cyclotomic polynomials since they allow the previously mentioned multiplication speedup. Although not required to follow the rest of this work, the cyclotomic polynomials are defined as the polynomials of degree $n = \varphi(m)$ whose m roots are

$$\omega_m^i = e^{\frac{2\pi\sqrt{-1}}{m}i}, \text{ for } 1 \leq i < m \text{ with } i \text{ coprime to } m$$

We only focus on cyclotomic polynomials in this work because they follow all the security proofs from the literature and, as stated previously, because they allow fast multiplications. Moreover, different cyclotomic polynomials may be used in the construction of lattice problems with equivalent hardness, but we will only use the ring $\mathbb{Z}[x]/\langle x^n + 1\rangle$ where n is a power of 2 (we used this ring in the previous example), as proposed by [LPR13].

When working with the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$, where we know that $f(x) = x^n+1$ is a cyclotomic polynomial, for n a power of 2, one obtains the family of the so called "anti-cyclic integer lattices", i.e., lattices in $\mathbb{Z}^n$ that are closed under the operation that cyclically rotates the coordinates and negates the cycled element.

Regarding the hardness of the problems seen in Section 3.10, no algorithm is known that solves them on ideal lattices any better than on general lattices; so it is reasonable to assume that solving lattice problems on ideal lattices is as hard as the general case, and thus we can use them to construct secure cryptographic schemes.

5.2 Ring-LWE

Ring-LWE is parametrized by a ring R of degree n over $\mathbb{Z}$, a positive integer modulus q defining the quotient ring $R_q = R/qR$, and an error distribution $\mathcal{X}$ over R. As mentioned previously, one usually takes R to be a cyclotomic ring, and $\mathcal{X}$ to be a discrete Gaussian distribution[4].

Definition 11 (Ring-LWE (R-LWE) - [**LPR10**]). *Let n, q (possibly prime) be positive integers, $\mathcal{X}$ be a discrete spherical probability distribution on R_q called the error distribution (possibly the discrete Gaussian distribution) and s a secret element in R_q. We denote by $A_{\mathbf{s},\mathcal{X}}$ the R-LWE probability distribution on $R_q \times R_q$ obtained by choosing $a \in R_q$ uniformly at random, choosing $e \in \mathbb{Z}^n$ according to $\mathcal{X}$ and considering it in R_q, and returning $(a, b = a \cdot s + e \bmod q) \in R_q \times R_q$.*

Decision-LWE *is the problem of deciding whether pairs $(a, c) \in R_q \times R_q$ are sampled according to $A_{\mathbf{s},\mathcal{X}}$ or the uniform distribution on $R_q \times R_q$.*

Search-LWE *is the problem of recovering s from $(a, b = a \cdot s + e \bmod q) \in R_q \times R_q$ sampled according to $A_{\mathbf{s},\mathcal{X}}$.*

For either version of the problem, the number m of R-LWE samples available is a polynomial of n.

Similarly to LWE, certain instantiations of Ring-LWE are supported by worst-case hardness theorems [LPR13]. For a discrete Gaussian error distribution $\psi = D_s$ with $s \geq 2 \cdot \omega(\sqrt{\log n})$ and for any ring, solving R-LWE is at least as hard as quantumly approximating the SVP on any ideal lattice to within $\gamma(n) = O(\sqrt{n} \cdot q/s)$ approximation factors.

Moreover, for any cyclotomic ring and for appropriate moduli q, decision R-LWE is classically equivalent to search R-LWE for any spherical error distribution [LPR13], such as the discrete Gaussian error used along this thesis. This shows that the R-LWE distribution is in fact pseudorandom assuming that the search problem is hard.

[4] Actually, for a general cyclotomic, $\mathcal{X}$ is a discrete Gaussian distribution in the canonical embedding of R, but for the particular case of $R_q = \mathbb{Z}[x]/\langle x^n + 1\rangle$, with n a power of 2, a spherical distribution in the canonical embedding turns out to be a spherical distribution in the coefficients of R_q. The definition of the canonical embedding relates to the Number Theory background used in the security proofs for ideal lattices and is out of the scope of this work.

5.2.1 Efficiency

In most applications, each sample $(a, b) = R_q \times R_q$ from the R-LWE distribution can replace n samples $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ from the standard LWE distribution. To see this, notice that the $(a, b = a \cdot s + e \bmod q) \in R_q \times R_q$ sample can be expressed in matrix notation as $(\mathbf{F}^*\mathbf{a}, \mathbf{b} = \mathbf{F}^*\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$, where $\mathbf{a}, \mathbf{b}, \mathbf{s}, \mathbf{e} \in \mathbb{Z}_q^n$ are vectors constructed by taking the polynomial coefficients from $a, b, s, e \in R_q$ (see Section 5.1 for a detailed example of this equivalence). We see that the matrix notation can be split into n LWE samples: $(\mathbf{F}^i\mathbf{a}, b^{(i)} = \mathbf{F}^i\mathbf{a} \cdot \mathbf{s} + e^{(i)}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ for $i \in \{0, \ldots, n-1\}$, where $b^{(i)}$ and $e^{(i)}$ are the ith coefficients of the vectors $\mathbf{b}$ and $\mathbf{e}$ respectively.

With this we can reduce the size of the public and the secret key in the schemes by a factor of n.

We also notice that the most expensive operation in R-LWE is the polynomial multiplication, which can be computed in $O(n \log n)$ as mentioned in Section 5.

6 Lattice-based public-key cryptography

6.1 Asymmetric encryption

A simple and efficient semantically secure lattice-based public-key cryptosystem is detailed in Section 7, along with a study of its decryption error probability, a review of proposals on choosing the parameters, and an analysis of the homomorphic properties of this scheme.

6.2 Digital signature

Several digital signature schemes have been presented in the recent years, both using general lattices and ideal lattices. The earliest proposals for lattice-based signatures were the GGH [GGH97] scheme and NTRUSign [HHPSW03], but unfortunately, both schemes can be broken in a strong asymptotic sense. A particular difference between asymmetric encryption and a signature scheme based on hard lattice problems is that usually, in an encryption scheme the secret is only used one time; whereas in signature schemes the secret is used every time a signature is generated. This was problematic for the earliest signature proposals: every signature exposes some information of the secret key and thus it can be recovered after analyzing a few signatures (by means of estimating their probability distribution).

New schemes have been recently proposed that fix this issue; for instance, BLISS [DDLL13], which has even been implemented in the StrongSwan open source VPN software [str05] [str15] , the very recent ring-TESLA [ABBKM16], or GLP [GLP12]. In particular, BLISS solves the problem of revealing secret information in the signature by controlling the signature probability distribution by means of repeating the signature process several times before returning the signature itself to modify the distribution (and thus remove any information about the secret key).

7 Ring-LWE encryption

7.1 Definition

Here we describe a simple and efficient semantically secure public-key cryptosystem. The selection of the ring must be carefully done so that the security of the scheme is guaranteed; for correctness in the security proofs (see [LPR13] for the details) and considering implementation optimizations using the Number Theoretic Transform, we fix the ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ with n a power of 2. All the arithmetic operations will be done in R_q, which is the previous ring where all coefficients are modulo q.

The original definition of the algorithm requires choosing 'small elements' in R from an error distribution at several points. For practical purposes we will construct these 'small elements' in R by taking their coefficients from an error distribution $\mathcal{X}$ which will be a discrete Gaussian distribution with parameter σ.

- **Key generation**

  Choose a uniformly random element $a \in R_q$ as well as two random 'small' elements $s, e \in R$ from $\mathcal{X}$. Output s as the secret key and the pair $(a, b = a \cdot s + e) \in R_q \times R_q$ as the public key.

