Fast Bootstrapping in $\mathbb{Z}_q$

by

Luis Ruiz-Lopez

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In 2015, Ducas and Micciancio presented a novel technique to compute the NAND gate using the Learning With Errors cryptosystem (LWE), along with a novel bootstrapping technique that turns turns this cryptosystem into a fully-homomorphic encryption scheme that allows a very short and fast implementation. We present an extension of their bootstrapping technique that allows refreshing encryptions of elements in $\mathbb{Z}_p$ and the homomorphic computation of arbitrary gates, alongside with an implementation that exploits the power of parallel computation.

Acknowledgements

I want to first thank my supervisor David Jao for having taken me over, for having supported me during my masters, for having introduced me to cryptography and for all the patience he has had with me.

To my readers Alfred Menezes and Michele Mosca for their very useful comments.

To Jean-Francois Biasse for having supported and encouraged me, and all the time we spent working on this thesis.

A mis papás, Martha y Luis, por todo el apoyo que me han dado a distancia. A mis hermanos, Erick y Carlos, por tenerme presente.

To Nargiz.

To my friends Pavel, Nathan, Anirudh, Alan, Naye, Randy, Max, Sara, Annie, Arash, etc. for having made me have a great time here in Canada.

A mis eternos amigos Abdón, Héctor, Poke y Daniel, porque siempre los estaré recordando.

To *them*, who are always there...

Table of Contents

LIST OF TABLES

List of Tables

LIST OF FIGURES

List of Figures

Chapter 1

Introduction

Since the appearance of the Diffie-Hellman key exchange protocol in 1976 [9], cryptography has become increasingly sophisticated. Public-key cryptosystems and digital signature schemes were made possible shortly thereafter, using (a variant of, in the case of digital signatures) the well-known protocol invented by Rivest, Shamir and Adleman [33] in 1977, where two integers $N = pq$ and $e \in \{0, \ldots, \varphi(N) - 1\}$ are made public, and to encrypt an message $\mu \in \mathbb{Z}_N^*$ we compute $\mathsf{Enc}(\mu) = \mu^e$. A (probably) accidental feature of this protocol is the fact that it is multiplicative: given two messages $\mu_1, \mu_2 \in \mathbb{Z}_N^*$ we have that $\mathsf{Enc}(\mu_1\mu_2) = (\mu_1\mu_2)^e = \mu_1^e\mu_2^e = \mathsf{Enc}(\mu_1)\mathsf{Enc}(\mu_2)$. This led Rivest et al. [34] to rise the question of the existence of an encryption scheme that is able to evaluate arbitrary circuits (and not only multiplications of elements in $\mathbb{Z}_N^*$). In other words, they proposed an encryption scheme equipped with a function Eval such that, on input any circuit $\mathcal{C}$ of arity t and any messages $\mu_1, \ldots, \mu_t$, successfully outputs

$$\mathsf{Enc}\big(\mathcal{C}(\mu_1, \ldots, \mu_t)\big) \leftarrow \mathsf{Eval}\big(\mathsf{Enc}(\mu_1), \ldots, \mathsf{Enc}(\mu_t), \mathcal{C}\big).$$

The existence of such a cryptosystem is by no means obvious, but rather a difficult problem that, in some sense, might seem paradoxical, since looking to successfully operate on encrypted data is analogous to looking to successfully perform a task while blindfolded. However, 30 years later, Gentry [15] succeeded for the first time in constructing a cryptosystem able to homomorphically evaluate any given circuit.

Previous constructions are able to homomorphically evaluate more than just multiplications in $\mathbb{Z}_N^*$. Many of them [33], [18], [12] can handle several additions over the integers, and some of them [5][22], the so-called somewhat homomorphic encryption schemes, can even perform a few multiplications. The main obstacle for many of them was the noise growth:

the noise required for semantic security is added every time an addition is performed, and is multiplied every time a multiplication is performed. Gentry's idea was to add a second layer of encryption to evaluate the decryption circuit with the purpose of eliminating the first layer with its corresponding noise, resulting in a new "fresh" encryption of the same message. Note that in order to evaluate the decryption circuit we have to, in some way, hand the decryption key to an untrusted party, and then to keep the system secure we hand the key encrypted under a second key. Therefore there are some requirements on the somewhat homomorphic cryptosystem to turn it into a fully homomorphic one: it must be able to evaluate a little more than its own decryption circuit, and it must be safe to give away an encryption of the decryption key. This process is best known in the literature as bootstrapping, and has been the main subject of study around fully homomorphic encryption both to base its security on standardized problems, and to improve its running time, since it is the main bottleneck in the performance of Gentry's construction and all the other later constructions.

Several improvements have been made in fully homomorphic encryption in terms of security (basing its security on well studied problems such as Ring-LWE) and also in terms of its performance (decreasing its running time from hours to seconds). In order to optimize the amortized running time per gate, many authors approached the problem by increasing the depth capacity of their cryptosystems, allowing more homomorphic gate-operations before running a single expensive bootstrapping operation. Nevertheless, another approach is also possible. Ducas and Micciancio [11] designed a protocol that performs a cheap bootstrapping operation after every gate. This approach has the advantage that it is no longer the user's concern to know when and where to apply the bootstrapping procedure. It uses a different and more robust cryptosystem to add a second layer of encryption and evaluate the decryption circuit of the lighter one, improving with this its speed and versatility. Although it is a very fast algorithm, the amortized running-time per gate reported by Ducas and Micciancio is still not better than other implementations such as the Shoup-Halevi implementation HElib [19] and it has the limitation that it can only evaluate a very small family of gates before bootstrapping.

1.1 Our Contribution

We improve the fully homomorphic encryption scheme proposed by Ducas and Micciancio [11] in both theory and practice. From the theoretical point of view, we extend the capabilities of the homomorphic accumulator algorithm (thus named by the authors) used to perform the bootstrapping operation. This algorithm was previously only able to "re-

fresh" ciphertexts encrypting messages in $\mathbb{Z}_2$. We use a special property of the cyclotomic polynomial $\Phi_p(x)$, where p is a prime, to handle ciphertexts encrypting messages in $\mathbb{Z}_p$. This modification, together with other techniques, allows us to apply the bootstrapping procedure after any arbitrary gate, which makes the task of designing circuits much easier.

From the practical point of view, we improve the C++ implementation given by Ducas and Micciancio, by parallelizing the bootstrapping procedure, thus speeding up the evaluation time.

1.2 Outline Of The Thesis

The thesis is divided in 6 chapters. In Chapter 2 we give an introduction to lattices, describing the hard problems on which the security of lattice-based cryptosystems is based, the main existing algorithms to solve them, and the mathematical theory that is behind them.

In Chapter 3 we introduce the Learning With Errors cryptosystem and describe the properties of this system that will be used in the following chapters.

In Chapter 4 we give an introductory overview of fully homomorphic encryption and a more detailed description of Gentry's construction and idea.

In Chapter 5 we will describe in detail the construction of the fully homomorphic cryptosystem given by Ducas and Micciancio [11].

In Chapter 6 we describe our extension of the Ducas-Micciancio cryptosystem.

In Chapter 7 we report the benchmarks and running times of our modified implementation, as well as the main ideas behind how we modified the C++ code.

Chapter 2

Lattices

In geometry, a lattice is a collection of points forming a uniform periodic pattern that extends to infinity. Such a set of points exists in any euclidean space, independently of the dimension. For example, in dimension 1 we can think about the integer numbers, which are points on the real line forming a uniform periodic pattern easily obtained by "summing/subtracting" 1 every time. This pattern extends to infinity by summing/subtracting as many ones as necessary. In dimension 2, we can obtain a similar set by taking two vectors and repeatedly summing/subtracting either one of them to obtain a periodic pattern, as shown in Figure 2.

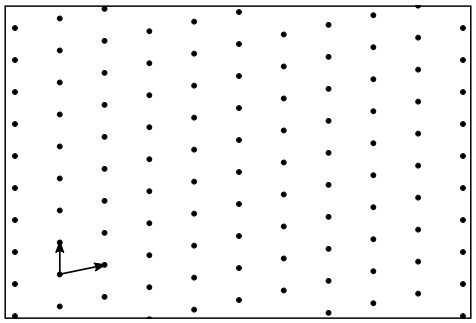


Figure 2.1: 2-dimensional lattice

In this chapter we give an brief introduction to lattices, explaining how they form an elegant bridge between geometry and algebra, and how cryptography uses this structure as a security guarantee for some of the most sophisticated cryptographic constructions.

2.1 Geometry of Numbers

From geometry we understand lattices as a uniform regular tiling of euclidean space. This regularity resembles the properties of the vectors with integer coordinates in $\mathbb{R}^n$; thus it is natural to try to describe the idea by saying that a lattice is a "skewed" version of $\mathbb{Z}^n$ in euclidean space. Thus, the idea of a lattice can be formalized with the following definition.

Definition 2.1.1 (Lattice)**.** Let $n, m \in \mathbb{Z}^+$ with $m \leq n$, and let $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$ be a collection of m linearly independent vectors. The *lattice* generated by $\mathbf{B}$ is the set

$$\mathcal{L} = \mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_m) = \left\{ \sum_{i=1}^{m} \xi_i \mathbf{v}_i : \forall i \in \{1, \ldots, m\}, \xi_i \in \mathbb{Z} \right\} \subseteq \mathbb{R}^n. \tag{2.1}$$

The set $\mathbf{B}$ is called a *basis* of the lattice. We say that $\mathbf{B}$ *generates* the lattice $\mathcal{L}$ and that $\mathcal{L}$ is of *rank* m. For $q \in \mathbb{Z}^+$, a lattice $\mathcal{L}$ is said to be *q-ary* if $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$. A *generating set* of a lattice is any set containing a basis as subset.

From the last definition we have that a lattice has an implicit group structure. In fact, a lattice can equivalently be defined as a discrete group in $\mathbb{R}^n$. On the other hand, a q-ary lattice has a richer structure, since having $q\mathbb{Z}^n$ as a subset implies that the lattice is "periodic modulo q", so all the information about the lattice can be obtained by looking at the residues of its entries modulo q. Thus, the q-ary lattices are in one-to-one correspondence with linear codes over $\mathbb{Z}_q$. These algebraic properties can be further exploited with the help of module theory and linear algebra. The following definition is motivated by the concept of a generator matrix in coding theory.

Definition 2.1.2 (Generator Matrix)**.** Given an (ordered) set of vectors $\mathbf{S} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$ we denote by

$$\mathbf{M_S} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_m \end{bmatrix}, \tag{2.2}$$

the matrix whose rows are the elements of $\mathbf{S}$. If $\mathbf{B}$ is a set of linearly independent vectors, the matrix $\mathbf{M_B}$ is called a *generator matrix* of the lattice $\mathcal{L}(\mathbf{B})$.

Definition 2.1.3 (Unimodular Matrices)**.** An integer matrix is said to be *unimodular* if its determinant is either 1 or -1.

A lattice $\mathcal{L}$ has, in general, several different bases. In particular, if $\mathbf{U}$ is a unimodular matrix then the matrix $\mathbf{UM_B}$ is also a generator matrix for $\mathcal{L}(\mathbf{B})$. The converse result is also true: given two different generator matrices of a lattice, one can be obtained from the other by multiplying by a unimodular matrix. This gives us the following proposition.

Proposition 2.1.4. *Let $\mathbf{B}, \mathbf{B}' \subseteq \mathbb{R}^n$ be two sets of linearly independent vectors. Then the lattices $\mathcal{L}(\mathbf{B})$ and $\mathcal{L}(\mathbf{B}')$ are equal if and only if there exists a unimodular matrix $\mathbf{U} \in \mathbf{M}_n$ such that $\mathbf{M_B} = \mathbf{UM_{B'}}$.*

The previous proposition tells us, in particular, that the determinant of a lattice is well defined, up to multiplication by -1. Therefore the notion of determinant can be, in a way, extended to lattices.

Definition 2.1.5 (Determinant of a Lattice)**.** Let $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subseteq \mathbb{R}^n$ be a collection of linearly independent vectors. The *determinant* of the lattice $\mathcal{L}(\mathbf{B})$ is the absolute value of the determinant of the generator matrix $\mathbf{M_B}$, i.e.

$$\mathrm{Det}(\mathcal{L}) = |\,\mathrm{Det}(\mathbf{M_B})|. \tag{2.3}$$

Given a lattice $\mathcal{L} \in \mathbb{R}^n$ of rank n and a basis $\mathbf{B}$, the basis determines a tiling of the $\mathbb{R}^n$ into parallelotopes, all of which are congruent to the parallelotope whose edges are parallel to the elements of $\mathbf{B}$. Moreover, it is well known that the absolute value of the determinant of $\mathbf{M_B}$ is the volume of this parallelotope. Consequently, by Proposition 2.1.4, we have that the parallelotopes determined by any two bases have the same volume. Hence, the *volume* of a lattice can be defined as

$$\mathrm{Vol}(\mathcal{L}) = \mathrm{Det}(\mathcal{L}). \tag{2.4}$$

An interesting question is "what is the smallest parallelotope that we can find in a lattice?". Note that this question is very easy to answer if we think of size in terms of the volume, since every fundamental region has the same volume, however, it is more interesting if we think of the size of the edges of the parallelotope. Intuitively, edge length defines the "most compact" tiling determined by the lattice. In the context of coding theory, this tiling is very useful for decoding. The following definition helps to formalize the previous idea.

Definition 2.1.6 (Successive Minima)**.** Let $n \in \mathbb{Z}^+$ and let $\mathcal{L}$ be a lattice on $\mathbb{R}^n$. Let Q be a quadratic form defined over $\mathbb{R}^n$ and let

$$B_1 = \{x \in \mathbb{R}^n : Q(x) \leq 1\}.$$

For $i \in \{1, \ldots, n\}$ we define the ith *successive minimum* to be

$$\lambda_i(\mathcal{L}) = \inf \Big\{ \lambda \in \mathbb{R} : \mathrm{Dim}\,(\lambda B_1 \cap \mathcal{L}) \geq i \Big\}. \tag{2.5}$$

We define the *minimal parallelotope* to be the parallelotope defined by a set of vectors that achieve the successive minima values, that is, a set $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ such that for all $i \in \{1, \ldots, n\}$ we have $|\mathbf{v}_i| = \lambda_i$.

2.2 Hard Lattice Problems

The problem of finding the successive minima is intuitively easy because when we think about a lattice, we imagine the set of all the vectors in the lattice, and then to find the ones that form the smallest parallelotope, it is enough to look for the set of independent vectors closest to the origin. However, that would require knowledge of all the vectors in the lattice, which is computationally unattainable. Even saving those vectors whose norm is bounded by a constant would require an exponential amount of memory. Nonetheless, a lattice is uniquely determined by the elements of a basis, therefore storing a basis is enough to completely describe it. In this case, finding the minimal parallelotope becomes a really hard problem.