- **Encryption**

  To encrypt an n-bit message $z \in \{0,1\}^n$, we view it as an element of R by using its bits as the 0-1 coefficients of a polynomial. The encryption algorithm then chooses three random 'small' elements $r, e_1, e_2 \in R$ from $\mathcal{X}$ and outputs the pair $(u, v) \in R_q \times R_q$ as the encryption of z, where

$$u = r \cdot a + e_1 \bmod q$$
$$v = b \cdot r + e_2 + \lfloor \frac{q}{2} \rceil z \bmod q$$

- **Decryption**

  The decryption algorithm simply computes

$$v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \lfloor \frac{q}{2} \rceil z \bmod q$$

For an appropriate choice of parameters (namely q and σ), the coefficients of $(r \cdot e - s \cdot e_1 + e_2) \in R$ have magnitude less than $q/4$, so the bits of z can be recovered by rounding each coefficient of $v - u \cdot s$ back to either 0 or $\lfloor \frac{q}{2} \rceil$, whichever is closest modulo q.

Notice that the cryptosystem can be extended to encrypt messages with elements bigger than a bit, i.e., $z \in \{0, 1, \ldots, k\}^n$. In order to do so, we will map the n-symbol message z to an element of R by scaling it with a factor of $\lfloor \frac{q}{2k} \rceil$, instead of $\lfloor \frac{q}{2} \rceil$. Finally, in the decryption step, the symbols of z can be recovered by rounding each coefficient of $v - u \cdot s$ back to $\lfloor \frac{q}{2k} \rceil$ for $i = \{0, 1, \ldots, k\}$ whichever is closest modulo q. The symbols will be properly decrypted whenever the coefficients of $(r \cdot e - s \cdot e_1 + e_2) \in R$ have magnitude less than $q/2k$.

The scheme is semantically secure following the pseudorandomness of Ring-LWE: recall from the definition of the Ring-LWE problem in Definition 11, Section 5.2 that differentiating Ring-LWE samples from uniform (random) samples is hard. Moreover, analogous to Lemma 2, Section 3.10, Ring-LWE samples are pseudorandom even when the secret is also chosen from the error distribution, by a transformation to the normal form. Therefore, the public key $(a, b) \in R_q \times R_q$ is pseudorandom, so that without loss of generality, we may replace it with a truly uniform pair. Then we see that the pairs $(a, v), (b, v) \in R_q \times R_q$, which constitute the entire view of a passive adversary, are Ring-LWE samples with secret r and hence also pseudorandom (considering that the addition of the message component $\lfloor \frac{q}{2} \rceil z$ modulo q doesn't change the pseudorandomness of the Ring-LWE sample), which implies semantic security.

7.2 Decryption error probability

The decryption procedure for Ring-LWE consists in computing the following

$$v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \lfloor \frac{q}{2k} \rceil z \bmod q$$

Where we consider the error term to be

$$err = (r \cdot e - s \cdot e_1 + e_2) \in R$$

Which will cause a decryption error when a coefficient of the err polynomial exceeds $\frac{q}{2k}$ where k is the symbol size (i.e., each symbol has $log_2(k)$ bits) used in the message z.

We now consider that we are working under the ring $R = \mathbb{R}[x]/\langle x^n + 1 \rangle$ with n a power of 2.

As we have seen before in Section 5, the multiplication of two polynomials $a, b \in R_q$ is equivalent to the product of the matrix $\mathbf{A} = \mathbf{F}^* \mathbf{a} \in \mathbb{Z}_q^{n \times n}$ with the vector $\mathbf{b} \in \mathbb{Z}_q^n$ modulo q, where the vectors $\mathbf{a}$ and $\mathbf{b}$ are created with the obvious embedding from R_q to $\mathbb{Z}_q^n$, that is, the coefficients of the polynomial are taken as the coefficients of a vector.

When working on the ring of cyclotomics we have the property that if the coefficients of $\mathbf{a}$ follow a symmetric distribution probability (such as a zero-centered discrete Gaussian distribution as it will be in our case), all the elements in all the columns of $\mathbf{A}$ follow the same exact distribution. This happens because the operator $\mathbf{F}$ corresponds to working with the class of anti-cyclic lattices and thus performs a shift on the vector and negates the shifted coefficient.

This way, if the coefficients of $\mathbf{a}$ follow a discrete Gaussian distribution with a parameter σ, all the elements in $\mathbf{A}$ will follow a discrete Gaussian distribution with the same parameter σ as well.

From this observation we can express the distribution of each coefficient of err as:

$$X_{err} = \sum_{i=1}^{n} X_r^i X_e^i - \sum_{i=1}^{n} X_s^i X_{e_1}^i + X_{e_2}$$

Where n is the number of coefficients of the polynomials (or equivalently the dimension of the ideal lattice), and each X_*^i is the discrete Gaussian variable with parameter σ corresponding to the coefficients of each polynomial. We can see, as we already know, that each coefficient of the polynomial that results from

multiplying two polynomials a and b in R_q is computed naively by adding n multiplications of pairs of coefficients from a and b. This is easily seen if we see the operation in the lattice embedding.

Now, considering the fact that all X_*^i follow the same symmetric distribution with a known standard deviation, we can find the exact standard deviation of X_{err}.

Let X and Y be two independent random variables following a zero mean symmetric distribution with standard deviation σ_X^2 and σ_Y^2, respectively:

$$\begin{aligned} \mathbb{E}[X] = \mathbb{E}[Y] = 0 \\ X = -X \\ Y = -Y \\ XY = -XY \end{aligned}$$

$$\sigma_X^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \mathbb{E}[X^2]$$

$$\sigma_{XY}^2 = \mathbb{E}[X^2Y^2] - \mathbb{E}[XY]^2 = \mathbb{E}[X^2Y^2] = \mathbb{E}[X^2]\mathbb{E}[Y^2] = \sigma_X^2\sigma_Y^2$$

$$\begin{aligned} \sigma_{X+Y}^2 = \mathbb{E}[(X+Y)^2] - \mathbb{E}[X+Y]^2 = \mathbb{E}[X^2 + 2XY + Y^2] = \\ \mathbb{E}[X^2] + 2\mathbb{E}[XY] + \mathbb{E}[Y^2] = \mathbb{E}[X^2] + \mathbb{E}[Y^2] = \sigma_X^2 + \sigma_Y^2 \end{aligned}$$

Thus, if we apply the formulas to X_{err} we obtain that

$$\sigma_{X_{err}}^2 = 2n\sigma^4 + \sigma^2$$

We can now use the fact that n will be a large enough value (we use values of 256 or 512 for practical instantiations of Ring-LWE in cryptographic schemes) to approximate X_{err} as a zero-centered Gaussian distribution with parameter $\sigma_{X_{err}}$ following the central limit theorem because X_{err} is the addition of $2n + 1$ independent variables.

We will have a decryption error on a symbol whenever its corresponding coefficient of the err polynomial is bigger than $q/2k$. The probability of this event happening is analyzed here, where P_{de} is the probability that a symbol from the message z is erroneously decrypted:

$$P_{de} = Pr\left[|X_{err}| > \frac{q}{2k}\right] = 2Pr\left[X_{err} > \frac{q}{2k}\right] = 1 - 2Pr\left[0 < X_{err} < \frac{q}{2k}\right] = 1 - 2\left(Pr\left[X_{err} < \frac{q}{2k}\right] - \frac{1}{2}\right) = 1 - 2\left[CDF_{\sigma_{X_{err}}}\left(\frac{q}{2k}\right) - \frac{1}{2}\right]$$

Where CDF_σ is the cumulative distribution function of a Gaussian with parameter σ.

Whereas the previous equation allows us to compute the decryption error probability per symbol given some Ring-LWE parameters, it makes it hard to reason about the relation between the parameters and the obtained error. To improve this situation we can rearrange the formula substituting the random variable X_{err} by a standard normal random variable. Before that, we give an approximation to the standard deviation of X_{err}:

$$\sqrt{2n\sigma^4 + \sigma^2} \approx \sqrt{2n\sigma^4} = \sqrt{2n}\sigma^2$$

Considering our constraints in our parameters, we argue that the error introduced by this approximation is very small considering that we are providing the $s = \sigma\sqrt{2\pi}$ parameter with 4 decimals:

$$n \geq 256$$
$$\sigma \geq \frac{8}{\sqrt{2\pi}}$$
$$\left|\frac{\sqrt{2n\sigma^4 + \sigma^2} - \sqrt{2n\sigma^4}}{\sqrt{2n\sigma^4 + \sigma^2}}\right| = \left|1 - \frac{\sqrt{2n\sigma^2}}{\sqrt{2n\sigma^2 + 1}}\right| \leq 9.589 \times 10^{-5}$$

Now we do the random variable substitution, obtaining:

$$N(0,1) \leftarrow Y$$
$$Y = \frac{X_{err}}{\sqrt{2n}\sigma^2}$$
$$P_{de} = Pr\left[|X_{err}| > \frac{q}{2k}\right] = Pr\left[|Y| > \frac{q}{2k\sqrt{2\pi}\sigma^2}\right]$$

Now we can see that to maintain a fixed decryption error probability per symbol, we need to keep the factor $z_\alpha = \frac{q}{2k\sqrt{2\pi}\sigma^2}$ constant. Moreover, to find different

sets of parameters that give the same error probability P_{de}, we just need to compute the critical value of the normal distribution (i.e., z_α s.t. $Pr\left[|Y| > z_\alpha\right] = P_{de}$) once, by means of a numerical approximation of the Gauss error function (erf) required to compute the cumulative distribution function of the standard normal distribution.

Finally, we analyze the probability of error when decrypting a message, P_{me}; that is, the probability that at least one symbol in the message gets decrypted erroneously. We use err_i as the event that the symbol i in a given message is decrypted erroneously ($Pr[err_i] = P_{de}$).

$$P_{me} = Pr[err_1 \cup err_2 \cup \cdots \cup err_n] \leq$$
$$Pr[err_1] + Pr[err_2] + \cdots + Pr[err_n] = n \cdot P_{de}$$

Where we have used the union bound ($Pr[A \cup B] \leq Pr[A] + Pr[B]$) to bound the error probability per message.

A different approach for estimating the probability of decryption error per symbol can be found in [LP11]. In that case, the authors use two lemmas that bound the tail of a discrete Gaussian distribution to find a non-tight upper bound on the decryption error probability in an LWE encryption scheme. From their result, the authors propose a procedure to obtain the parameters of the encryption scheme given the upper bound in the decryption error probability.

7.3 Choosing the parameters

The authors of the paper "On the concrete hardness of Learning with Errors" [APS15] provided a Sage [Sag] module for estimating the bit security of LWE instances [APGS15], in order to provide designers with an easy way to choose parameters that resist known attacks and also to enable comparisons of different cryptanalysis for a given set of parameters.

With the help of this module we are able to provide proposals of parameters for practical uses in Ring-LWE encryption intended for asymmetric encryption only (i.e., no homomorphic operations) where we are able to achieve the desired security level and decryption error probability. Note however that these parameters will be defined based on the best attacks to *LWE*. It is still unknown if attacks can take advantage of the special structure of ring lattices using a proper set of parameters.

For homomorphic operations, an upper bound on the number of operations that will be applied to the ciphertexts must be provided because this value will determine the decryption error after the operations. This is because the error probability keeps growing after every homomorphic operation (see Section 7.4), so the parameters q and s must be set accordingly. Another option for homomorphic operations is to take advantage of the results of fully homomorphic encryption; particularly the bootstrapping algorithm, which is a procedure to reduce the error in the ciphertext after a number of homomorphic operations. This later option is out of the scope of this thesis.

We can find two proposals of methods to choose *LWE* parameters in the literature that consider the requirements of security reductions [Reg09] and an upper bound on the decryption error probability [LP11]. But these proposals don't consider all the known attacks, and may give unreasonable parameters for some settings. For an implementation of LWE oracles generating samples according to these proposals see [LS15].

Unfortunately, the runtime and memory consumption of the LWE known attacks are estimations that require numerical computations (sometimes a bit intensive). This means that we are unable to provide a general formula to achieve some desired characteristics, but we are left with the tedious task to manually tune and adjust the parameters repeatedly until we verify that the desired level of security is reached according to the estimations from the Sage module.

Recall that in schemes relying on *LWE* we will have samples of the form $(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, with e drawn from a discrete Gaussian distribution with standard deviation σ.

The list of *LWE* parameters is the following:

- n: dimension of the lattice.
- q: modulo used in the operations (this also defines the modulo in the q-ary lattice).
- s: parameter for the Gaussian error.
- $\sigma = s/\sqrt{2\pi}$: standard deviation for the Gaussian error.
- $\alpha = s/q$: spread of the Gaussian error relative to q.

7.3.1 Regev's procedure [Reg09]

Select n as the security parameter.

$$q = \text{next\_prime}(n^2)$$

$$s = \frac{q}{\sqrt{n} \cdot log_2^2 n}$$

7.3.2 Linder and Peikert's procedure [LP11]

Select n as the security parameter, and δ as the upper bound on decryption probability error per symbol.

Solve the following equation for c, and compute s_t with the result

$$2 \cdot n \cdot log(c) + n(1 - c^2) + 40log2 = 0$$

$$s_t = \frac{\sqrt{2}\pi\sqrt{2 \cdot n \cdot log(2/\delta)}}{c}$$

The parameters will be

$$q = \text{next\_prime}(\lfloor 2 \cdot \lfloor log_2(256/s_t) \rceil \rfloor)$$

$$s = \sqrt{s_t \cdot \lfloor q/4 \rfloor}$$

7.4 Homomorphic sum in Ring-LWE

The proposed Ring-LWE scheme allows homomorphic addition without any modification, as seen in the following equations where the elements with super index (i) refer to some ciphertexts generated as defined in Section 7 and the elements with super index (s) refer to the summation of the (i) elements, that is, the result of adding the ciphertexts.

We define a public key from a secret key

$$a, b = a \cdot s + e$$

Each ciphertexts will have the form

$$u^{(i)} = a \cdot r^{(i)} + e_1^{(i)}$$
$$v^{(i)} = b \cdot r^{(i)} + e_2^{(i)} + \lfloor \frac{q}{2k} \rceil z^{(i)}$$

And so, the summation of ciphertexts will be

$$u^{(s)} = \sum_i u^{(i)} = \sum_i a \cdot r^{(i)} + \sum_i e_1^{(i)} = a \sum_i r^{(i)} + \sum_i e_1^{(i)}$$

$$r^{(s)} = \sum_i r^{(i)} \qquad e_1^{(s)} = \sum_i e_1^{(i)}$$

$$v^{(s)} = \sum_i v^{(i)} = \sum_i (b \cdot r^{(i)} + e_2^{(i)} + \lfloor \frac{q}{2k} \rceil z^{(i)}) =$$
$$b \sum_i r^{(i)} + \sum_i e_2^{(i)} + \lfloor \frac{q}{2k} \rceil \sum_i z^{(i)}$$

$$r^{(s)} = \sum_i r^{(i)} \qquad e_2^{(s)} = \sum_i e_2^{(i)} \qquad z^{(s)} = \sum_i z^{(i)}$$

So the decryption of the result ends up being

$$v^{(s)} - u^{(s)} \cdot s = (r^{(s)} \cdot e - s \cdot e_1^{(s)} + e_2^{(s)}) + \lfloor \frac{q}{2k} \rceil z^{(s)} =$$
$$\sum_i (r^{(i)} \cdot e - s \cdot e_1^{(i)} + e_2^{(i)}) + \lfloor \frac{q}{2k} \rceil \sum_i z^{(i)}$$

It can be seen that the summation of Ring-LWE ciphertexts, each one with its own randomness but using the same public key, results in a new Ring-LWE ciphertext that contains the summation of the plaintext underlying the summed ciphertexts.