The Shortest Vector Problem Finding the minimal parallelotope requires finding the shortest vector in the lattice. This, by itself, is a very important problem in geometry of numbers.

Definition 2.2.1 (The Shortest Vector Problem (SVP))**.** Let $\mathcal{L}$ be a lattice and let $\mathbf{B}$ be a basis for $\mathcal{L}$. The *shortest vector problem* (SVP) is the problem of finding a vector $\mathbf{v} \in \mathcal{L}$ such that $|\mathbf{v}| = \lambda_1(\mathcal{L})$.

The shortest vector problem depends on the norm that we are considering. The complexity of the problem may also vary depending on the norm. It is well known that SVP is an NP-Hard problem for randomized reductions when we consider the euclidean norm [2]. However, the problem is NP-Hard in the traditional sense if we consider the infinity norm, which is defined for $\mathbf{v} = (v_1, \ldots, v_n)$ as $||\mathbf{v}||_\infty = \max\{|v_1|, \ldots, |v_n|\}$[38]. Some other variants of the problem can also be considered.

Definition 2.2.2 (The Approximate Shortest Vector Problem (SVP_γ))**.** Let $\mathcal{L}$ be a lattice and let $\mathbf{B}$ be a basis for $\mathcal{L}$. Let $\gamma \geq 1$ be an approximation factor. The *approximate shortest vector problem* (SVP_γ) is the problem of finding a vector $\mathbf{v} \in \mathcal{L}$ such that $0 < |\mathbf{v}| \leq \gamma\lambda_1(\mathcal{L})$.

Definition 2.2.3 (The Gap Shortest Vector Problem (GapSVP_γ))**.** Let $\mathcal{L}$ be a lattice and let $\mathbf{B}$ be a basis for $\mathcal{L}$. Let $\gamma \geq 1$ be an approximation factor and $B > 0$ be a real constant. The *gap approximate shortest vector problem* (GapSVP_γ) is the problem of deciding whether $\lambda_1(\mathcal{L}) \leq B$, in which case the output is `TRUE`; or if $\lambda_1(\mathcal{L}) > B\gamma$, in which case the output is `FALSE`.

In cryptography it is commonly accepted that SVP_γ is a hard problem when γ is polynomially bounded. Later in this thesis we describe a polynomial-time algorithm that approximates SVP within exponential factors.

The Closest Vector Problem Another interesting problem is to find the distance from an arbitrary vector in the space to the lattice. This problem is a natural generalization of the problem of decoding in coding theory, where the task is to find the closest codeword to the received message.

For a set $\mathbf{S} \subseteq \mathbb{R}^n$ and an element $\mathbf{w} \in \mathbb{R}^n$ define

$$\text{Dist}(\mathbf{S}, \mathbf{w}) = \inf\{|\mathbf{v} - \mathbf{w}| : \mathbf{v} \in \mathbf{S}\}. \tag{2.6}$$

Definition 2.2.4 (The Closest Vector Problem (CVP))**.** Let $\mathcal{L}$ be a lattice, let $\mathbf{B}$ be a basis for $\mathcal{L}$ and let $\mathbf{w} \notin \mathcal{L}$ be a target vector. The *approximate closest vector problem* (CVP) is the problem of finding a vector $\mathbf{v} \in \mathcal{L}$ such that $|\mathbf{v} - \mathbf{w}| = \text{Dist}(\mathcal{L}, \mathbf{w})$.

The closest vector problem can be seen as a generalization of the shortest vector problem: naively we can think that we are trying to find the closest vector to the origin, but the origin is part of the lattice. However, it is well known that SVP can be efficiently reduced to CVP [27] (which implies that CVP is also a hard problem). Similarly we can also consider some other variants of this problem.

Definition 2.2.5 (The Approximate Closest Vector Problem (CVP_γ))**.** Let $\mathcal{L}$ be a lattice, let $\mathbf{B}$ be a basis for $\mathcal{L}$ and let $\mathbf{w} \notin \mathcal{L}$ be a target vector. Let $\gamma \geq 1$ be an approximation factor. The *approximate closest vector problem* (CVP_γ) is the problem of finding a vector $\mathbf{v} \in \mathcal{L}$ such that $0 < |\mathbf{v} - \mathbf{w}| \leq \gamma \,\text{Dist}(\mathcal{L}, \mathbf{w})$.

Definition 2.2.6 (The Gap Closest Vector Problem (GapCVP_γ))**.** Let $\mathcal{L}$ be a lattice, let $\mathbf{B}$ be a basis for $\mathcal{L}$ and let $\mathbf{w} \notin \mathcal{L}$ be a target vector. Let $\gamma \geq 1$ be an approximation factor and let $B > 0$ a real constant. The *approximate closest vector problem* (GapCVP_γ) is the problem of deciding whether $\text{Dist}(\mathcal{L}, \mathbf{w}) \leq B$ in which case the output is `TRUE`; or if $dist(\mathcal{L}, w) > B\gamma$, in which case the output is `FALSE`.

In cryptography, these problems are considered to be hard when the approximation factor is a polynomial. It is known that the approximation version of both problems are in NP $\cap$ coNP when the approximation factor is, at least $\sqrt{n}$. However it is unlikely that they are NP-Hard for those approximation factors. In the next section we describe the main algorithms that are used to solve the previously defined problems.

2.3 Lattices and Cryptography

In the previous section we presented some of the most important computational problems in the context of lattices: the shortest vector problem (SVP) and the closest vector problem (CVP). Both problems and their corresponding variants have been observed to be hard, even in the context of quantum algorithms. In this section we describe how lattice problems can be used as security guarantees for cryptographic constructions, specifically for a hash function introduced by Ajtai [1]. He proved that inverting this hash function is computationally as hard as solving classical lattice problems (such as SVP_γ) in the worst case.

The Short Integer Solution Problem Suppose that we are given the following system of linear equations over $\mathbb{Z}_q$.

$$\begin{array}{rcccc} a_{1,1}v_1 + a_{1,2}v_2 + \ldots + a_{1,m}v_m & = & 0 & \pmod q \\ a_{2,1}v_1 + a_{2,2}v_2 + \ldots + a_{2,m}v_m & = & 0 & \pmod q \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1}v_1 + a_{n,2}v_2 + \ldots + a_{n,m}v_m & = & 0 & \pmod q. \end{array}$$

Suppose that $n \leq m$. Since the system is consistent (because it is homogeneous) it has a solution that is easy to find by using Gaussian elimination. However, if we restrict ourselves to find a "small" nonzero solution, the problem is much more difficult.

Definition 2.3.1 (The Short Integer Solution Problem (SIS))**.** Let $n, m, q, s \in \mathbb{Z}^+$ with $s < q$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The *short integer solution problem* is the problem of finding $\mathbf{v} \in \mathbb{Z}^m$ with $0 < |\mathbf{v}| \leq s$ and such that $\mathbf{A}\mathbf{v}^T = \vec{0} \pmod q$.

Ajtai's One-Way Function In 1996, Ajtai published the seminal paper *Generating Hard Instances of Lattice Problems* where he presents a reduction from SIS to worst-case hardness of SVP_{n^c}, for some constant $c > 0$ (which has been proven to be very close

to 1 [28]). Ajtai also used the SIS problem as a security guarantee for a cryptographic construction for the first time (destructive applications were previously found in [37])

Definition 2.3.2 (Ajtai's One-Way Function)**.** Let $n, m, q, d \in \mathbb{Z}^+$ with $n < m$, $d \leq q$ and $q \in O(\log n)$. *Ajtai's one-way function* is the function $f\colon \mathbb{Z}_q^{n\times m} \times \{0,\dots,d-1\}^m \to \mathbb{Z}_q^n$ that takes $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ uniformly at random and is defined for a vector $\mathbf{v} \in \{0,\dots,d-1\}^m$ by

$$f_{\mathbf{A}}(\mathbf{v}) = \mathbf{A}\mathbf{v}^T. \tag{2.7}$$

Note that this function is very easy to implement, since it only involves additions and multiplications modulo q (which is in $O(n)$), so there is no need for "big numbers". However, the memory requirements are much bigger than for other cryptographic constructions, since the size of the keys is $O(mn)$.

2.4 Solving Hard Lattice Problems

Since we are using the difficulty of solving lattice problems as a security guarantee for cryptographic constructions, we need to know the best existing algorithms to solve them in order to establish the security parameters. In this section we give an introductory review of two of the most important approaches to solve SVP_γ and CVP_γ.

2.4.1 Lattice Reduction

The difficulty of solving lattice problems such as CVP_γ or SVP_γ can vary, depending on how "good" of "bad" the given basis is. For instance, suppose that the lattice consists of all the integer vectors in $\mathbb{R}^n$ and we are given, as basis, the standard basis[1]. Then the solution for SVP is immediate and CVP can be solved by just rounding the coordinates of the target vector. A similar procedure can be used if we are given a basis consisting of "short" and "close to orthogonal" vectors. Note that, in such a basis, the volume of the parallelotope is very close to the product of the lengths of the vectors in the basis. With this idea in mind we can think of a basis as "good" if

$$\frac{\prod_{i=1}^n |\mathbf{v}_i|}{\text{Vol}(\mathcal{L})} \tag{2.8}$$

[1]The *standard basis* in $\mathbb{R}^n$ is the set of vectors $\{\mathbf{e}_1,\dots,\mathbf{e}_n\}$ where $\mathbf{e}_i$ is the vector whose ith entry is 1, and the rest are all equal to 0.

is not much larger than 1. However, when trying to solve SVP, the length of the shortest vector of the basis is a better parameter to take into account. In [14], Gama and Nguyen reported an extensive experiment on which they determined that given an algorithm that takes a lattice $\mathcal{L}$ of dimension n, and outputs a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ (in non-decreasing order with respect to their norm), the *root Hermite factor* defined as

$$\delta = \left(\frac{|\mathbf{v}|^{1/n}}{\mathrm{Vol}(\mathcal{L})} \right)^{1/n} \tag{2.9}$$

is very similar for most lattices and bases (thus declared as an invariant of the algorithm), and a good indicator of how good an algorithm is.

Recall that the Gram-Schmidt process finds an orthonormal basis in polynomial time, starting with a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of the vector space. This is done by subtracting from $\mathbf{v}_j$ its projection $\mathbf{v}_j^*$ on the space generated by $\{\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}\}$, which is a linear combination of these vectors with coefficients

$$\mu_{i,j} = \frac{\langle \mathbf{v}_i, \mathbf{v}_j^* \rangle}{\langle \mathbf{v}_j^*, \mathbf{v}_j^* \rangle}, \text{ for } i \in \{1, \ldots, j-1\}. \tag{2.10}$$

These are called the *Gram-Schmidt coefficients*.

Lenstra-Lentra-Lovász In 1982, Arjen Lenstra, Hendrik Lenstra and László Lovász [24] introduced an algorithm to approximate the shortest vector of a lattice. The special feature of this algorithm is that it runs in polynomial time in the bit size of the vectors and the dimension of the lattice. The algorithm works in a similar way as the Gram-Schmidt process, and outputs a reduced basis, that is a basis consisting of "small" and "more orthogonal" vectors. The precise notion of what reduced means in the context of Lenstra, Lenstra and Lovász is given by the following definition.

Definition 2.4.1 (LLL-Reduced Basis)**.** Let $n \in \mathbb{Z}^+$, let $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subseteq \mathbb{Z}^n$ a set of linearly independent vectors, and let $\mathbf{B}* = \{\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*\}$ the basis resulting from the Gram-Schmidt process applied to $\mathbf{B}$. Let $\mathcal{L} = \mathcal{L}(\mathbf{B})$ be the lattice generated by $\mathbf{B}$. The basis $\mathbf{B}$ is said to be *size-reduced* if for all $i, j \in \{1, \ldots, n\}$ with $i < j$

$$|\mu_{i,j}| \leq \frac{1}{2}. \tag{2.11}$$

The basis $\mathbf{B}$ is said to be LLL-*reduced* for a parameter $\tau \in \left(\frac{1}{4}, 1\right]$ if

1. it is size-reduced.

2. for all $i \in \{2, \ldots, n\}$ the following equation holds

$$\left|\mathbf{v}_{i-1}^*\right|^2 \leq \frac{1}{\tau}\left(\left|\mathbf{v}_{i-1}^*\right|^2 + \mu_{i,i-1}^2 \left|\mathbf{v}_{i-1}^*\right|^2\right). \tag{2.12}$$

The last condition is called the *Lovász condition*.

An LLL-reduced basis is then understood as a "short" and "orthogonal" basis. The following definition describes an efficient algorithm to compute an LLL-reduced basis. In the literature, the parameter τ is usually denoted with the letter δ, but this would cause a conflict with the Hermite factor notation (Equation 2.9), so we use τ instead. This parameter is used to adjust the running-time/precision of the algorithm: the higher the parameter is, the more precise the result will be, at the cost of increased running time.

Definition 2.4.2 (The Lenstra-Lenstra-Lovász Algorithm (LLL))**.** Let $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$ be a set of vectors. Without loss of generality, assume that $\mathbf{B}$ is linearly independent. Let $\mathbf{B}^* = \{\mathbf{v}_1^*, \ldots, \mathbf{v}_m^*\}$ be the result of applying the Gram-Schmidt process to $\mathbf{B}$ and $\mu_{i,j}$ the (i,j)-Gram-Schmidt coefficient. The *Lenstra-Lenstra-Lovász* algorithm is the following

algorithm.