From the equation results we also see that after performing the summations, the error of all the original ciphertexts is added in the final ciphertexts. This means that the error grows linearly with the number of operands in the summation. For this reason, in the design of an encryption system where we wish to use the homomorphic properties of the Ring-LWE scheme, we should know in advance the maximum number of elements that will be summed in order to find the proper parameters (namely s and q) that allow those many operations while keeping the final decryption error probability upper bounded to a small (or negligible) value. From this we also see that, intuitively, if we want to target an upper bound of the final decryption error probability, the q parameter will grow at least linearly with the maximum number of operations that we support, with the consequence that the keys and ciphertexts would grow with the same factor. Notice however that

this is just a general idea: keeping the same n and s while growing q may lower the security of the scheme, so further analysis would need to be done in this sense, which we leave as future work.

8 Implementation of Ring-LWE encryption

8.1 NFLlib

NFLlib [ALGG15] is an efficient NTT-based open-source C++ library dedicated to ideal lattice cryptography. It is specialized in elements from the ring of polynomials modulo a cyclotomic whose degree is a power of two of the form $R = \mathbb{Z}[x]/\langle x^n+1\rangle$ with n a power of 2. The library combines algorithmic optimizations (Chinese Remainder Theorem, optimized Number Theoretic Transform) together with programming optimization techniques (SSE and AVX2 specializations, C++ expression templates, etc.).

The library provides an API to generate elements of the ring R that will work in modulo q (for arbitrarily large q) in the following three ways: by manually specifying the coefficients, by drawing the coefficients from a uniform distribution given a range, by drawing the coefficients from a discrete Gaussian distribution with a given parameter σ. The specific selection of the q parameter is internal to the library in order to be able to apply the Chinese Remainder Theorem optimizations to hold arbitrarily large coefficients; the user is only able to select the number of bits used for the q parameter.

Internally, the library has precomputed tables of prime numbers (along with related values needed to operate with the Chinese Reminder Theorem) that are the size (in bits) of CPU words minus 2 bits; specifically: 14, 30 and 62. The user can then select multiples of these values as the size in bits of q, which will be generated by a product of primes (each one the size of a word minus 2 bits). With this decomposition, the library is able to perform all the operations in the Chinese Reminder Form, allowing each congruence for each polynomial coefficient to be stored in a single CPU word, which in turn allows fast arithmetic operations: there is no need to use any "big num" library. In fact, the library works with the Chinese Reminder Form all the time except when the full polynomial coefficients are requested. This optimization has as a consequence the fact that the library doesn't always work with q prime.

From the generated polynomials, the library offers the following operations: coefficient-wise sum, coefficient-wise product, number theoretic transform, and

inverse number theoretic transform (the later two operations are needed to perform the standard product between polynomials efficiently).

8.2 Parameters proposal and analysis

To propose LWE parameters to use in our implementation based on NFLlib, we fix n, q and a probability of decryption error per symbol and then we do a binary search using the approximation formulas shown in Section 7.2 to obtain the s parameter for the discrete Gaussian distribution. Notice that the proposed q parameters are some of the ones available in NFFlib; as mentioned before, the library only allows us to select the number of bits but not the value itself. The procedure to obtain s in such a way can be found in function `find_s` at Appendix B.1.

In Table 1 we show a comparison of the LWE parameter proposals found in [LP11] with our proposals, with the s parameter computed as mentioned above. The last column of the table shows the estimated decryption error probability using our approximation implemented in Appendix B.1.

name	n	q	s	α	P(err)
192_LP	192	4093	8.8700	0.002167	0.0031%
256_LP	256	4093	8.3500	0.002040	0.0046%
320_LP	320	4093	8.0000	0.001955	0.0072%
256_14	256	15361	16.5554	0.001078	0.0100%
256_30	256	1073479681	4376.4140	0.000004	0.0100%
512_14	512	15361	13.9214	0.000906	0.0100%
512_30	512	1073479681	3680.2387	0.000003	0.0100%
256_14'	256	15361	14.7648	0.000961	0.0001%
256_30'	256	1073479681	3903.1101	0.000004	0.0001%
512_14'	512	15361	12.4155	0.000808	0.0001%
512_30'	512	1073479681	3282.1790	0.000003	0.0001%

Table 1: Comparison of parameters proposals for use in Ring-LWE schemes with estimated error probability, where LP refers to the proposals from the work of Linder and Peikert [LP11] and the rest are proposals from this thesis.

The Sage module provided by [APS15] will estimate the cost of different attacks on the given parameters of an LWE instance. The main procedure gives us

various parameters for each attack, which can be divided into three groups: memory cost (mem), the number of LWE oracle calls needed to perform the attack (smp) and different forms of runtime costs. Depending on the attack, the runtime cost may be expressed as the number of binary operations (bop), the runtime cost of the BKZ 2.0 lattice reduction algorithm (bkz2) along with the number of calls to the sieve algorithm (sieve)[5] and the number of extra enumerations when exploring the possible decodings (enum).

The attacks analyzed by the module are the following: Meet-in-the-Middle (MitM), BKW, distinguishing attack (SIS), decoding attack (Dec) and Kannan. The decoding attack is the one explained in Section 4.2 with further improvements found in [LP11] and [LN13]. To learn more about the rest of the attacks we refer the reader to [APS15] where they are described in detail with references to the publications where they first appeared.

In Table 2 we show the cost estimates for different attacks given by the Sage module on the LWE parameter proposals found in [LP11] as well as our parameter proposals.

	MitM			Coded-BKW			SIS			Dec			Kannan		
name	bop	mem	smp	bop	mem	smp	bkz2	sieve	smp	bop	enum	smp	bkz2	sieve	smp
192_LP	314	311	7	103	88	87	123	107	25	91	75	16	109	95	14
256_LP	404	401	8	128	112	111	190	141	29	121	105	23	180	128	15
320_LP	493	489	8	141	124	124	260	174	36	153	137	29	264	163	15
256_14	531	527	8	132	116	115	180	136	27	117	101	21	168	123	15
256_30	1562	1557	8	173	155	154	98	92	18	83	67	13	86	84	15
512_14	986	983	9	230	213	212	489	264	46	240	224	30	536	260	16
512_30	3047	3042	9	681	664	663	301	185	27	168	152	18	287	173	16
256_14'	510	506	8	132	116	115	174	132	25	114	98	21	158	119	15
256_30'	1541	1536	8	173	155	154	95	90	18	81	64	17	83	83	15
512_14'	944	940	9	216	199	198	469	256	44	232	216	29	509	251	16
512_30'	3005	3000	9	681	664	663	293	182	27	166	150	15	277	170	16

Table 2: Analysis of attacks to the LWE problem for different sets of parameters. All the values are in log_2 scale.

Considering the high number of LWE oracle calls required for most of the attacks, they couldn't be used against certain schemes where not enough LWE samples are exposed (for instance, consider the Ring-LWE scheme explained in

[5]The sieve is a method to solve the exact SVP and it's used in the BKZ 2.0 algorithm as a subroutine, while working in smaller dimensions.

Section 7 where only n samples are exposed in the public key, and $2n$ samples are exposed in the ciphertext).

Even though the proposals 256_30 and $256\_30'$ seem weaker than the rest, considering the SIS, Dec and Kannan attacks, it's hard to estimate the security parameter because we don't know whether the attacks could be adapted to use a lower number of samples (at the cost of increasing their runtime), and if so, how worse they would perform, considering that the current trade off would have been optimized for performance. Moreover, most of the work in the literature focuses on attacks on LWE and other lattice problems, leaving the cryptanalysis of specific lattice-based schemes untouched, which corresponds to a very specific instance of the problem and can't be generalized. Nevertheless there is literature on cryptanalysis on the older lattice-based encryption scheme called NTRU. From that work for example, Bernstein et. al. are able to provide a security estimate for their proposal NTRU Prime [BCLV16]. It remains a future work for the research community to cryptanalize specific Ring-LWE schemes.

8.3 Benchmarking

In order to evaluate the performance of Ring-LWE encryption a C++ implementation has been developed using the NFLlib library [ALGG15]. The cryptographic scheme is implemented as described in Section 7. The full source code of the implementation can be found in the publicly available git repository [San16].

The source code available at the git repository includes error checking procedures in various functions, queries to a monotonic clock to benchmark the speed of the different operations and counting of the decryption error rates for a big number of encryption/decryption operations of random plaintexts. To ease the understanding of the source code, a cleaned version that only contains the essential functions to perform key generation, encryption and decryption can be found in the Appendix A.

For the analysis of performance of the cryptographic operations of the implementation the proposed parameters from Section 7.3 have been used.

The results of the benchmark can be seen in the Table 3, where "msg err" refers to the percentage of decryption errors per ciphertext, "sym err" refers to the percentage of decryption errors per symbol, and "pk size" and "sk size" refers to the public key and private key sizes respectively.

name	key gen	encrypt	decrypt	msg err	sym err	pk size	sk size
256_14	165 μs	218 μs	125 μs	2.8300%	0.0113%	7.0 kb	3.5 kb
256_30	200 μs	271 μs	135 μs	2.7340%	0.0109%	15.0 kb	7.5 kb
512_14	365 μs	456 μs	264 μs	5.1440%	0.0104%	14.0 kb	7.0 kb
512_30	415 μs	575 μs	261 μs	5.1600%	0.0104%	30.0 kb	15.0 kb
256_14'	165 μs	217 μs	126 μs	0.0220%	0.0001%	7.0 kb	3.5 kb
256_30'	198 μs	268 μs	125 μs	0.0300%	0.0001%	15.0 kb	7.5 kb
512_14'	336 μs	450 μs	280 μs	0.0440%	0.0001%	14.0 kb	7.0 kb
512_30'	404 μs	546 μs	264 μs	0.0760%	0.0001%	30.0 kb	15.0 kb

Table 3: Benchmarking results for the Ring-LWE scheme, taken from 50000 runs using random plaintexts.

9 E-voting

Electronic voting (also known as e-voting) consists of adapting some (or all) of the steps involved in a voting process to use electronic mechanisms. The definition ranges from aiding in the process of vote counting to implementing a full-functioning online voting system that allows people to vote from their homes. In general, some of the advantages of electronic voting are: speeding the counting of ballots, reducing the cost of paying staff to count votes manually and provide improved accessibility for disabled voters as well as better commodity to regular voters.

An e-voting scheme must comply with the standards and laws established by regulatory bodies, and must be capable of fulfilling strong requirements associated with security, integrity, privacy, auditability among others. Some of these requirements can be achieved by means of cryptographic protocols.

9.1 Voter's anonymization

One of the main features desired in an e-voting system is the anonymity of the voters: despite the fact that during the process we will verify that each accepted ballot comes from an authorized voter, we want the relation between the voter and the contents of their ballot to remain secret.

To achieve this, there have been two main proposals in the literature:

- **Tally homomorphic**: This approach takes advantage of encryption schemes that support homomorphic operations (addition in this case). In this form,

each voter will send their encrypted ballot (probably with a signature to prove the voter's authenticity to a server). The encrypted ballots will then be summed to generate a new ciphertext, which will then be decrypted by a trusted authority with its secret key, to reveal the ballot count results. Notice that at some point before the summation of the encrypted ballots, the validity of the selected options underlying each encrypted ballot must be checked; for example, ballots containing multiple answers to a single answer question must be rejected.

- **Mix-nets**: A different approach is to add a "scrambling" operation in the process before counting. The encrypted ballots will be stripped from their signature, and then re-encrypted and randomly permuted by one or several mix-nets, with the purpose of removing the relation between the casted encrypted ballot and the encrypted ballot that will be used for the counting. Once the encrypted ballots have been mixed, they are decrypted individually to count the results by a trusted authority. Notice that this method requires the mix-nets to provide a plaintext equality proof so that the correctness of their scrambling process, i.e., that the contents of their output are the same as the contents of their input, can be verified by some entity.

The concept of using mix-nets for lattice-based cryptography is very new in the research literature, and as such, there are not many proposed schemes. Nevertheless, there has been extensive research on the general topic of mix-nets with applications on e-voting. For instance, there is the work by Jens Groth [Gro10] that provides a mix-net scheme for general homomorphic encryption schemes (so that it could be adapted to work with Ring-LWE encryption), with further improvements on the zero-knowledge proofs of knowledge sizes in [BG12] and [CKLM12]. There is also a parallel work by Douglas Wikström [Wik11] which improves on efficiency by doing an offline commitment on the permutation before the ciphertexts are received.

This thesis only gives an introduction to the elements required to adapt an e-voting scheme to make it quantum resistant, and so, providing anonymity for the voters is left as future work.

9.2 E-voting at Scytl

The voting scheme implemented by Scytl[6] uses the mix-net strategy to provide anonymity properties for the voters.

Apart from the standard cryptographic properties desired in an e-voting scheme, Scytl provides a novel property called "cast as intended", which allows the voter to verify that the ballot server received the vote they selected without the need to trust the software running in the voting machine (which may be a personal computer). In order to achieve this property, a set of return codes are used as explained later.

The actors that participate in the voting procedure are the following: *the* N *voters*, *the ballot server*, *the mix-net*, *the counter* and *the key generator*.

The ballot must be defined in terms of M questions with Q_j answers each question.

The following indexes will be used:

- $i \in [1, \ldots, N]$ to refer to a voter i.
- $j \in [1, \ldots, M]$ to refer to ballot question j.
- $k \in [1, \ldots, Q_j]$ to an answer k of the ballot question j.

The voting procedure consists of several steps:

1. *Ballot definition*

   For each answer k of each question j, a unique prime number $p_{j,k}$ different than 1 is assigned. These prime numbers will be public.

2. *Key Generation*

   This step must be performed by a trusted entity: *the key generator*.

   First, a random Login Key LK of sufficient length is generated for each voter. This key can be split into two smaller keys: the voter's ID (ID_i) and the voter's password (PWD_i).

   The following asymmetric key pairs are generated:

   - PK_b, SK_b: ElGamal key pairs for ballot encryption.
   - PK_{v_i}, SK_{v_i} with $i \in [1, \ldots, N]$: RSA key pairs for voter signature.

[6] https://www.scytl.com/

The following secret keys are generated:

- SK_{rc_i}: secret keys used in the return code procedure for each voter.

Additionally, a procedure is defined to derive a symmetric key KK_i from the voter's password PWD_i.

A set of return code tables will be generated for each question j in the ballot, which will contain two columns: long code and return code. The long code is computed by $LC_{i,j,k} = HMAC(p_{j,k}^{SK_{rc_i}}, ID_i)$ for each possible answer j of every question k and each voter i. The return code ($RC_{i,j,k}$) is computed by expressing the long code in decimal form and taking the last 4 digits. There will be M return code tables of length $N \sum_{j=1}^{M} Q_j$ each, containing the long code and return codes for the answers k corresponding the each question j. In each return code table, the entries will be scrambled to avoid revealing the voter i and the answer k of any entry.