Algorithm 1: The Lenstra-Lenstra-Lovász Algorithm

Input : Basis $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$, parameter τ.
Output: LLL reduced basis $\mathbf{B}$.

```
B* ←GramSchmidt(B);
k ← 2;
while k ≤ n do
    for j ← 1 to k − 1 do
        v_k ← v − ⌊μ_{k,j}⌉ v_j;
    end
    if ||v*_k||² ≥ (τ − μ²_{k,k−1})||v*_{k−1}||² then
        k ← k + 1;
    else
        u ← v_k;
        v_k ← v_{k−1};
        v_{k−1} ← u;
        k ← max(k − 1, 2);
    end
end
return B;
```

The following theorem states the correctness of the LLL algorithm and says exactly what its running time is. Observe that the running time is polynomial in the number of bits necessary to describe the basis. However, from Definition 2.4.1 we can only conclude that the shortest vector $\mathbf{v}_1$ of the output satisfies

$$|\mathbf{v}_1| \leq \left(\frac{2}{\sqrt{4\tau - 1}}\right)^{n-1} \lambda_1(\mathcal{L}). \tag{2.13}$$

Theorem 2.4.3. *Let* $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^n$ *and suppose that* $\mathbf{v}_n$ *is a longest vector in* $\mathbf{B}$. *The Lenstra-Lenstra-Lovász algorithm computes an* LLL*-reduced basis for* $\mathcal{L}(\mathbf{B})$ *in time*

$$O\big(m^5 n (\ln \mathbf{v}_n)^3\big). \tag{2.14}$$

The LLL algorithm can be understood as a fast-but-bad approximation for SVP. Nevertheless, it is well known that, in practice, it performs much better in both aspects: running time and quality. In fact it can even be used as a SVP oracle for small dimensions. This is very important for the development of other algorithms that use LLL as a subroutine to obtain a much better quality of the output.

(Block) Korkine-Zolotarev

The objective of a lattice reduction algorithm is to obtain a basis that is close to the minimal parallelotope (see Definition 2.1.6), that is, a basis consisting of short and close to orthogonal vectors. The LLL algorithm runs in polynomial time, but, the quality of the result is not good enough for many applications (such as attacking cryptosystems). Following the opposite philosophy we can try to find a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ that achieves a good quality without caring too much about the running time. Assume for a moment that we have access to an n-dimensional SVP oracle (such an assumption holds in low dimensions as we mentioned before). Suppose that the oracle outputs $\mathbf{v}_1$ for the lattice $\mathcal{L}$. By projecting the lattice onto its orthogonal space $\mathbf{v}_1^{\perp}$, we reduce the dimension of the problem by one. Let $\tilde{\mathbf{v}_2}$ be the output of the oracle for the projection of $\mathcal{L}$ onto $\mathbf{v}_1^{\perp}$. Then $\tilde{\mathbf{v}_2}$ gives us a good idea of the second successive minimum of the original lattice (since this can be easily lifted to the original lattice by adding some small multiple of $\mathbf{v}_1$). Following this idea we obtain a *Korkine-Zolotarev reduced basis*, but the algorithm described is very inefficient, since it requires n calls to the SVP oracle, hence it is as hard as solving SVP in dimension n.

A different and more efficient approach comes when we combine the two previously described algorithms. In 1991, Schnorr and Euchner [36] described an algorithm that is a running-time/quality trade-off. It runs the Korkine-Zolotarev algorithm (the one previously described) on blocks of size k (that are thought to be small), using practical algorithms (such as LLL) to solve SVP on those blocks, projecting the lattice onto the orthogonal space of the block and applying LLL reduction again. The efficiency and quality of the algorithm is regulated by k: for $k = n$ we have the Korkine-Zolotarev algorithm, and for $k = 1$, we have the LLL algorithm, therefore the bigger the block size is, the stronger the reduction is. This process is known as the *block Korkine-Zolotarev algorithm* (BKZ), and it is the lattice reduction algorithm that is used to measure the security of lattice-based cryptosystems. The algorithm outputs a lattice basis whose shortest vector satisfies

$$|\mathbf{v}_1| \leq (1+\varepsilon)\gamma_k^{\frac{n-1}{2k-1}} \operatorname{Vol}(\mathcal{L})^{\frac{1}{n}}. \tag{2.15}$$

There is not a good known upper bound on the complexity of BKZ. An exponential bound was given in [20], but, in practice it performs very well for relatively small block sizes (being very practical for $k \leq 20$ and significantly decreasing in efficiency for $k \geq 25$ [7]). Hanrot et al. [20] give a more detailed analysis of a simplified variant the algorithm, observing that the most important changes in the basis occur during the first steps, and achieving a basis $\mathbf{B}$ whose shortest vector satisfies

$$|\mathbf{v}_1| \leq 2\gamma_k^{\frac{n-1}{2k-1}+\frac{3}{2}} \operatorname{Vol}(\mathcal{L})^{\frac{1}{n}}. \tag{2.16}$$

Notice that this bound is only slightly worse than Equation 2.15.

2.4.2 Enumeration Algorithms

Lattice-reduction algorithms help to find better quality bases and shorter vectors, but an additional routine is necessary to solve CVP. Given a basis $\mathbf{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ and a target vector $\mathbf{w}$, we can reduce $\mathbf{w}$ modulo $\mathbf{B}$ to obtain a vector $\tilde{\mathbf{w}}$ inside the parallelotope determined by $\mathbf{B}$. Projecting $\mathbf{w}$ to the closest facet of the parallelotope that is not parallel to $\mathbf{v}_n$ we reduce the problem by 1 in dimension. When reverting the reduction (with respect to $\mathbf{B}$) in the final solution, we obtain an element in the lattice that, naively, we expect to be close to $\mathbf{w}$. This approach is referred in the literature as *Babai's nearest plane algorithm* and clearly runs in polynomial time. Nonetheless, given an arbitrary basis, it does not guarantee any closeness of the result to the target vector. However, given a reduced basis it provides an exponential approximation.

Babai's nearest plane algorithm has been generalized by other authors (such as Pohst [31], Kannan [23] and Schnor-Euchner [36]) to other deterministic algorithms to solve the exact version of CVP, with exponential running time. It can also be generalized to randomized algorithms that run in polynomial time and provide better approximations to the closest vector with high probability.

Chapter 3

Learnings With Errors

The cryptographic applications of lattices that we have presented so far are limited to hash functions: the one-way function introduced by Ajtai [1] in his seminal paper *Generating Hard Instances of Lattice Problems* published in 1996. The first cryptosystem was introduced shortly after, in 1997, by Ajtai and Dwork [3]. The security of this cryptosystem was based on the difficulty of solving the worst case of a variant of SVP. However, one year later, Stern and Nguyen published an attack that made the cryptosystem impractical for real-life applications. In the same year, Goldreich, Goldwasser and Halevi (GGH) [17] proposed a cryptosystem based on lattices, which is an analog to the earlier McEliece cryptosystem that is based on linear codes. In 1998, a construction based on q-ary lattice that arise naturally from ideals over rings called NTRU was published by Hoffstein, Pipher and Silverman [21], representing the first practical cryptosystem based on lattices. However, neither GGH nor NTRU have a proof of security that ensures that breaking the cryptosystem is at least as hard as solving some lattice problem.

Almost a decade after, in 2005, Oded Regev [32] presented a problem related to the minimum distance decoding problem, a well known difficult problem that arises in coding theory. In this paper he gave a (quantum) reduction between this generalization (which he called learning with errors) and the decisional variant of SVP (GapSVP). Using this reduction as a security guarantee, one naturally obtains public-key and symmetric-key cryptosystems which form the basis for many cryptographic constructions based on lattices.

In this chapter we give a brief introduction to the problem and the cryptosystems, which form the basis for the constructions given in Chapters 5 and 6. We start by giving the necessary mathematical background in Section 3.1.

3.1 Mathematical Background

In Chapter 2 we introduced lattices as one of the main objects of study of the Geometry of Numbers. However, in some cases lattices are embedded in number fields, thus enjoying extra properties. We start this chapter with a brief reminder of the essential properties of number fields of interest in the context of lattice-based cryptography. Afterwards we will give a brief review of the concepts of probability that we need, and we end the section with some results on random matrices over a cyclotomic ring.

3.1.1 Cyclotomic Fields

A *number field* is a field extension of $\mathbb{Q}$ of finite degree. Number fields can be obtained by adjoining elements in $\mathbb{C} \setminus \mathbb{Q}$. We are particularly interested in the extensions that we obtain by adjoining a root of unity.

Definition 3.1.1 (Roots of Unity)**.** Let $n \in \mathbb{Z}^+$. A number $\zeta \in \mathbb{C}$ is called an *nth root of unity* if $\zeta^n = 1$. If ζ is a root of unity and for all $j \in \{1, \ldots, n-1\}$ we have that $\zeta^j \neq 1$, then it is called a *primitive nth root of unity.*

Let $n \in \mathbb{Z}^+$ and consider the set $\mathbf{\Pi}_n = \{j \in \mathbb{Z}^+ : 1 \leq j < n \text{ and } \gcd(j, n) = 1\}$. We define the *Euler's totient function* to be the function $\varphi(n) = |\mathbf{\Pi}_n|$. It is well known that if

$$n = p_1^{d_1} \ldots p_k^{d_k}$$

then $\varphi(n)$ is given by

$$\varphi(n) = (p_1 - 1) \ldots (p_n - 1) p_1^{d_1 - 1} \ldots p_k^{d_k - 1}.$$

This function naturally counts the cardinality of the set of generators of $\mathbb{Z}_n$ (which is the same set as $\mathbb{Z}_n^*$).

The complex numbers have a natural multiplicative subgroup isomorphic to $\mathbb{Z}_n$, namely, the group of complex solutions to the equation $x^n = 1$. Note that the set of roots of unity forms a regular polygon on the complex plane. With this picture in mind the following result is intuitively evident.

Lemma 3.1.2. *Let $n \in \mathbb{Z}^+$ and let $\{\zeta_1, \ldots, \zeta_n\}$ be the set of nth roots of unity. Then $\sum_{j=1}^{n} \zeta_j = 0$.*

As mentioned before, the set of roots of unity in the complex numbers is completely characterized as the set of complex roots of the polynomial $x^n - 1$. On the other hand, the set of primitive roots of unity in the complex numbers can be characterized as the set of roots of the polynomials described by the following definition.

Definition 3.1.3 (The nth Cyclotomic Polynomial)**.** Let n be a positive integer. The *nth cyclotomic polynomial* is the unique irreducible polynomial $\Phi_n \in \mathbb{Z}[x]$ that divides $x^n - 1$ and, for all $m < n$, satisfies the property that Φ_n does not divide $x^m - 1$. Alternatively, if $\mathcal{P}_n \subseteq \mathbb{C}$ is the set of primitive nth roots of unity, then

$$\Phi_n(x) = \prod_{\zeta \in \mathcal{P}_n} (x - \zeta). \tag{3.1}$$

The previous definition does not provide any easy way to compute the coefficients of the cyclotomic polynomial. Moreover, from the equation 3.1 it is not clear that the polynomial even has integer coefficients. However, for our purposes, we are only interested in the case where n is a prime power. Then the cyclotomic polynomial has a nice form described by the following proposition.

Proposition 3.1.4. *Let p be a prime number and let d be a positive integer. Then the p^dth cyclotomic polynomial can be writen as $\Phi_{p^d}(x) = \sum_{j=0}^{p-1} x^{jp^{d-1}}$.*

Proof. Let $f(x) = \sum_{j=0}^{p-1} x^{jp^{d-1}}$. Since $\deg f = \deg \Phi_{p^d}$ and all the roots of Φ_{p^d} are different (i.e. Φ_{p^d} is a separable polynomial), it is only left to prove that all the roots of Φ_{p^d} are roots of $f(x)$ as well. Let ζ be a root of Φ_{p^d}. Note that, for $j \in \{0, \dots, p-1\}$, $\zeta^{jp^{d-1}}$ is a pth root of unity. Moreover, since ζ is a primitive root of unity, if $j \neq j'$ then $\zeta^{jp^{d-1}} \neq \zeta^{j'p^{d-1}}$. Hence, the set $\left\{\zeta^{jp^{d-1}} : 0 \leq j \leq p-1\right\}$ is the set of pth roots of unity. Therefore, using the Lemma 3.1.2 we have that $\sum_{j=0}^{p-1} \zeta^{jp^{d-1}} = 0$. □

Corollary 3.1.5. *Given a positive integer d, the cyclotomic polynomial Φ_{2^d} is given by*

$$\Phi_{2^d} = x^{2^{d-1}} + 1. \tag{3.2}$$

Definition 3.1.6 (The Cyclotomic Field)**.** Let $n \in \mathbb{Z}^+$ and let $\zeta \in \mathbb{C}$ be a primitive nth root of unity. The *nth cyclotomic field* is the field $\mathbb{Q}[\zeta]$.

Remark 3.1.7. Since the cyclotomic polynomial is irreducible, we can also define, the nth cyclotomic field to be the field $\mathbb{Q}[x]/\Phi_n(x)$.

Definition 3.1.8 (Ring of Integers in the Cyclotomic Field)**.** Let $n \in \mathbb{Z}^+$ and consider its cyclotomic field $\mathbb{Q}[x]/\Phi_n(x)$. The *ring of integers* of $\mathbb{Q}[x]/\Phi_n(x)$ is the quotient $\mathbb{Z}[x]/\Phi_n(x)$.

3.1.2 Probability Distributions

For a cryptosystem to be semantically secure the encryption procedure cannot be deterministic; some randomness has to be involved. The security and even the correctness of some cryptosystems depend on the probability distribution that is used. In several cases it is convenient to use a Gaussian distribution, since it is well-studied and easy to describe.

Definition 3.1.9 (Gaussian Distribution)**.** For $o, \theta \in \mathbb{R}$, the *Gaussian distribution* of *mean* o and *standard deviation* θ is the probability distribution with density given by

$$\psi_{o,\theta}(s) = \frac{1}{\sqrt{2\pi}\theta} e^{\frac{(s-o)^2}{2\theta^2}}. \tag{3.3}$$

Definition 3.1.10 (Moment-Generating Function)**.** Let X be a real-valued random variable. The *moment generating function* of X is the function $M_X \to \mathbb{R} \to \mathbb{R}$ defined by

$$M_X(s) = \mathbb{E}\Big[e^{sX}\Big]. \tag{3.4}$$

whenever the expectation function exists.

Definition 3.1.11 (Subgaussian Distribution)**.** Let X be a real-valued random variable. We say that X is *subgaussian* if there exists $\theta > 0$ such that for all $s \in \mathbb{R}$

$$M_X(s) \leq e^{\frac{\theta^2 s^2}{2}}.$$

A subgaussian distribution is a probability distribution that is dominated by a Gaussian. As a consequence, we have the following result.

Proposition 3.1.12. *Let X be a real-valued random variable and let $B > 0$. If $P(|X| < B) = 1$, then X is a subgaussian random variable of parameter $B\sqrt{2\pi}$.*

Theorem 3.1.13. *Let ψ, ψ' be two Gaussian distributions of means and standard deviations (o, θ) and (o', θ'), respectively. Then the sum $\psi + \psi'$ is a Gaussian distribution of mean $o + o'$ and standard deviation $\sqrt{\theta^2 + \theta'^2}$.*

3.1.3 Random Matrices Over a Cyclotomic Ring

In several cases it is very important to have a notion of the "size" of a matrix, in terms of how big its entries are. Later in the thesis we use random matrices to add some noise to certain ciphertexts, and therefore it is crucial to bound how big the entries of the matrix are. A natural way to formalize the notion of size is to see how big the images of the unitary vectors are when seen as linear functions.