SK_{v_i} and SK_{rc_i} are symmetrically encrypted with the KK_i key (derived from PWD_i) producing the ciphertext KS_i which we call the keystore.

Finally, SK_b is given to *the counter*. The set of ID_i, PK_{v_i}, KS_i and the M return code tables are given to *the ballot server*.

3. *Voting Card generation*

   For each voter, a voting card must be generated and be sent in a confidential manner. For example, it could be printed on paper and sent via conventional mail.

   The voting card must contain the following information:

   - Login Key LK_i
   - For each question j
     - For each answer k that belongs to the question j: return code $RC_{i,j,k}$

   See Figure 8 for an example with some sample questions.

4. *Voting*

   The voter sends a request to *the ballot server* with their ID_i to retrieve the keystore KS_i. If the voter sent their unique ID_i, they will be able to

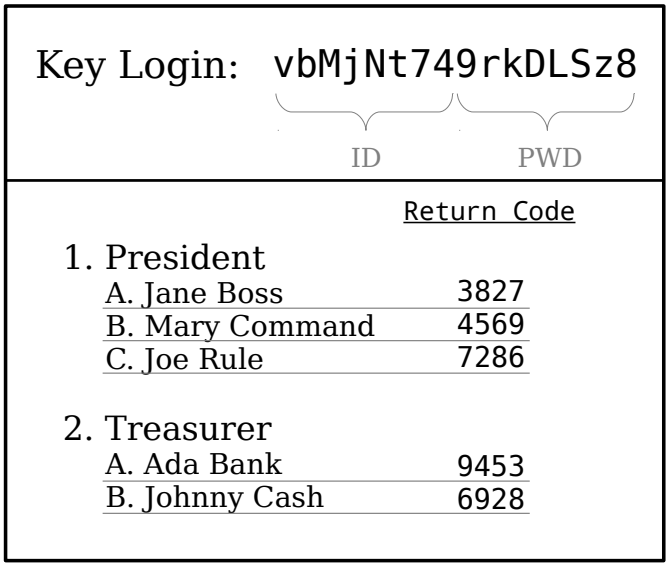


Figure 8: Voting Card example.

decrypt KS_i using the key KK_i derived from PWD_i to obtain SK_{v_i} and SK_{rc_i}. Since the tuple ID_i and PWD_i is unique and secret for each voter, no voter can decrypt the keystore of another voter.

The voter, with the help of the voting software, proceeds to select the prime numbers $p_{j,k}$ associated to their desired answers to form the ballot. On one hand, the voter will encrypt the product of the selected primes $p_{j,k}$ with ElGamal using the key PK_b, generating the ciphertext V_i, and then send it to *the ballot server*. On the other hand, the voter will encrypt each prime $p_{j,k}$ deterministically using SK_{rc_i}, generating the ciphertext $A_{i,j}$. The voter sends these ciphertexts along with a signature (Sig_i) of them using the key SK_{v_i}.

The ballot server will first verify the authenticity of the vote by checking the signature Sig_i against the public key PK_{v_i} associated with the ID_i. Secondly *the ballot server* will verify that the contents of the vote V_i are the same as the elements $A_{i,j}$, using an equality proof of knowledge. If the verification succeeds, *the ballot server* computes the corresponding long codes $LC_{i,j,k}$ of the elements $A_{i,j}$ and verifies that they are in the proper tables, sends the associated return codes $RC_{i,j,k}$ to the voter and stores the en-

crypted vote V_i with it's signature Sig_i. By computing the proof of knowledge and checking that the long codes are in the correct tables (i.e., the tables corresponding to each question j), *the ballot server* verifies that the received vote is valid (i.e., each question has only one answer). After validating all the votes and verifying their authenticity, the ballot server can sign the concatenation of all the votes. On the other hand, the voter is able to check that their intended options have been received successfully and without modification to the *ballot server* by checking that the received return codes match the ones listed in the voting card for the intended options. Furthermore, there's an additional step not detailed here for succinctness in which the voter sends a final confirmation to the *ballot sever* after verifying the correctness of the return codes.

5. *Vote mixing*

   The mix-net will receive the encrypted votes V_i (with their corresponding signatures Sig_i detached) from *the ballot server* and will perform a random permutation in the order of the received votes as well as a blind re-encryption of the votes (V_i'). *The mix-net* must provide a proof of the correct permutation and re-encryption of the votes, that can be used to verify that the votes content hasn't been modified.

6. *Vote counting*

   The counter will receive the anonymous votes V_i' from *the mix-net*, for which the correctness can be verified by means of checking the correctness of the mix-net proof, which proves that those anonymous votes have the same contents as the input votes used by the mix-net, which are verified to be the votes casted by the voters by checking the vote server signature on their concatenation.

   The counter will then proceed to decrypt the anonymous votes V_i' and to obtain the contained answers by factorizing the plaintext (which will provide the primes $p_{j,k}$ associated with each answers). After decryption, the answers are counted to provide the results, which can be made public.

In the previous description, some simplifications have been made on how the voter receives the key store (KS_i) from the server in order to focus on the parts relevant to this thesis analysis (namely, an authenticated login mechanism for the voter before retrieving the key store has been skipped).

9.3 Post-Quantum proposal

For a recent proposal on a post-quantum e-voting scheme, as an alternative to the idea we propose in this thesis, we refer the reader to [CGGI16]. In the cited paper, the authors present an e-voting scheme that uses LWE fully homomorphic encryption in order to provide a tally homomorphic system. The scheme performs several bootstrapping operations on the encrypted ballots (which contain a single question with a single answer) to allow small sized ciphertexts to be sent by the voters while also allowing counting the results by using homomorphic addition. The authors also propose a new procedure to distribute the decryption task among trusted parties where each one provides independent proofs of the correct decryption. The presented work is rather theoretical and it lacks a proposal of specific parameters in the LWE instantiation, which would be needed to evaluate the size of the keys and ciphertexts, and more importantly, to verify that time needed to perform all the required bootstrapping operations and decryptions is reasonable.

In the following part of this section we will propose an adaptation of the current voting scheme used by Scytl to make it post-quantum.

The first observation we make to the Scytl scheme in its current implementation is that it doesn't allow vote repetition: the ballot cannot be changed once it has been sent. This characteristic comes as a legal requirement in some countries, so it's not always a design choice. For this reason, we propose a simplification of the return codes mechanism by adapting the vote encryption process to be deterministic. Since the voter is not allowed to vote more than once, the voter will only be able to encrypt a ballot once, so there is no security penalty with this modification.

We only give an initial idea of a possible proposal, leaving the anonymization of the votes as future work and skipping specific details of the signature procedure.

Our proposal consists in the following modifications:

- Instead of using ElGamal to encrypt the ballots, we would use the Ring-LWE encryption scheme described in Section 7. With this change, the key pairs PK_b, SK_b would now correspond to the Ring-LWE scheme.

- Instead of using RSA to sign the ballots, we would use a lattice-based signature scheme like the ones mentioned in Section 6.2. In this case we leave the selection of a specific digital signature scheme as well as its details and analysis for future work, as in this thesis we have focused on a public-key

encryption scheme. With this change, the key pairs PK_{v_i}, SK_{v_i} would now correspond to a lattice-based signature scheme.