Definition 3.1.14 (Induced Norm)**.** Let $\mathbb{K}$ be a real or complex field, let $d_1, d_2 \in \mathbb{Z}$ and suppose that $||\cdot||_{(1)}$ and $||\cdot||_{(2)}$ are norms over $\mathbb{K}^{d_1}$ and $\mathbb{K}^{d_2}$, respectively. The *induced norm* associated to them is the norm $||\cdot||\colon \mathbb{K}^{d_1 \times d_2} \to \mathbb{R}$ defined by

$$||\mathbf{A}|| = \sup\left\{||\mathbf{A}\mathbf{v}||_{(2)} : ||\mathbf{v}||_{(1)} = 1\right\}. \tag{3.5}$$

In the special case where the norms $||\cdot||_{(2)}$ and $||\cdot||_{(1)}$ are both Euclidean, the induced norm of $\mathbf{A}$ is called the *spectral norm*, and is denoted by $s_1(\mathbf{A})$.

As mentioned, the induced norm is defined for real and complex fields; however the field $\mathbb{Q}[x]/\Phi_n(x)$ field has several embeddings in $\mathbb{C}^n$. The *canonical embedding* $\iota\colon \mathbb{Q}[x]/(x^n - 1) \to \mathbb{C}^n$ is defined by

$$\iota : \alpha(x) \mapsto \big(\alpha(\zeta_1), \ldots, \alpha(\zeta_n)\big). \tag{3.6}$$

Notice that for any complex number ζ (not necessarily a root of unity) and for any $\alpha, \beta \in \mathbb{Q}[x]/(x^n - 1)$ we have that $\alpha(\zeta)\beta(\zeta) = \alpha\beta(\zeta)$. Therefore, since any nth degree polynomial is determined by its value on n different elements, we have that

$$\iota(\alpha \cdot \beta) = \iota(\alpha) \odot \iota(\beta), \tag{3.7}$$

where $\odot$ denotes the component-wise product. This identity is fundamental to prove the following proposition.

Proposition 3.1.15. *Let $R = \mathbb{Z}[x]/(x^n - 1)$. Let ψ be a Gaussian distribution of parameter θ and let $\mathbf{A} = (a_{i,j}) \leftarrow R^{d_1 \times d_2}$ be a matrix such that each of whose entries is sampled according to ψ. Then with overwhelming probability we have that*

$$s_1(\mathbf{A}) \leq \theta\sqrt{n}\; O\left(\sqrt{d_1} + \sqrt{d_2} + \omega\sqrt{n}\right)$$

The proof is completely analogous to the proof of Fact *6* of [10]. However, to have a similar bound for matrices over $\mathbb{Q}[x]/\Phi_n(x)$, where $n = p^d$ is a prime power, we first lift the matrix to a matrix $\tilde{\mathbf{A}} \in (\mathbb{Q}/x^n - 1)^{d_1 \times d_2}$, so that when taking the quotient by $\Phi_n(x)$, the error is bounded by

$$s_1(\mathbf{A}) \leq \theta p\sqrt{n} O\left(\sqrt{d_1} + \sqrt{d_2} + \omega\sqrt{n}\right). \tag{3.8}$$

Lemma 3.1.16 (Hensel's Lemma)**.** *Let $f(x) \in \mathbb{Z}[x]$ be a monic polynomial and consider $R = \mathbb{Z}[x]/f(x)$. Then for all primes p and for all $u \in R_{p^e}$, if u is invertible when reducing modulo p (as an element of R_p), then it is also invertible in R_{p^e}.*

Lemma 3.1.17. *Let $p, p' \in \mathbb{Z}^+$ be primes such that p is a primitive root modulo p'^2 and let l be a power of p'. Let $R_l = \mathbb{Z}[x]/\langle \Phi_n(x), l \rangle$ and suppose that $\mathbf{A}$ is sampled uniformly at random from $R_l^{k \times k}$, for some dimension $k \in \mathbb{Z}^+$, where l is a power of p'. Then there exists a negligible function $\epsilon(l)$ such that $\mathbf{A}$ is invertible with probability $1 - \epsilon(l)$.*

Proof. A matrix $\mathbf{A} \in R_l^{k \times k}$ is invertible if and only if its determinant $D = \mathrm{Det}(\mathbf{A})$ is invertible in R_l. By Lemma 3.1.16, D is invertible in R_l if it is invertible when reducing modulo p. From our choice of p we have that $\Phi_n(x)$ is irreducible modulo p and R_p is a field of p^n elements. Hence D is invertible in R_l with probability $1 - \frac{1}{1-l}$. □

In Table 3.1 we present the numerical results for the first primes and the cases where this parameters can be applied.

Table 3.1: Smallest primitive root q modulo p^2

Prime p	Primitive root q
3	2
5	2
7	3
11	2
13	2
17	3
19	2
23	5
27	2
31	3

3.2 Regev's Learning With Errors Cryptosystem

Despite the fact that cryptographic constructions based on lattice problems started to appear in the late '90s, they did not receive much attention from the community at that time, since the only known encryption scheme (the Ajtai-Dwok construction [3]) required impractical key-size to be secure [29]. Other constructions such as GGH [17] and NTRU [21] appeared after Ajtai's work, but neither of them enjoyed of a theoretical reduction to the hardness of classical lattice problems. It was not until 2005 and Regev's learning with errors cryptosystem that lattice based cryptography started growing, and since then many other cryptographic constructions have been built around Regev's cryptosystem.

The Learning With Errors Problem

The learning with errors problem is, in some sense, similar to the short integer solution problem. The main difference is that in this case the system is overdetermined and noisy. Suppose that we are given m samples of the form

$$\begin{array}{rcll} a_{1,1}s_1 + a_{1,2}s_2 + \ldots + a_{1,n}s_n & = & b_1 + \epsilon_1 & (\text{mod } q) \\ a_{2,1}s_1 + a_{2,2}s_2 + \ldots + a_{2,n}s_n & = & b_2 + \epsilon_2 & (\text{mod } q) \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1}s_1 + a_{m,2}s_2 + \ldots + a_{m,n}s_n & = & b_m + \epsilon_m & (\text{mod } q) \end{array}$$

where the vector $\mathbf{s} = \{s_1, \ldots, s_n\}$ is unknown, the coefficients $\{a_{i,j}\}_{i\in\{1,\ldots,n\},j\in\{1,\ldots,m\}}$ are uniformly sampled from $\mathbb{Z}_q$ and the error ϵ_i is sampled according a probability distribution ψ. The problem is to find the secret vector $\mathbf{s}$. Notice that with no error the problem can be efficiently solved by using Gaussian elimination; however, when error is added the problem becomes much more challenging.

Definition 3.2.1 (The Learning With Errors Problem ($\text{LWE}_{q,\psi}$))**.** Let $n, q \in \mathbb{Z}^+$ and let ψ be a probability distribution on $\mathbb{Z}_q$. Let $\mathbf{s} \in \mathbb{Z}_q^n$. The *learning with errors problem* ($\text{LWE}_{n,q,\psi}$) is the problem of finding $\mathbf{s}$ given any number of samples of the form

$$\big(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + \epsilon\big)$$

where $\mathbf{a}$ is sampled uniformly from $\mathbb{Z}_q^n$ and ϵ is sampled from $\mathbb{Z}_q$ according to ψ.

The learning with errors problem was introduced in 2005 by Oded Regev [32] as a generalization of the learning parity with noise problem. Regev gave a quantum reduction from LWE to the GapSVP problem. The latter is, to this date, believed to be a hard problem even in the context of quantum algorithms. In addition he also described a cryptosystem whose difficulty is based on LWE. The problem and the cryptosystem have played a central role in the development of lattice-based cryptography. Since its publication, several other constructions whose security is based on LWE have appeared [30][6][16], and better security guarantees have been given.

Theorem 3.2.2. *Let $n, q \in \mathbb{Z}^+$ and let $\theta \in (0, 1)$ a real number such that $\theta q > \sqrt{n}$. If there exists an efficient algorithm that solves $\text{LWE}_{n,q,\psi}$, then there exists an efficient algorithm to solve $\text{GapSVP}_{\tilde{O}(n/\theta)}$ in the worst case* [1].

[1] In [32], Regev also gives a reduction from $\text{LWE}_{n,q,\psi}$ to $\text{SIVP}_{\tilde{O}(n/\theta)}$

3.2.1 Symmetric-Key LWE Cryptosystem

The learning with errors problem is about the difficulty of learning a linear function $L_{\mathbf{a}}\colon \mathbf{a} \mapsto \langle \mathbf{a}, \mathbf{s} \rangle$, given an arbitrary number of samples of approximate evaluations of it. Using this problem as a base, Regev had the idea of hiding the message by transforming the linear function into an affine function, translating the image by a secret message μ, resulting in

$$\mu \mapsto (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \mu + \epsilon).$$

However, if both domain and range are of the same size, the noise would make the decryption impossible, due to overlapping. This issue can be easily fixed by initially taking messages from a smaller set and then "scaling" the message to the bigger set. This escalation process may require to round the result. Then to formalize the idea we give the following definition

Definition 3.2.3 (Randomized Rounding Function)**.** A *randomized rounding function* is a function $\varrho\colon \mathbb{R} \to \mathbb{Z}$ such that for all $x \in \mathbb{R}$ and for all $a \in \mathbb{Z}$

$$\varrho(x + a) = \varrho(x). \tag{3.9}$$

The function $\lfloor \psi \rceil(x) := \varrho(x) - x$ is called the *rounding error of* ϱ.

A randomized rounding function is then completely determined by its value on the interval $[0, 1)$. When this function is restricted to $\mathbb{Z}$ we are adding the same "error" $\psi(0)$. With this definition we can formalize the previous idea of a cryptosystem based on the learning with errors problem.

Definition 3.2.4 (LWE Symmetric Encryption Scheme (symLWE))**.** Let $n, t, q \in \mathbb{Z}^+$ such that $t \geq 2$ and $q = n^{O(1)}$. Let $\varrho : \mathbb{R} \to \mathbb{Z}$ a randomized rounding function. The *Learning With Errors symmetric encryption scheme* is the encryption scheme with message space $\mathbb{Z}_t$, and ciphertext space $\mathbb{Z}_q^{n+1}$, that consists of the following algorithms:

KeyGen: Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random or as a random short vector. The element $\mathbf{s}$ is the shared secret key.

Encryption: Given a message $\mu \in \mathbb{Z}_t$ and the key $\mathbf{s}$, sample $\mathbf{a} \leftarrow_R \mathbb{Z}_q^n$ uniformly at random. An encryption of μ under $\mathbf{s}$ is the element

$$\mathsf{symLWE}_{t:q}^{\psi}(\mathbf{s}, \mu) = \left(\mathbf{a}, \varrho\left(\langle \mathbf{a}, \mathbf{s} \rangle + \frac{\mu q}{t}\right) \pmod q\right). \tag{3.10}$$

Decryption: Given a ciphertext $\mathbf{c} = (\mathbf{a}, b)$ and the key $\mathbf{s}$, the message can be recovered using the following equation

$$\mu' = \left\lfloor \frac{t}{q}(b - \langle \mathbf{a}, \mathbf{s} \rangle) \right\rceil . \tag{3.11}$$

The *error* of a ciphertext $(\mathbf{a}, b)$ (with respect to μ and $\mathbf{s}$) is the function

$$\epsilon_{\mu,\mathbf{s}}(\mathbf{a},\ b) = b - \langle \mathbf{a}, \mathbf{s} \rangle - \mu \frac{t}{p} \pmod{q}, \tag{3.12}$$

taking its value in the set $\left\{-\frac{q}{2}, -\frac{q}{2}+1, \ldots, \frac{q}{2}-1, \frac{q}{2}\right\}$.

In the previous definition, it is not clear that the decryption procedure will recover the original message; in fact, the message computed will not always be equal to the original, because the message can be mapped to any element in $\mathbb{Z}_q^{n+1}$, since we did not impose any bound on the error. To ensure the correctness of the previous cryptosystem we have to control the error added by ϱ. For a message $\mu \in \mathbb{Z}_t$ and an error $e \in \mathbb{Z}^+$ let

$$\mathsf{symLWE}^e_{t:q}(\mu, \mathbf{s}) = \left\{(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1} : |\epsilon_{\mu,\mathbf{s}}(\mathbf{a},\ b)| \leq e\right\}. \tag{3.13}$$

Proposition 3.2.5 (Correctness of symLWE)**.** *Let $e \in \{-\frac{q}{2}, \ldots, \frac{q}{2}\}$ and let $\mathbf{c} \in \mathsf{symLWE}^e_{t:q}(\mu, \mathbf{s})$. If $e < \frac{q}{2t}$ then the decryption algorithm recovers correctly the encrypted message.*

Proof. Let $(\mathbf{a}, b) \in \mathsf{symLWE}^e_{t:q}(\mu, \mathbf{s})$ and let $\epsilon = \epsilon_{\mu,\mathbf{s}}(\mathbf{a}, b)$ be the error in the ciphertext. Then, by Equation 3.11,

$$\begin{aligned} \mu' &= \left\lfloor \tfrac{t}{q}\big(b - \langle \mathbf{a}, \mathbf{s} \rangle\big) \right\rceil \\ &= \left\lfloor \tfrac{t}{q}\big(\tfrac{q}{t}\mu + \epsilon\big) \right\rceil \\ &= \left\lfloor \mu + \tfrac{t}{q}\epsilon \right\rceil \\ &= \mu + \left\lfloor \tfrac{t}{q}\epsilon \right\rceil . \end{aligned}$$

Finally, since $\epsilon \in \{-e, \ldots, e-1, e\}$ and $e < \frac{1}{2}$, we have that $\left\lfloor \frac{t}{q}\epsilon \right\rceil = 0$. Therefore $\mu' = \mu$. □

Homomorphic property of LWE One of the most important properties of the LWE cryptosystem is the fact that it is "homomorphic": the sum of two LWE ciphertexts is an LWE ciphertext of the sum of the messages, under only the condition that the error is not too big. This property is explained on the following proposition and further explored in the following chapters.