- We remove the secret keys used in the return code procedure SK_{rc_i}.
- We define a "deterministic" randomness for each unique pair of voter and ballot question, for example, constructed like $RND_{i,j} = hash(PWD_i \| j)$ (where the operator $\|$ is the concatenation), so that each voter can obtain these values without modifying the fields found in the Scytl voting card.
- Since each answer is encrypted only individually now, the plaintext corresponding to each answer would now be a binary vector instead of a prime number. The vector (with size the number of answers for the given question) would have all elements set to 0 except for the element in the position corresponding to the selected answer, which would be set to 1.
- In the voting step, instead of encrypting the product of the prime numbers $p_{j,k}$ associated with the desired answer; *the voter* just encrypts, for every question k, the desired answer in vector form (as described above) individually using the Ring-LWE scheme with the key PK_b and randomness $RND_{i,j}$, thus making it a deterministic encryption, similarly to the second encryption part of the Scytl voting step. We call these encrypted answers $A_{i,j}$ for their similarity with the original scheme. On the other hand, during the key generation, the long codes would have been computed with all the possible $A_{i,j}$ encryptions for every voter, i.e., $LC_{i,j,k} = HMAC(Enc_{PK_b,RND_{i,j}}(v_{j,k}))$, where $Enc_{PK_b,RND_{i,j}}$ obviously refers to the Ring-LWE encryption using key PK_b and randomness $RND_{i,j}$, and $v_{j,k}$ is the answer vector of question j and selected answer k.

  With this modification we are able to remove the plaintext equality proof in the return code procedure: the return codes are now generated with the encrypted answers that will form the ballot used in the counting, which are now deterministic, and so *the ballot server* can still verify the validity of the ballot and *the voter* can verify that *the ballot server* has received the intended answers.
- Now *the voters*' ballot that gets stored in *the ballot server* would be split into one ciphertext for every question. *The voter* would sign the concatenation of the encrypted answers for authentication by *the ballot server*.

We refer to the benchmark results shown in Section 8.3 to argue that the sizes of keys and ciphertexts are small enough and that the encryption/decryption speeds are fast to show that the modifications in our proposal are completely reasonable and don't cause any penalty on the e-voting scheme.

As mentioned before, even though the anonymization part of the e-voting scheme is left as future work, we will give some comments on the tally homomorphic variant.

If tally homomorphic is desired, a quick analysis following the results in Section 7.4 shows us that, should the encrypted ballots be naively summed by *the counter*, the size of keys and ciphertexts would grow linearly with the number of voters, so the scheme could become impractical because we would end up with keys and ciphertexts too big. A possible solution for this, which is out of the scope of this thesis, would be to leverage on the results of fully homomorphic encryption; namely using the bootstrapping process that allows a ciphertext to be re-encrypted without modifying the plaintext into a new ciphertext with a new (in this case bigger) q without requiring knowledge of the secret key. The bootstrapping could be performed on the *ballot server* to allow the voter to encrypt with a smaller q (and thus use smaller keys and send smaller ciphertexts) while still allowing *the counter* to operate on the encrypted ballots.

10 Conclusions

As we have seen, lattice theory offers a new framework to build cryptographic primitives. Being so recent in the field of cryptography means that there is still a lot of work to do. Most of the current literature is focused on theoretical results of lattice problems and scheme proposals are often shown rather briefly. Some of the practical issues that arise from those schemes are not entirely addressed, for instance, the problems that occur when working with an encryption scheme that has a non-negligible decryption error and how to solve them practically.

Focusing more on the details of specific schemes (a trend that is happening in recent publications) will help with the task of the much needed cryptanalysis of such proposed schemes, which is a bit lacking currently. Only attacks on the underlying hard problems are studied in the literature, and those may not always relate directly to attacking the proposed schemes.

More research and analysis on this topic (particularly, on specific schemes) is needed before any post-quantum algorithm is recommended for use. The current

proposals have not received nearly as much scrutiny from the cryptographic community as the currently deployed algorithms. Apart from building confidence in post-quantum cryptography, we also face the challenge to improve its efficiency and usability.

Current lattice-based scheme proposals, like the one described in this thesis, are still quite new; so we should proceed with caution. Nevertheless, we should be already thinking about proposing standards for lattice-based schemes if we want to replace current pre-quantum schemes. In that sense, it is expected that the National Institute of Standards and Technology of the USA (NIST) will soon announce a competition for post-quantum public-key schemes[7], where researchers will be able to propose their best schemes which will be analyzed in depth for some years by the cryptography community before determining a winner.

Regarding the public-key encryption scheme based on the Ring-LWE problem described in this work, as seen in Section 8.3, the benchmarking results are very promising: the speed of the operations is very fast and the key and ciphertexts sizes are very competitive. This makes it a proper candidate to replace current pre-quantum schemes. Moreover, thinking further on an e-voting scheme, where the number of transactions per users are very low (each voter only casts a ballot once, where the number of questions is usually small), even if the ciphertexts grow a bit the impact on the e-voting scheme is minimal.

Studying a post-quantum e-voting scheme proposal made us realize that if voting is only allowed once per voter, we can simplify the return code mechanism, reducing the complexity of the scheme without compromising any feature. From this fact, and considering the rest of our proposal in Section 9.3, we conclude that adapting a current e-voting scheme to make it post-quantum should be completely feasible without any apparent drawback.

[7] http://www.nist.gov/itl/csd/nist-kicks-off-effort-to-defend-encrypted-data-from-quantum-computer-threat.cfm

References

[ALGG15] Carlos Aguilar, Tancrède Lepoint, Adrien Guinet, and Serge Guelton. *NFLlib: NTT-based Fast Lattice library*. 2015. URL: https://github.com/quarkslab/NFLlib.

[ABBKM16] Sedat Akleylek, Nina Bindel, Johannes A. Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. "An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation". In: *Progress in Cryptology - AFRICACRYPT 2016 - 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings*. 2016, pp. 44–60.

[APGS15] Martin R. Albrecht, Rachel Player, Florian Göpfert, and Sam Scott. *Estimator for the Bit Security of LWE Instances*. 2015. URL: https://bitbucket.org/malb/lwe-estimator.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *J. Mathematical Cryptology* 9.3 (2015), pp. 169–203.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. 2009, pp. 595–618.

[Bab85] László Babai. "On Lovász' Lattice Reduction and the Nearest Lattice Point Problem (Shortened Version)". In: *STACS 85, 2nd Symposium of Theoretical Aspects of Computer Science, Saarbrücken, Germany, January 3-5, 1985, Proceedings*. 1985, pp. 13–20.

[BG12] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. 2012, pp. 263–280.

[BB84] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theor. Comput. Sci.* 560 (1984), pp. 7–11.

[Ber09] Daniel J. Bernstein. “Post-Quantum Cryptography”. In: ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Chap. Introduction to post-quantum cryptography, pp. 1–14.

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, eds. *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[BCLV16] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. *NTRU Prime*. Cryptology ePrint Archive, Report 2016/461. http://eprint.iacr.org/. 2016.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. “Malleable Proof Systems and Applications”. In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. 2012, pp. 281–300.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. “A Homomorphic LWE Based E-voting Scheme”. In: *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*. 2016, pp. 245–265.

[DGHV09] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *Fully Homomorphic Encryption over the Integers*. Cryptology ePrint Archive, Report 2009/616. http://eprint.iacr.org/. 2009.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. “Lattice Signatures and Bimodal Gaussians”. In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. 2013, pp. 40–56.

[GN08] Nicolas Gama and Phong Q. Nguyen. “Predicting Lattice Reduction”. In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. 2008, pp. 31–51.

[Gen09] Craig Gentry. “A fully homomorphic encryption scheme”. crypto.stanford.edu/craig. PhD thesis. Stanford University, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. “Homomorphic Evaluation of the AES Circuit”. In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. 2012, pp. 850–867.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. “Public-Key Cryptosystems from Lattice Reduction Problems”. In: *Advances in Cryptology - CRYPTO ’97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. 1997, pp. 112–131.

[Gro10] Jens Groth. “A Verifiable Secret Shuffle of Homomorphic Encryptions”. In: *J. Cryptology* 23.4 (2010), pp. 546–579.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. “Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems”. In: *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. 2012, pp. 530–547.

[HHPSW03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. “NTRUSIGN: Digital Signatures Using the NTRU Lattice”. In: *Topics in Cryptology - CT-RSA 2003, The Cryptographers’ Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*. 2003, pp. 122–140.

[KS99] Aviad Kipnis and Adi Shamir. “Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization”. In: *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. 1999, pp. 19–30.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. “Factoring polynomials with rational coefficients”. In: *Mathematische Annalen* 261.4 (1982), pp. 515–534.

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. 2011, pp. 319–339.

[LN13] Mingjie Liu and Phong Q. Nguyen. "Solving BDD by Enumeration: An Update". In: *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*. 2013, pp. 293–309.

[LS15] The Developers of LWE and Sage. *(Ring-)LWE oracle generators*. 2015. URL: http://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/lwe.html.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *J. ACM* 60.6 (2013), p. 43.

[MR09] Daniele Micciancio and Oded Regev. "Post-Quantum Cryptography". In: ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Chap. Lattice-based Cryptography, pp. 147–191.

[Pei09] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem: extended abstract". In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. 2009, pp. 333–342.

[Pei16] Chris Peikert. "A Decade of Lattice Cryptography". In: *Foundations and Trends in Theoretical Computer Science* 10.4 (2016), pp. 283–424.

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. 2005, pp. 84–93.

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *J. ACM* 56.6 (2009).

[Sag] The Developers of Sage. *SageMath, the Sage Mathematics Software System*. URL: http://www.sagemath.org.

[San16] Eduard Sanou. *C++ implementation of ring-LWE asymmetric encryption scheme using NFLlib*. 2016. URL: https://gitlab.com/dhole/ring-LWE.

[SE94] Claus-Peter Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems". In: *Math. Program.* 66 (1994), pp. 181–199.

[str05] The Developers of strongSwan. *strongSwan, the OpenSource IPSec-based VPN Solution*. 2005. URL: https://www.strongswan.org/.

[str15] The Developers of strongSwan. *Bimodal Lattice Signature Scheme (BLISS) implementation in strongSwan*. 2015. URL: https://wiki.strongswan.org/projects/strongswan/wiki/BLISS.

[Wik11] Douglas Wikström. "A Commitment-Consistent Proof of a Shuffle". In: *IACR Cryptology ePrint Archive* 2011 (2011), p. 168.

A Ring-LWE source code

```
using poly_type = nfl::poly_from_modulus<P_TYPE, N, BITS>;
using gauss = nfl::gaussian<uint8_t,
                            typename poly_type::value_type, 2>;
using fast_gauss = FastGaussianNoise<uint8_t,
                                     typename poly_type::value_type, 2>;

struct PublicKey {
    // $a$ and $b$ are stored in the NTT form for efficiency.
    poly_type a_i;
    poly_type b_i;
};

struct SecretKey {
    // $s$ is stored in the NTT form for efficiency.
    poly_type s_i;
};

struct Encryption {
    poly_type u;
    poly_type v;
};

size_t q = poly_type::get_modulus(0);
size_t q_1_2 = q/2 + 1;
size_t q_1_4 = q/4;
size_t q_3_4 = q/2 + q/4 + 1;

void
poly_from_bitvector(poly_type &p, std::vector<bool> &vec)
{
    // Since poly coefficients are 0 or 1, we can find them all at cm = 0
    for (size_t i = 0; i < p.degree; i++) {
        p(0, i) = (bool) vec[i];
    }
    return 0;
}

void
bitvector_from_poly(std::vector<bool> &vec, poly_type &p)
{
    for (size_t i = 0; i < p.degree; i++) {
        vec[i] = (bool) p(0, i);
    }
    return 0;
}

void
scale_poly_1_q2(poly_type &p)
{
    for (auto& it : p) {
        it *= q_1_2;
    }
}

```

```
55 void
56 scale_poly_q2_1(poly_type &p)
57 {
58     for (auto& it : p) {
59         if (it < q_1_4 || it > q_3_4) {
60             it = 0;
61         } else {
62             it = 1;
63         }
64     }
65 }
66
67 void
68 gen_key_pair(PublicKey &pk, SecretKey &sk, gauss &gauss_s)
69 {
70     pk.a_i = poly_type( (nfl::uniform()) );
71     sk.s_i = poly_type(gauss_s);
72     poly_type e(gauss_s);
73
74     pk.a_i.ntt_pow_phi();
75     sk.s_i.ntt_pow_phi();
76
77     pk.b_i = pk.a_i * sk.s_i;
78     e.ntt_pow_phi();
79     pk.b_i = pk.b_i + e;
80 }
81
82 void
83 encrypt(PublicKey &pk, std::vector<bool> &msg, Encryption &enc,
84         gauss &gauss_s)
85 {
86     poly_type zq2;
87     poly_from_bitvector(zq2, msg);
88     scale_poly_1_q2(zq2);
89
90     poly_type r(gauss_s);
91     poly_type e1(gauss_s);
92     poly_type e2(gauss_s);
93
94     r.ntt_pow_phi();
95     enc.u = pk.a_i * r;
96     enc.v = pk.b_i * r;
97     enc.u.invntt_pow_invphi();
98     enc.v.invntt_pow_invphi();
99     // No need to invntt a and b because they are local copies
100    enc.u = enc.u + e1;
101    enc.v = enc.v + e2 + zq2;
102 }
103
104 void
105 decrypt(SecretKey &sk, std::vector<bool> &msg, Encryption &enc)
106 {
107     // We use a copy of u because we will be transforming (ntt) it's values
108     // during the process.
109     poly_type u = enc.u;
110     poly_type zq2;
111
```

```
112     u.ntt_pow_phi();
113     zq2 = u * sk.s_i;
114     zq2.invntt_pow_invphi();
115     zq2 = enc.v - zq2;
116 
117     scale_poly_q2_1(zq2);
118 
119     bitvector_from_poly(msg, zq2);
120 }
```

rlwe.cpp

B Sage code

B.1 Error probability

```
def p_error(n, q, s, sym=2):
    # sym: number of symbols
    sigma0 = s/sqrt(2*pi.n())
    sigma = sqrt(2*n*sigma0**4 + sigma0**2)
    T = RealDistribution('gaussian', sigma)
    p_err = 1 - 2*(T.cum_distribution_function(q/(sym*2)) - 0.5)
    return p_err

def bin_search(target_y, min_x, max_x, func, eps = 0.001, iter=1000000):
    eps = target_y * eps
    for i in range(iter):
        m = min_x + (max_x - min_x) / 2
        y = func(m)
        if y <= max(target_y - eps, 0):
            min_x = m
        elif y > (target_y + eps):
            max_x = m
        else:
            return m.n()
    return m.n()

def find_s(n, q, p_err, max_s):
    func = lambda s: p_error(n, q, s)
    s = bin_search(p_err, 1, max_s, func)
    return s
```

err–prob.py

B.2 Babai's Nearest Plane

```
from sage.modules.misc import gram_schmidt

def nearest_plane(B, w0):
    # B: lattice basis, w0: target vector
    n = B[0].degree()
    w = matrix(QQ, n)
    y = matrix(ZZ, n)
    B1, mu = gram_schmidt(list(B))
    w[n-1] = w0
    for i in range(n)[::-1]:
        li = (w[i]*B1[i]) / (B1[i]*B1[i])
        y[i] = round(li)*B[i]
        if i != 0:
            w[i-1] = w[i] - (li - round(li)) * B1[i] - round(li)*B[i]
    return sum(y)
```

nearest–plane.py