Proposition 3.2.6. *Let $t, q, n \in \mathbb{Z}^+$, $\mu, \mu' \in \mathbb{Z}_t$, $\mathbf{s} \in \mathbb{Z}_q^n$ and let $(\mathbf{a}, b) \in \mathsf{LWE}_{t:q}^e(\mathbf{s}, \mu)$ and $(\mathbf{a}', b) \in \mathsf{LWE}_{t:q}^e(\mathbf{s}, \mu)$. If ψ is a subgaussian distribution of parameter θ, then*

$$(\mathbf{a} + \mathbf{a}', b + b') \in \mathsf{LWE}_{t:q}^{2e}(\mathbf{s}, \mu + \mu').$$

is an LWE *encryption of $\mu + \mu'$ of parameter $\sqrt{2}\theta$.*

Proof. The proposition follows immediately from the definition of the LWE encryption scheme and Theorem 3.1.13. □

LWE Modulus-Switch The LWE encryption scheme allows us to change the ciphertext modulus even after the message has been encrypted, just by scaling the coordinates of the ciphertext (no key material is necessary). The process will, of course, modify the noise of the ciphertext as well, which may be desirable in some settings.

Definition 3.2.7 (Randomized Scaling Function)**.**
Let $q, l \in \mathbb{Z}^+$. The *randomized scaling function* is the function $[\cdot]_{l:q} \colon \mathbb{Z}_l \to \mathbb{Z}_q$ defined by

$$[a]_{l:q} = \left\lfloor \frac{aq}{l} \right\rfloor + \varsigma, \tag{3.14}$$

where $\varsigma \in \{0, 1\}$ is such that

$$\Pr(\varsigma = 1) = \frac{aq}{l} - \left\lfloor \frac{aq}{l} \right\rfloor \in [0, 1).$$

Definition 3.2.8 (Modulus-Switching Operation)**.**
Let $t, q, l, n \in \mathbb{Z}^+$ and let $\mu \in \mathbb{Z}_t$. The *modulus-switching operation* is the function $\mathrm{ModSwitch} \colon \mathbb{Z}_l^{n+1} \to \mathbb{Z}_q^{n+1}$ defined by

$$\mathrm{ModSwitch}(a_0, a_1, \ldots, a_n) = \big([a_0]_{l:q}, [a_1]_{l:q}, \ldots, [a_n]_{l:q}\big).$$

Proposition 3.2.9. *Let $t, q, l, n \in \mathbb{Z}^+$, $\mu \in \mathbb{Z}_t$ and $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}_q^n$. If $(\mathbf{a}, b) \in \mathsf{LWE}_{t:l}^{\psi}(\mathbf{s}, \mu)$ is an encryption of μ where ψ is a subgaussian distribution of parameter θ, then $\mathrm{ModSwitch}(\mathbf{a}, b) \in \mathsf{LWE}_{t:q}^{\psi'}(\mathbf{s}, \mu)$ is an encryption of μ where ψ' is a subgaussian distribution of parameter*

$$\sqrt{\left(\frac{q\theta}{l}\right)^2 + 2\pi(|\mathbf{s}|^2 + 1)}.$$

Proof. Note that for all $a \in \mathbb{Z}$,

$$[a]_{l:q} = \frac{q}{l}a + r,$$

where $r < 1$. Hence r can be regarded as a subgaussian distribution of parameter $\sqrt{2\pi}$ (see Proposition 3.1.12).

Let $\mathbf{c} = (\mathbf{a}, b) = (a_0, \ldots, a_{n-1}, b)$ and let $[c]_{l,q} = \left([a_0]_{l,q}, \ldots, [a_{n-1}]_{l,q}, [b]_{l,q}\right) = (\mathbf{a}', b')$. Then

$$\begin{aligned}
\epsilon_{\mu,\mathbf{s}}(\mathbf{a}', b') &= b' - \langle \mathbf{a}', \mathbf{s} \rangle - \tfrac{qm}{t} \\
&= \tfrac{q}{l}b + r_b - \textstyle\sum_{i=0}^{n-1} \left(\tfrac{q}{l}a_i + r_i\right) s_i \\
&= \tfrac{q}{l}(b - \langle \mathbf{a}, \mathbf{s} \rangle) + r_b + \textstyle\sum_{i=0}^{n-1} r_i s_i \\
&= \tfrac{q}{l}\epsilon_{\mu,\mathbf{s}}(\mathbf{a}, b) + \textstyle\sum_{i=0}^{n-1} r_i s_i.
\end{aligned}$$

Therefore, since $\epsilon_{\mu,\mathbf{s}}(\mathbf{a}, b), r_b, r_0, \ldots, r_{n-1}$ are independent variables we have that, by Theorem 3.1.13, the sum is a subgaussian distribution of parameter

$$\sqrt{\frac{q}{l}\theta + 2\pi\left(\sum_{i=0}^{n-1} s_i^2 + 1\right)}.$$

□

LWE Key-Switching

The key-switching property allows a third party to change the key of an LWE ciphertext without learning anything about the message or the encryption keys. This process takes advantage of a certain "key homomorphic" property that LWE possesses, and it is described in the following definitions.

Definition 3.2.10 (Key-Switching Key)**.** Let $t, q, n, m, B_{ks}, d_{ks} \in \mathbb{Z}^+$ with $B_{ks} < q$ and $d_{ks} = \log_{B_{ks}} q$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{z} = (z_0, \ldots, z_{m-1}) \in \mathbb{Z}_l^m$. For $i \in \{1, \ldots, m\}, j \in \{0, \ldots, d_{ks} - 1\}$ and $\xi \in \{0, \ldots, B_{ks}\}$ let

$$\mathfrak{k}_{i,j,\xi} \in \mathsf{LWE}_{q:q}^{\psi}\left(\mathbf{s}, \xi z_i B_{ks}^j\right)$$

where ψ is a subgaussian distribution. A *key-switching key* from $\mathbf{z}$ to $\mathbf{s}$ is a set of the form

$$\mathfrak{K}_{ks} = \left\{\mathfrak{k}_{i,j,\xi} : i \in \{0, \ldots, m-1\}, j \in \{0, \ldots, d_{ks} - 1\}, a \in \{0, \ldots, B_{ks}\}\right\}.$$

Note htat the key-switching key is not unique and, since the message and the ciphertext moduli used in Definition 3.2.10 are equal, the elements of the key are not decryptable ciphertexts. The size of the key-switching key directly depends on the parameters B_{ks} and d^{ks}.

Definition 3.2.11 (Key-Switching Operation)**.** Let $t, q, n, m \in \mathbb{Z}^+$ and let $\mu \in \mathbb{Z}$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{z} \in \mathbb{Z}_q^m$. Let $\mathfrak{K}_{ks}$ be a key-switching key. The *key-switching operation* is the function $KeySwitch\colon \mathbb{Z}_q^{m+1} \to \mathbb{Z}_q^{n+1}$ defined by

$$KeySwitch(a_0, a_1, \ldots, a_m) = (\vec{0}, a_m) - \sum_{i=0}^{m-1} \sum_{j=1}^{d_{ks}-1} \mathfrak{k}_{i,j,a_{i,j}}$$

where for $i \in \{0, \ldots, m-1\}$ and $j \in \{0, \ldots, d_{ks}-1\}$, we define $a_{i,j}$ to satisfy the equation $a_i = \sum_{j=0}^{d_{ks}} B_{ks}^j a_{i,j}$.

Proposition 3.2.12. *Let $t, q, n, m, B_{ks}, d_{ks} \in \mathbb{Z}^+$ with $B_{ks} < q$ and $d_{ks} = \log_{B_{ks}} q$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{z} \in \mathbb{Z}_q^m$. Let $\mathfrak{K}$ be a key-switching key from $\mathbf{z}$ to $\mathbf{s}$ and suppose that its elements were created using a subgaussian distribution of parameter θ_{ks}. If $(\mathbf{a}, b) \in \mathsf{LWE}_{t:q}^{\psi}(\mathbf{z}, \mu)$ for a message $\mu \in \mathbb{Z}_t$ and a subgaussian distribution ψ of parameter θ, then $KeySwitch(\mathbf{a}, b) \in \mathsf{LWE}_{t:q}^{\psi'}(\mathbf{s}, \mu)$, where ψ' is a subgaussian distribution of parameter*

$$\sqrt{\theta^2 + m d_{ks} \theta_{ks}^2}.$$

Proof. Suppose that $\mathbf{z} = (z_0, \ldots, z_{m-1})$. For an element

$$\mathfrak{k}_{i,j,\xi} = \left(\mathbf{a}^{(i,j,\xi)}, b_0^{(i,j,\xi)}\right) = \left(\left(a_0^{(i,j,\xi)}, \ldots, a_{m-1}^{(i,j,\xi)}\right), b^{(i,j,\xi)}\right) \in \mathfrak{K},$$

let $\epsilon_{i,j,\xi} = \epsilon_{\mu,\mathbf{s}}(\mathfrak{k}_{i,j,\xi})$ be the error in the LWE ciphertext. By definition of the key-switching operation, if $\text{KeySwitch}(\mathbf{a}, b) = (\mathbf{a}', b') = (a'_0, \ldots, a'_{n-1}, b)$, then we have that

$$\mathbf{a}' = -\sum_{i=0}^{m-1} \sum_{j=0}^{d_{ks}-1} \mathbf{a}^{(i,j,a_{i,j})}.$$

On the other hand we also have that

$$\begin{aligned}
b' &= b - \textstyle\sum_{i=0}^{m-1} \sum_{j=0}^{d_{ks}-1} b^{(i,j,a_{i,j})}. \\
&= b - \textstyle\sum_{i=0}^{m-1} \sum_{j=0}^{d_{ks}-1} \left\langle \mathbf{a}^{(i,j,a_{i,j})}, \mathbf{s} \right\rangle + a_{i,j} z_i B_{ks}^j + \epsilon_{i,j,a_{i,j}} \\
&= b + \left\langle - \textstyle\sum_{i=0}^{m-1} \sum_{j=0}^{d_{ks}-1} \mathbf{a}^{(i,j,a_{i,j})}, \mathbf{s} \right\rangle - \left(\textstyle\sum_{i=0}^{m-1} z_i \sum_{j=0}^{d_{ks}-1} a_{i,j} B_{ks}^j \right) + \epsilon \\
&= b + \langle \mathbf{a}', \mathbf{s} \rangle - \left(\textstyle\sum_{i=0}^{m-1} z_i a_i \right) + \epsilon \\
&= b + \langle \mathbf{a}', \mathbf{s} \rangle - \langle \mathbf{z}, \mathbf{a} \rangle + \epsilon,
\end{aligned}$$

where

$$\epsilon = \sum_{i=0}^{m-1} \sum_{j=0}^{d_{ks}-1} \epsilon_{i,j,a_{i,j}}$$

is, by Theorem 3.1.13, a subgaussian distribution of parameter $\sqrt{md_{ks}}\theta_{ks}$. Finally, to compute the total error distribution, note that

$$\begin{aligned} \epsilon_{\mu,\mathbf{s}}(\mathbf{a}', b') &= b' - \langle \mathbf{a}', \mathbf{s} \rangle - \tfrac{q\mu}{t} \\ &= b + \langle \mathbf{a}', \mathbf{s} \rangle - \langle \mathbf{a}, \mathbf{z} \rangle + \epsilon - \langle \mathbf{a}', \mathbf{s} \rangle - \tfrac{q\mu}{t} \\ &= b - \langle \mathbf{a}, \mathbf{z} \rangle - \tfrac{q\mu}{t} + \epsilon \\ &= \epsilon_{\mu,\mathbf{z}}(\mathbf{a}, b) + \epsilon, \end{aligned}$$

which is a subgaussian distribution of parameter $\sqrt{\theta^2 + md_{ks}\theta_{ks}^2}$. □

3.2.2 Public-Key LWE Cryptosystem

In the last subsection we explained the construction of a symmetric-key cryptosystem whose security is based on the learning with errors problem. It is possible as well to construct a public-key cryptosystem based on the same problem. We proved that the symmetric-key cryptosystem has certain homomorphic properties that, based on a result that is explained further in Section 4.1, are enough to create a public-key cryptosystem. However, we give the construction explicitly here.

Definition 3.2.13 (LWE Public-Key Encryption Scheme)**.** Let $n, m, t, q \in \mathbb{Z}^+$ such that $t \geq 2$ and $q = n^{O(1)}$. Let $\varrho\colon \mathbb{R} \to \mathbb{Z}$ a randomized rounding function. The *Learning With Errors public-key encryption scheme* is the encryption scheme with message space $\mathbb{Z}_t$, and ciphertext space $\mathbb{Z}_q^{n+1}$, that consists of the following algorithms:

KeyGen: Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random or as a random short vector. The secret key is

$$\mathsf{sk} = \mathbf{s}. \tag{3.15}$$

Sample $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ uniformly at random and sample $\mathbf{E} \in \mathbb{Z}_q^m$ choosing each entry according to the distribution ψ_θ. The public key is

$$\mathsf{pk} = \left(\mathbf{A}, \mathbf{P} = \mathbf{A}\mathbf{s}^T + \mathbf{E}\right) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m. \tag{3.16}$$

Encryption: Given a message $\mu \in \mathbb{Z}_t$ and the public key $(\mathbf{A}, \mathbf{P})$, sample $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random. An encryption of μ under $\mathbf{s}$ is the element

$$\left(\mathbf{u} = \mathbf{A}\mathbf{a}^T, b = \varrho\left(\langle \mathbf{P}, \mathbf{a}\rangle + \frac{\mu q}{t} \pmod q\right)\right). \tag{3.17}$$

Decryption: Given a ciphertext $\mathbf{c} = (\mathbf{a}, b)$ and the private key $\mathbf{s}$, the message can be recovered using the following equation

$$\mu' = \left\lfloor \frac{p}{q}\big(b - \langle \mathbf{a}, \mathbf{s}\rangle\big) \right\rceil. \tag{3.18}$$

The *error* of a ciphertext $(\mathbf{a}, b)$ (with respect to μ and $\mathbf{s}$) is the function

$$\epsilon_{\mu,\mathbf{s}}(\mathbf{a},\ b) = b - \langle \mathbf{a}, \mathbf{s}\rangle - \mu\frac{q}{t} \pmod q, \tag{3.19}$$

taking its value in the set $\left\{-\frac{q}{2}, -\frac{q}{2}+1, \ldots, \frac{q}{2}-1, \frac{q}{2}\right\}$.

Note that the public key is just a collection of encryptions of 0, in terms of the symmetric-key cryptosystem that we previously defined.

3.3 Learning With Errors Over Rings

The public-key cryptosystem based on the learning with errors problem involves very simple operations in the encryption and decryption procedures, only sums and multiplications modulo q. This is an advantage when compared to classical cryptosystems such as RSA or other constructions based on elliptic curves. However, the key-size is a great disadvantage, making it very difficult to deploy in practice. The same problem is found in Ajtai's construction of a one-way function (see Section 2.3). One possible solution is to use circulant matrices instead of totally random matrices. A *circulant matrix* is a matrix of the form

$$\mathbf{A} = \begin{pmatrix} a_0 & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_1 & a_0 & a_{n-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix}.$$

Notice that all the information necessary to represent a circulant matrix is encoded on the first column, reducing the key size to $n \log q$. It also reduces (asymptotically) the running time required to compute matrix products, using the fast Fourier transform (see Section 7.1). However, the adaptation of Ajtai's function to this family of matrices [26] was proven to be weak against a very simple cryptographic attack that finds collisions [25, 28].

It is possible to establish a relation between these matrices and quotients of the polynomial ring $\mathbb{Z}[x]$. Given a vector $\mathbf{v} = (v_0, \ldots, v_{n-1}) \in \mathbb{Z}_q^n$, multiplying on the left by the circulant matrix $\mathbf{A}$ results in

$$\mathbf{A}\mathbf{v}^T = \begin{pmatrix} a_0 v_0 + a_{n-1} v_1 + a_{n-2} v_2 + \ldots + a_1 v_{n-1} \\ a_1 v_0 + a_0 v_1 + a_{n-1} v_2 + \ldots + a_2 v_{n-1} \\ \vdots \\ a_{n-1} v_0 + a_{n-2} v_1 + a_{n-3} v_2 + \ldots + a_0 v_{n-1} \end{pmatrix}.$$

which is the vector of coefficients of the product $(v_0 + v_1 x + \ldots + v_{n-1} x^{n-1})(a_0 + a_1 x + \ldots + a_{n-1} x^{n-1})$ as elements of the ring $\mathbb{Z}[x]/\langle x^n - 1, q \rangle$. Moreover, given any polynomial $f(x) = f_0 + f_1 x + \ldots + f_n x^n + x^{n-1} \in \mathbb{Z}[x]$, the ring $R_q = \mathbb{Z}[x]/\langle f(x), q \rangle$ has a q-ary lattice structure given by the matrix

$$\left(\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & -f_0 \\ 1 & 0 & \cdots & 0 & -f_1 \\ 0 & 1 & \cdots & 0 & -f_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -f_n \end{array}\right) \in \mathbb{Z}_q^{n+1}.$$

This structure is known as an *ideal lattice*.

The *ring learning with errors problem* is the restriction of the learning with errors problem to the class of ideal lattices (instead of totally random lattices). The use of this variant can be traced back to NTRU [21] and no significantly better attacks have been found to ideal lattices when compared to their totally random counterparts.

Chapter 4

Fully-Homomorphic Encryption

When we think about encrypted messages we can imagine that the messages are covered under a cloak that makes the message, and even its "shape", unrecognizable (which is what we formally call semantic security). Therefore, being able to successfully manipulate encrypted messages without compromising the semantic security of the cryptosystem may seem paradoxical. However, many widely deployed cryptosystems allow to compute certain operations on ciphertexts, such as RSA [33], Goldwasser-Micali [18], ElGamal [12] and LWE [32], although none of these cryptosystems can effectively evaluate arbitrary (computational) functions. The ones that are able to do so are formally described by the following definition.

Definition 4.0.1 (Fully-Homomorphic Encryption)**.** Let $E = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an (either public-key or symmetric-key) encryption scheme. We say that E is *fully-homomorphic* if there is a computationally efficient function Eval and a polynomial f such that, for all parameters λ, for all circuits $\mathcal{C}$ of arity t and depth at most $f(\lambda)$, and for all messages $\mu_1, \ldots, \mu_t$, if $c_1, \ldots, c_t$ are their corresponding ciphertexts, then

$$\mathsf{Eval}(\mathsf{ek}, c_1, \ldots, c_t) \tag{4.1}$$

is a valid encryption of $\mathcal{C}(\mu_1, \ldots, \mu_t)$, where ek is an evaluation key possibly containing (encrypted) information about the encryption key.

The existence of a fully-homomorphic encryption scheme would have great theoretical and practical impact: in theory, many other constructions would become possible by using a fully-homomorphic encryption scheme, such as functional encryption, obfuscation, homomorphic signatures, etc. In practice, encrypted data could be sent to an untrusted

party with high computational power (such as the cloud) and computations on the data could be outsourced to the third party without revealing any information about the data or any intermediate result. However, for over 30 years it remained a longstanding open problem to find a construction satisfying these properties, until in 2009, when Craig Gentry [15] published the first construction of a fully-homomorphic cryptosystem based on ideal lattices (ideal lattices are described in Chapter 3).

In this chapter we give an introductory overview of fully-homomorphic encryption, where we abstractly describe the bootstrapping procedure, which is the main idea behind Gentry's construction. Moreover, for all existing constructions of fully-homomorphic cryptosystems, a bootstrapping step is needed and it is the main bottleneck for their performance.

4.1 Homomorphic Encryption

In algebra, a homomorphism between two algebraic structures G and H is a function $f : G \longrightarrow H$ that "preserves their corresponding structure". For example if G and H are groups, then a homomorphism is a function that preserves the group operation: for $g, g' \in G$, computing gg' (as an element of G) and then computing $f(gg')$ is equivalent to computing $f(g)$ and $f(g')$, and then $f(g)f(g')$ (as elements of H).

The word "homomorphism" has a different meaning depending on the context in which it is used. A *group homomorphism* from G to H is a function $f : G \longrightarrow H$ with the property described above (for all $g, g' \in G$, $f(gg') = f(g)f(g')$). There are several examples of cryptosystems for which the encryption function is a group homomorphism. For example, the RSA encryption function is a homomorphism from $\mathbb{Z}_n^*$ onto itself since, for all $\mu, \mu' \in \mathbb{Z}_n^*$

$$\mathsf{Enc}(\mu\mu') = (\mu\mu')^e = \mu^e \mu'^e = \mathsf{Enc}(\mu)\mathsf{Enc}(\mu').$$

Nevertheless, the message and ciphertext spaces can have more complicated structures. For instance consider the following cryptosystem

KeyGen: Choose $n \in 2\mathbb{Z} + 1$ a large random odd integer. For $i \in \{1, \ldots, m\}$ let $\mathfrak{k}_i = a_i n + 2r_i$, where $a_i, r_i \in \mathbb{Z}$ are random integers such that $|r_i| < \frac{n}{2l}$. The secret key is $\mathsf{sk} = n$, and the public key is $\mathsf{pk} = \{\mathfrak{k}_i\}_{i \in \{1,\ldots,m\}}$.

Encryption: Given a message $\mu \in \{0, 1\}$, choose a random bit string $s_1 \ldots s_m$ and output

$$\begin{aligned} c &= \sum_{i=1}^m s_i \mathfrak{k}_i + \mu \\ &= n \sum_{i=1}^m s_i a_i + 2 \sum_{i=1}^m s_i r_i + \mu. \end{aligned}$$

Decryption: Given a ciphertext $c \in \mathbb{Z}$, compute

$$\begin{aligned} \tilde{c} &= c \pmod{n} \\ &= 2r + \mu, \end{aligned}$$

where $r = \sum_{i=1}^{m} r_i$. Then μ can be recovered by computing $\text{LSB}(\tilde{c})$.

The correctness of the scheme follows from the fact that $|r_i| < \frac{n}{2l}$. It is also easy to see that, given some extra conditions on the size of r_i, the encryption function is a ring homomorphism. Suppose that $|r_i| < \frac{n}{4l}$, let $\mu, \mu' \in \mathbb{Z}_2$, and let $c, c' \in \mathbb{Z}$ be their corresponding encryptions. Then we have that

$$\begin{aligned} c + c' &= \Big(\textstyle\sum_{i=1}^{m} s_i \mathfrak{k}_i + \mu\Big) + \Big(\sum_{i=1}^{m} s'_i \mathfrak{k}_i + \mu'\Big) \\ &= n\textstyle\sum_{i=1}^{m} \big(s_i + s'_i\big)a_i + 2\sum_{i=1}^{m} \big(s_i + s'_i\big)r_i + \mu + \mu_i, \end{aligned}$$

which is a valid encryption of $\mu + \mu'$ since $2\sum_{i=1}^{m}(s_i + s'_i)r_i < n$. Multiplication works in a similar manner, but the required bound on $|r_i|$ is much more restrictive.

$$\begin{aligned} c \cdot c' &= \Big(\textstyle\sum_{i=1}^{m} s_i \mathfrak{k}_i + \mu\Big) \cdot \Big(\sum_{i=1}^{m} s'_i \mathfrak{k}_i + \mu'\Big) \\ &= \Big(n\textstyle\sum_{i=1}^{m} s_i a_i + 2\sum_{i=1}^{m} s_i r_i + \mu\Big) \cdot \Big(n\sum_{i=1}^{m} s'_i a_i + 2\sum_{i=1}^{m} s'_i r_i + \mu'.\Big) \\ &= sn + 4\big(\textstyle\sum_{i=1}^{m} s_i r_i\big)\big(\sum_{i=1}^{m} s'_i r_i\big) + 2\mu\sum_{i=1}^{m} s'_i r_i + 2\mu'\sum_{i=1}^{m} s_i r_i + \mu\mu' \\ &= sn + 4\textstyle\sum_{i=1}^{m}\sum_{j=1}^{m} r_i s_i r_j s'_j + 2\sum(\mu' r_i s_i + \mu r_i s'_i) + \mu\mu'. \end{aligned}$$

Therefore, it is necessary that $|r_i| < k\sqrt{n}$ for some constant $k < 1$.

Even if the protocol is designed with these bounds on the "noise", this noise grows too fast with every multiplication, making the homomorphic operations unsustainable (in constrast with RSA, that can homomorphically evaluate any number of multiplications). A cryptosystem like the one described above is called *somewhat homomorphic* since, even though it can successfully evaluate both ring operations, the number of operations is bounded by a constant (that, in this case, depends on the parameters of the cryptosystem). The *depth capacity* of a homomorphic encryption scheme $\mathcal{E}$ is the largest integer Δ such that $\mathcal{E}$ can evaluate all circuits of size less than or equal to Δ.

Symmetric-Key vs Public Key In the context of homomorphic encryption there is no theoretical difference between symmetric-key and public-key encryption. In 2011,

Rothblum [35] proved that a homomorphic cryptosystem with big enough depth capacity can be easily transformed into a public-key encryption scheme by a generic operation that uses the fact that the cryptosystem can evaluate sums and products. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a symmetric-key homomorphic encryption scheme and suppose that the message space is $\{0,1\}$. To get a public-key encryption scheme, the private key σ and the decryption algorithm Dec stay the same.

Public Key: Sample k random bits $b_1, \ldots, b_k$ and compute

$$c_i \leftarrow \mathsf{Enc}(\sigma, b_i).$$

The public key is the list

$$\mathsf{pk} = \big((b_1, c_1), \ldots, (b_k, c_k)\big).$$

Encryption: Given a message $\mu \in \{0,1\}$, sample a bit string $\rho = \rho_1 \ldots \rho_k$ such that $\sum_{i=1}^{k} \rho_i b_i = \mu$. Output the ciphetext

$$\begin{aligned} c &= \textstyle\sum_{i=1}^{k} \rho_i c_i \\ &= \textstyle\sum_{i=1}^{k} \rho_i \mathsf{Enc}(\sigma, b_i) \\ &= \mathsf{Enc}\left(\sigma, \textstyle\sum_{i=1}^{k} \rho_i b_i\right) \\ &= \mathsf{Enc}(\sigma, \mu). \end{aligned}$$

The cryptosystems that we present in the rest of the thesis are all symmetric-key encryption schemes, since they are generally easier to describe. However, all of them can be easily transformed into public-key schemes.

Universal sets of gates In this section we have been describing the intuition of what a homomorphic cryptosystem is: one that can effectively evaluate a family of circuits (for instance, the product of elements in $\mathbb{Z}_n^*$). We also defined a fully-homomorphic cryptosystem as one that can effectively evaluate any arbitrary circuit. Nevertheless, we have been treating them as schemes that can effectively evaluate any number of additions and multiplications. The formal definition suggests that, in order to prove that a cryptosystem is fully-homomorphic, we have to prove that we can evaluate any list of gates, and for that we would have to classify all possible gates. However, there is an easier way to solve this problem by describing a set of atomic elements. For that we first introduce the following definition.

Definition 4.1.1 (Universal Set of Gates)**.** A set of boolean gates $\mathcal{S}$ is called *universal* if for any truth table T there exists a circuit $\mathcal{C}$ with elements in $\mathcal{S}$ such that $\mathcal{C}$ has T as truth table.

Since every computational function is a binary function, we have that all computational functions can be computed by a circuit with elements in a universal set. We may think that finding a universal set of gates is hard; however, there are several examples of such sets, some of which are very simple.

Definition 4.1.2 (NAND and NOR Gates)**.** Let $\mu_1, \mu_2 \in \{0, 1\}$. The NAND function $\mu_1 \barwedge \mu_2$ of μ_1 and μ_2 is the function defined by

$$\text{NAND}(\mu_1, \mu_2) = \mu_1 \barwedge \mu_2 = 1 - \mu_1\mu_2. \tag{4.2}$$

The NOR function $\mu_1 \veebar \mu_2$ of μ_1 and μ_2 is the function defined by

$$\text{NOR}(\mu_1, \mu_2) = \mu_1 \veebar \mu_2 = 1 - \mu_1 - \mu_2 + \mu_1\mu_2. \tag{4.3}$$

Theorem 4.1.3. *The sets* {NAND} *and* {NOR} *are universal.*

Using the definition that we give of NAND and NOR, it is immediate to see that those gates can be effectively computed using addition and multiplication gates. Then by the previous theorem we have that it is equivalent to describe fully-homomorphic encryption using additions and multiplications or using any other universal set of gates.

4.2 Gentry's Bootstrapping Procedure

Several constructions exist of cryptosystems that can homomorphically evaluate both operations in a ring. However, all of them seem to have the same issue: the "noise" growth limits the number of operations that can be performed. One possibility is to try to reduce the noise every time it grows too much. This operation could be performed by decrypting the message, but that would reveal (the current state of) the message. Gentry introduced a technique to turn a somewhat homomorphic cryptosystem into a fully-homomorphic system. The idea is to reduce the noise of the ciphertext by evaluating the decryption circuit, but taking advantages of the (somewhat) homomorphic properties of the cryptosystem. This technique is best known in the literature as bootstrapping. To describe it, we introduce the following definition.

Definition 4.2.1 (Bootstrappable Encryption)**.** Let $\mathcal{E}(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a somewhat homomorphic encryption scheme and let $\mathcal{D}$ be a minimal circuit that computes the decryption function. We say that $\mathcal{E}$ is *bootstrappable* if its depth capacity is strictly greater than the depth of $\mathcal{D}$.

Definition 4.2.2 (The Bootstrapping Procedure)**.** Let $\mathcal{E}(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a bootstrappable encryption scheme and let $\mathcal{D}$ be the decryption circuit corresponding to Dec. Let σ_0, σ_1 be two secret keys and let $\sigma_{bs} = \mathsf{Enc}(\sigma_1, \sigma_0)$ be an encryption of σ_0 under σ_1. The *bootstrapping procedure* takes as input a ciphertext c_0 encrypting a message μ under the key σ_0 and the *bootstrapping key* σ_{bs}, and computes c_1 using Algorithm 2.

Algorithm 2: The bootstrapping procedure

Input : Bootstrapping key σ_{bs}, ciphertext c_0
Output: ciphertext c_1
$c' \leftarrow \mathsf{Enc}(\sigma_1, c_0)$;
$c_1 \leftarrow \mathsf{Eval}(\mathcal{D}, \sigma_{bs}, c')$;
return c_1;

Note that the output of the Algorithm 2 is a ciphertext c_1 given by

$$\begin{aligned} c_1 \leftarrow \mathsf{Eval}(\mathcal{D}, \sigma_{bs}, c') &= \mathsf{Eval}\big(\mathcal{D}, \mathsf{Enc}(\sigma_1, \sigma_0), \mathsf{Enc}(\sigma_1, c_0)\big) \\ &= \mathsf{Enc}\big(\sigma_1, \mathsf{Dec}(\sigma_0, c_0)\big) \\ &= \mathsf{Enc}(\sigma_1, \mu). \end{aligned}$$

Note that the algorithm Eval increases the noise of $\mathsf{Enc}(\sigma_1, c_0)$, however, since the depth of $\mathcal{D}$ is less than the capacity of the cryptosystem, the noise added by Eval is smaller than the noise of c_0. Under these conditions we can say that the bootstrapping procedure is correct (assuming that c_0 has the maximum decryptable noise).

Given a bootstrappable encryption scheme, there is still not an obvious way to turn it into a fully-homomorphic scheme. One possibility is to store a list of keys and publish a list of bootstrapping keys, but in order to evaluate a circuit it is necessary to know its depth beforehand to have the number of keys necessary to reach the required depth capacity. Such a scheme is called *leveled homomorphic*.

However, Gentry's solution just requires a finite set of keys. The idea is to "recycle the keys". More formally, given a sequence of encryption keys

$$(\sigma_1, \sigma_2, \ldots, \sigma_k),$$

publish the sequence of bootstrapping keys

$$(\sigma_{bs}^{(1,2)}, \ldots, \sigma_{bs}^{(k-1,k)}, \sigma_{bs}^{(k,1)}),$$

where $\sigma_{bs}^{(i,j)} = \mathsf{Enc}(\sigma_j, \sigma_i)$. This gives the scheme the capacity to evaluate circuits of arbitrary depth, but requires the assumption that giving away this collection of bootstrapping keys does not convey any information about the keys. Such an assumption is fairly non-trivial, since the list of bootstrapping keys is a circular encryption of the keys that are used for encryption. This assumption is called the *circular assumption*.

Theorem 4.2.3. *Under the circular assumption, if there exists a bootstrappable encryption scheme that can effectively evaluate a universal set of gates, then there exists a fully-homomorphic encryption scheme.*

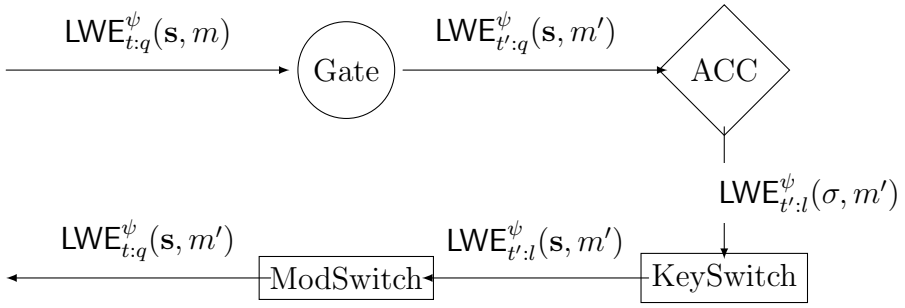
Chapter 5

Ducas-Micciancio LWE-Based Fully-Homomorphic Cryptosystem

Since the first construction of a fully-homomorphic encryption scheme in 2009, several other schemes have been invented with the purpose of improving security and, primarily, efficiency. The main bottleneck in the design of all current fully-homomorphic encryption schemes is the bootstrapping step, since it requires the homomorphic evaluation of the decryption circuit every time the ciphertext has too much noise (which can potentially be every time a single operation is performed). Many constructions were mainly centered around making the depth capacity of the cryptosystem as large as possible, to reduce the amortized cost of the bootstrapping per operation. This approach, in general, requires a robust cryptosystem, which very likely has a complex decryption algorithm. In 2015, Ducas and Micciancio [11] approached the problem in a different way. Their idea was to use a lightweight encryption scheme with a small decryption circuit. This scheme would probably not be able to perform the bootstrapping on its own (because the depth capacity might be too shallow), so a different and more robust encryption scheme is used to evaluate the decryption circuit, to then be collapsed to an encryption of the message under the original encryption scheme. This lightweight bootstrapping is performed after every "non-free" operation, offering the highest possible granularity.

In this chapter we describe in detail the Ducas-Micciancio encryption scheme. We start by giving the necessary mathematical background. Many of the statements and results in Section 3.1 are given in a generalized way, so as to use them in the following chapters. In Section 5.2 we describe a bootstrapping procedure for LWE called the homomorphic accumulator. We dedicate the rest of the chapter to describing in detail the construction. It is helpful to see Figure 5.1 for a graphic description of the idea.

Figure 5.1: High level description of the Ducas-Micciancio cryptosystem



5.1 Computing the NAND Gate

By Theorem 4.2.3 it is enough for a bootstrappable encryption scheme to be able to effectively evaluate a universal set of gates (such as {NAND} and {NOR}) in order to turn it into a fully-homomorphic scheme. For expository reasons, Ducas and Micciancio centered the construction given in [11] around evaluating the NAND gate. However, as we explain in Section 5.5, their idea can be easily adapted to efficiently evaluate other sets of gates.

The idea starts with the following observation: given two bits $\mu_1, \mu_2 \in \{0, 1\}$, it is possible to recover complete information about $\mu_1 \barwedge \mu_2$ from $\mu_1 + \mu_2$ when they are regarded as elements of a bigger group, say $\mathbb{Z}_4$ instead of $\mathbb{Z}_2$ since

$$\begin{aligned} \mu_1 \barwedge \mu_2 = 1 &\quad \Leftrightarrow \quad \mu_1 + \mu_2 \in \{0, 1\} \\ \mu_1 \barwedge \mu_2 = 0 &\quad \Leftrightarrow \quad \mu_1 + \mu_2 \in \{2\}. \end{aligned}$$

Nonetheless, the left-hand side is still a boolean value (an element of $\mathbb{Z}_2$), while now the right-hand side is an element of a bigger set.

The next step consists of modifying the definition of the bootstrapping procedure described in 4.2.2. If the message bits are regarded as elements of $\mathbb{Z}_4$ then, by Proposition 3.2.6, the sum of the LWE ciphertexts is an encryption of the sum of the messages (provided that certain bounds on the noise are satisfied). Then, after a small transformation, a natural modulus change allows us to interpret the information as the result of evaluating the NAND gate on the given messages.

The formal statement and the proof are given by the following proposition.

Proposition 5.1.1. *[11, Lemma 7] Let $r \geq 4$ be a positive integer and let $q = 2^r$. Let $n \in \mathbb{Z}^+$ and let $\mathbf{s} \in \mathbb{Z}_q^n$. For $i \in \{0, 1\}$ let $m_i \in \{0, 1\}$ and $\mathbf{c}_i = (\mathbf{a}_i, b_i) \in \mathsf{LWE}^{\frac{q}{16}}_{4:q}(\mathbf{s}, m_i)$.*

Then the pair

$$\mathsf{homNAND}(\mathbf{c}_0, \mathbf{c}_1) := \left(-\mathbf{a}_0 - \mathbf{a}_1, \frac{5q}{8} - b_0 - b_1\right) \tag{5.1}$$

is an element of $\mathsf{LWE}^{\frac{q}{4}}_{2:q}(\mathbf{s}, m_1 \barwedge m_2)$.

Proof. Let $\mathbf{a} = -\mathbf{a}_0 - \mathbf{a}_1$ and $b = \frac{5q}{8} - b_0 - b_1$. Let $e_0 = \epsilon_{\mu,\mathbf{s}}(\mathbf{c}_0)$ and $e_1 = \epsilon_{\mu,\mathbf{s}}(\mathbf{c}_1)$. Recall that $m_0 \barwedge m_1 = 1 - m_0 m_1$. Then it is enough to prove that the corresponding error is bounded by $\frac{q}{4}$. We have that

$$(m_0 - m_1)^2 = m_0^2 - 2m_0 m_1 - m_1^2.$$

Then, since $m_0, m_1 \in \{0, 1\}$,

$$\begin{aligned} m_0 m_1 &= -\tfrac{(m_0-m_1)^2 - m_0^2 - m_1^2}{2} \\ &= -\tfrac{(m_0-m_1)^2 - m_0 - m_1}{2}. \end{aligned}$$

Note that the previous calculations hold over the rationals. Using this we have that

$$\begin{aligned} b - \langle \mathbf{a}, \mathbf{s}\rangle - (m_0 \barwedge m_1) &= b - \langle \mathbf{a}, \mathbf{s}\rangle - (1 - m_0 m_1) \\ &= \left(\tfrac{5q}{8} - b_0 - b_1\right) - \langle -\mathbf{a}_0 - \mathbf{a}_1, \mathbf{s}\rangle - \tfrac{q}{2}(1 - m_0 m_1) \\ &= \tfrac{q}{2}\left(\tfrac{5}{4} - 1 + m_0 m_1\right) - (\langle \mathbf{a}_0, \mathbf{s}\rangle - b_0) - (\langle \mathbf{a}_1, \mathbf{s}\rangle - b_1) \\ &= \tfrac{q}{2}\left(\tfrac{1}{4} - \tfrac{(m_0-m_1)^2}{2}\right) - (\langle \mathbf{a}_0, \mathbf{s}\rangle - b_0 - \tfrac{m_0 q}{4}) - (\langle \mathbf{a}_1, \mathbf{s}\rangle - b_1 - \tfrac{m_1 q}{4}) \\ &= \tfrac{q}{2}\left(\tfrac{1}{4} - \tfrac{(m_0-m_1)^2}{2}\right) - e_0 - e_1 \\ &= (-1)^{m_0+m_1}\tfrac{q}{8} - (e_0 + e_1). \end{aligned}$$

Therefore $\mathsf{homNAND}(\mathbf{c}_0, \mathbf{c}_1) = (\mathbf{a}, b)$ is a valid ciphertext for $\mathsf{LWE}^{\psi}_{2:q}\frac{q}{4}(\mathbf{s}, m_0 \barwedge m_1)$. □

5.2 Homomorphic Accumulator as a Bootstrapping Technique for LWE

The decryption algorithm for LWE, as described in Definition 3.2.4, is computationally very simple. To decrypt a ciphertext $(\mathbf{a}, b)$ we need to compute the inner product between $\mathbf{a}$ and the secret key $\mathbf{s}$ (which involves n multiplications and $n - 1$ additions modulo q). Then we subtract the result from b, and then we perform a (rounding) scale back to the

message modulus. However, the homomorphic properties of LWE itself are not enough to evaluate its own decryption circuit. To do so, Ducas and Micciancio used an alternate procedure to the original bootstrapping described by Gentry [15]. Instead of using the cryptosystem to evaluate its own decryption circuit, they use a different and more robust encryption scheme, and the resulting ciphertext is then collapsed back to LWE.

The following definition outlines this procedure in an abstract setting. Naturally, a homomorphic accumulator requires key material. To have some control over how much material must be stored, the accumulator is parameterized by a base $B_r < q$ in which elements $a \in \mathbb{Z}_q$ are expressed, and $d_r = \lceil \log_{B_r} q \rceil$. Under these parameters, the encryption key $\mathbf{s} = (s_1, \ldots, s_n)$ is hidden in the *refreshing key* as follows

$$\mathfrak{K}_r = \Big\{ \mathcal{E}\big(u s_i B_r^j \pmod q\big) : u \in \{0, \ldots, B_r - 1\}, j \in \{0, \ldots, d_r - 1\}, i \in \{1, \ldots, n\} \Big\}. \quad (5.2)$$

The construction given in [11] is described in the next section.

Definition 5.2.1 (Homomorphic Accumulator)**.** Let $n, t, q, B_r, d_r \in \mathbb{Z}^+$ with $d_r = \lceil \log_{B_r} q \rceil$. Let $\mathcal{F}$ be a family of functions evaluated over $\mathbb{Z}_q$. A *homomorphic accumulator scheme* is a tuple of algorithms

$$\mathcal{H}om\mathcal{A}cc = (\mathcal{E}, \mathsf{init}, \mathsf{incr}, \mathsf{collapse})$$

such that

$\mathcal{E}$**:** is a homomorphic encryption scheme with $\mathbb{Z}_q$ as message space.

init**:** takes as input an element $a_0 \in \mathbb{Z}_q$ and initializes the variable Acc.

incr**:** takes as input the current value of Acc and an $\mathcal{E}$-encryption of some element $a' \in \mathbb{Z}_q$ and outputs an updated value of Acc.

collapse**:** takes as input the current value of Acc and a function $f \in \mathcal{F}$, and outputs an element $\mathbf{c} \in \mathbb{Z}_q^{n+1}$.

Correctness: Let $e : \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+$ be an error function. A homomorphic accumulator is *e-correct* if for all $(a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ and for all functions $f \in \mathcal{F}$ we have that the *refresh*

procedure described below, outputs a vector $\mathbf{c} \in \mathsf{LWE}_{t:q}^{e(n)}\big(\mathbf{s}, f(v)\big)$, where $v = a_0 + \sum_{i=1}^{n} a_i$.

Algorithm 3: The refresh procedure

```
function Refresh(a_0, {E.Enc(a_i) : 1 ≤ i ≤ n})
    Acc ← init(a_0);
    for i ∈ {1, ..., n} do
        Compute a_{i,1}, ..., a_{i,d_r} such that −a_i = Σ_{j=0}^{d_r} a_{i,j} B_r^j ;
        for j ∈ {1, ..., d_r} do
            Acc ← incr(Acc, E(a_i));
    c ← collapse(Acc);
    return c;
```

Note that the definition of homomorphic accumulator that we give here generalizes the one given by Ducas and Micciancio [11]. In that paper they fixed fix $\mathcal{F}$ to be the set containing only the function $\mathrm{MSB} : \mathbb{Z}_q \to \{0,1\}$, the most significant bit function. In this case we call msbExtract the collapsing function that only takes that function as input. This functionality is enough for them to prove the following result.

Theorem 5.2.2. *[11, Theorem 1] Suppose that $\mathcal{HomAcc} = (\mathcal{E}, \mathsf{init}, \mathsf{incr}, \mathsf{msbExtract})$ is a correct homomorphic accumulator scheme. Let $(\mathbf{a}, b) \in \mathsf{LWE}_{2:q}^{\frac{q}{4}}(\mathbf{s}, \mu)$ and let $\mathfrak{K}_r = \big\{\mathcal{E}(a_{i,j}) : i \in \{1, \ldots, n\}\big\}$. Then the* msbExtract *procedure outputs an* LWE *encryption of* $\mathrm{MSB}(v)$.

Proof. From the definition of correctness, it is enough to prove that the refresh procedure outputs an encryption of $\mathrm{MSB}(v)$. Suppose that e is the error of $(\mathbf{a}, b)$. Then $b - \langle \mathbf{s}, \mathbf{a} \rangle = \frac{q}{2}\mu + e$. Therefore

$$\begin{aligned} v &= b + \tfrac{q}{4} + \textstyle\sum_{i=1}^{n} \sum_{j=0}^{d_r - 1} a_{i,j} s_i B_r^j \\ &= b + \tfrac{q}{4} + \textstyle\sum_{i=1}^{n} s_i \sum_{j=0}^{d_r - 1} a_{i,j} B_r^j \\ &= b + \tfrac{q}{4} - \textstyle\sum_{i=1}^{n} s_i a_i \\ &= b - \langle \mathbf{s}, \mathbf{a} \rangle \\ &= \tfrac{q}{2}\mu + e + \tfrac{q}{4}. \end{aligned}$$

Since $|e| \leq \frac{q}{4}$ we have that $0 \leq e + \frac{q}{4} < \frac{q}{2}$. Hence v is such that

$$v \in \begin{cases} \left(0, \frac{q}{2}\right) & \text{if } \mu = 0 \\ \left(\frac{q}{2}, q\right) & \text{if } \mu = 1. \end{cases}$$

Therefore it is enough to identify the most significant bit of v. □

5.3 GSW Construction

The definition of the homomorphic accumulator (Definition 5.2.1) requires an encryption scheme with homomorphic properties for the increasing loop (step 3 of Algorithm 3). In this section we describe a homomorphic encryption scheme which is a variant of the construction given by Gentry, Sahai and Waters in [16].

The idea is to embed the LWE instance in a bigger group containing $\mathbb{Z}_q$ as subgroup. Suppose the message modulus of the LWE instance is 2, the ciphertext modulus is $q = 2^{d_1}$, and the dimension is m. Let $2^d = n > q$. The bigger group to be used is the multiplicative group of the ring $R = \mathbb{Z}[x]/\left(x^{\frac{n}{2}} + 1\right)$. The subgroup isomorphic to $\mathbb{Z}_q$ is the group generated by a qth root of unity, which is given by $y = x^{\frac{n}{q}}$. Recall that the polynomial $x^{\frac{n}{2}} + 1$ is the cyclotomic polynomial $\Phi_n(x)$ (since n is a power of 2). See Proposition 3.1.4.

Since $\mathbb{Z}_q$ is identified with a multiplicative group, we need to give an efficient way to homomorphically compute the multiplication of messages (this is the reason why a variant of the GSW encryption scheme is convenient for this application). We start by defining the main components of the construction. The reader may prefer to go straight to Definition 5.3.4, and leave Definitions 5.3.1-5.3.3 only as reference.

Definition 5.3.1 (B_g Decomposed Form of a Matrix)**.** Let $l, B_g \in \mathbb{Z}^+$ with $B_g < l$ and let $d_g = \left\lceil \log_{B_g} l \right\rceil$. Let $\mathbf{M} \in R_l^{2d_g \times 2}$. The *$B_g$-decomposed form* of $\mathbf{M}$ is the matrix

$$\text{Decomp}_{B_g}(\mathbf{M}) = \left[\mathbf{M}^{(0)} \middle| \mathbf{M}^{(1)} \middle| \cdots \middle| \mathbf{M}^{(d_g-1)}\right] \in R_l^{2d_g \times 2} \tag{5.3}$$

such that

$$\sum_{i=0}^{dg-1} \mathbf{M}^{(i)} d_g^i = \mathbf{M}.$$

Definition 5.3.2 (The $*$ Operation of Matrices)**.** Let $l, B_g \in \mathbb{Z}^+$ with $B_g < l$ and let $d_g = \left\lfloor \log_{B_g} l \right\rfloor$. Let $\mathbf{M}_1, \mathbf{M}_2 \in R_l^{2d_g \times 2}$. We define the operation $* : R_l^{2d_g \times 2} \times R_l^{2d_g \times 2} \longrightarrow R_l^{2d_g \times 2}$ by

$$*(\mathbf{M}_1, \mathbf{M}_2) = \mathbf{M}_1 * \mathbf{M}_2 = \text{Decomp}_{B_g}(\mathbf{M}_1)\, \mathbf{M}_2. \tag{5.4}$$

Definition 5.3.3 (Gadget Matrix)**.** Let $l, B_g \in \mathbb{Z}^+$ with $B_g < l$ and let $d_g = \left\lfloor \log_{B_g} l \right\rfloor$. Let $\mathbf{I}$ be the identity matrix in $R^{2\times 2}$. The *gadget matrix* $\mathbf{G}_{B_g}$ is the matrix

$$\mathbf{G}_{B_g} = \left[\mathbf{I} \middle| B_g \mathbf{I} \middle| \cdots \middle| B_g^{d_g-1} \mathbf{I}\right]^T. \tag{5.5}$$

Definition 5.3.4 (Bootstrapping Encryption Scheme)**.** Let $q, l, p, d, h \in \mathbb{Z}^+$ with p a prime number, $l \geq q$, $p^d \mid q$ and $h \in \mathbb{Z}_l^* \cap \left\{ \left\lfloor \frac{l}{2t} \right\rfloor, \left\lceil \frac{l}{2t} \right\rceil \right\}$ [1]. Let $y = x^{\frac{p^d}{q}} \in R_q$. Given a message $v \in \mathbb{Z}_q$ and a secret key $\sigma \in R_q$, sample $\boldsymbol{\alpha} \leftarrow R_q^{2d_g}$ uniformly at random and $\boldsymbol{\varepsilon} \leftarrow R_q^{2d_g}$ according to a probability distribution ψ

$$\mathsf{Enc}(\sigma, \mu) = (\boldsymbol{\alpha}, \sigma\boldsymbol{\alpha} + \boldsymbol{\varepsilon}) + hy^v\mathbf{G}. \tag{5.6}$$

Note that by multiplying by the gadget matrix $\mathbf{G}_{b_g}$ we can revert the decomposition of a matrix $\mathbf{M}$ since

$$\begin{aligned} \mathrm{Decomp}_{B_g}(\mathbf{M})\,\mathbf{G}_{b_g} &= \left[\mathbf{M}_1^{(0)} \middle| \mathbf{M}_1^{(1)} \middle| \cdots \middle| \mathbf{M}_1^{(d_g-1)}\right] \left[\mathbf{I} \middle| B_g\mathbf{I} \middle| \cdots \middle| B_g^{d_g-1}\mathbf{I}\right]^T \\ &= \mathbf{M}_1^{(0)}\mathbf{I} + B_g\mathbf{M}_1^{(1)}\mathbf{I} + \cdots + B_g^{d_g-1}\mathbf{M}_1^{(d_g-1)}\mathbf{I} \\ &= \mathbf{M}_1^{(0)} + B_g\mathbf{M}_1^{(1)} + \cdots + B_g^{d_g-1}\mathbf{M}_1^{(d_g-1)} \\ &= \mathbf{M}. \end{aligned}$$

Then we have the following remark.

Remark 5.3.5. For any matrix $\mathbf{M} \in R^{2d_g \times 2}$, the following identity holds:

$$\mathrm{Decomp}_{B_g}(\mathbf{M})\,\mathbf{G}_{b_g} = \mathbf{M}.$$

Proposition 5.3.6. *Let* $\mathbf{M}_1 = [\boldsymbol{\alpha}_1, \sigma\boldsymbol{\alpha}_1 + \boldsymbol{\varepsilon}_1] + \eta_1\mathbf{G}$ *and* $\mathbf{M}_2 = [\boldsymbol{\alpha}_2, \sigma\boldsymbol{\alpha}_2 + \boldsymbol{\varepsilon}_2] + \eta_2\mathbf{G}$ *be two matrices in* $R_l^{2d_g \times 2}$*. Then* $\mathbf{M}_1 * \mathbf{M}_2$ *is of the form*

$$\mathbf{M}_1 * \mathbf{M}_2 = [\boldsymbol{\alpha}_3, \sigma\boldsymbol{\alpha}_3 + \boldsymbol{\varepsilon}_3] + \eta_1\eta_2\mathbf{G} \tag{5.7}$$

where $\boldsymbol{\varepsilon}_3$ *is given by*

$$\mathit{Decomp}(\mathbf{M}_1)\,\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1.$$

Proof. By definition,

$$\begin{aligned} \mathbf{M}_1 * \mathbf{M}_2 &= \mathrm{Decomp}(\mathbf{M}_1)\,\mathbf{M}_2 \\ &= \mathrm{Decomp}(\mathbf{M}_1)\left([\boldsymbol{\alpha}_2, \sigma\boldsymbol{\alpha}_2 + \boldsymbol{\varepsilon}_2] + \eta_2\mathbf{G}\right) \\ &= \mathrm{Decomp}(\mathbf{M}_1)\,[\boldsymbol{\alpha}_2, \sigma\boldsymbol{\alpha}_2 + \boldsymbol{\varepsilon}_2] + \eta_2\mathrm{Decomp}(\mathbf{M}_1)\,\mathbf{G}. \end{aligned}$$

The term $\mathrm{Decomp}(\mathbf{M}_1)\,[\boldsymbol{\alpha}_2, \sigma\boldsymbol{\alpha}_2 + \boldsymbol{\varepsilon}_2]$ can be written as the matrix

$$\left[\mathrm{Decomp}(\mathbf{M}_1)\,\boldsymbol{\alpha}_2 \middle| \sigma\mathrm{Decomp}(\mathbf{M}_1)\,\boldsymbol{\alpha}_2 + \mathrm{Decomp}(\mathbf{M}_1)\,\boldsymbol{\varepsilon}_2\right]. \tag{5.8}$$

[1] Notice that $\mathbb{Z}_l^* \cap \left\{ \left\lfloor \frac{l}{2t} \right\rfloor, \left\lceil \frac{l}{2t} \right\rceil \right\}$ is non-empty.

On the other hand, by Remark 5.3.5 we have that

$$\begin{aligned} \eta_2 \text{Decomp}\left(\mathbf{M}_1\right)\mathbf{G} &= \eta_2(\mathbf{M}_1) \\ &= \eta_2\big(\left[\boldsymbol{\alpha}_1, \sigma\boldsymbol{\alpha}_1 + \boldsymbol{\varepsilon}_1\right] + \eta_1\mathbf{G}\big) \\ &= \eta_2\left[\boldsymbol{\alpha}_1, \sigma\boldsymbol{\alpha}_1 + \boldsymbol{\varepsilon}_1\right] + \eta_1\eta_2\mathbf{G}. \end{aligned} \tag{5.9}$$

Therefore, from Equations 5.8 and 5.9 it follows that

$$\begin{aligned} \mathbf{M}_1 * \mathbf{M}_2 \;=\; \Big[\text{Decomp}\left(\mathbf{M}_1\right)\boldsymbol{\alpha}_2 + \eta_2\boldsymbol{\alpha}_1 \Big| \sigma\big(\text{Decomp}\left(\mathbf{M}_1\right)\boldsymbol{\alpha}_2 + \eta_2\boldsymbol{\alpha}_1\big) \\ + \text{Decomp}\left(\mathbf{M}_1\right)\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1\Big] + \eta_1\eta_2\mathbf{G}. \end{aligned}$$

□

Corollary 5.3.7. *Let $v_1, v_2 \in \mathbb{Z}_q$ and let $\mathbf{C}_1 = [\boldsymbol{\alpha}_1, \sigma\boldsymbol{\alpha}_1 + \boldsymbol{\varepsilon}_1] + y^{v_1}\mathbf{G}$ and $\mathbf{C}_2 = [\boldsymbol{\alpha}_2, \sigma\boldsymbol{\alpha}_2 + \boldsymbol{\varepsilon}_2] + y^{v_2}\mathbf{G}$ be encryptions of v_1 and v_2. Then $\mathbf{C}_1 * \mathbf{C}_2$ is an encryption of $v_1 + v_2$ with error*

$$\textit{Decomp}\left(\mathbf{C}_1\right)\boldsymbol{\varepsilon}_2 + y\boldsymbol{\varepsilon}_1.$$

5.4 MSB Test

So far in the chapter we have described the construction of an encryption scheme with homomorphic properties for the purpose of using it in the bootstrapping procedure. However, there exists a fundamental difference between this approach to refresh a ciphertext and the one originally described by Gentry [15]: in this case we have to "collapse" the $\mathcal{E}$-ciphertext encrypting a message v to an LWE-ciphertext that encrypts the message $\mu = \text{MSB}(v)$. To explain the idea we first introduce some notation.

Notation 5.4.1. For $\alpha = a_0 + a_1 x + \ldots + a_{2^{d-1}-1}x^{2^{d-1}-1} \in R_l$, we denote by $\overrightarrow{\alpha}$ the vector

$$\overrightarrow{\alpha} = (a_0, a_1, \ldots, a_{2^{d-1}-1}) \in \mathbb{Z}_l^{2^{d-1}}. \tag{5.10}$$

We can also characterize α by the linear functional $L_\alpha \colon R \to R$ that results from multiplying by α. We denote by

$$\overrightarrow{\overrightarrow{\alpha}} \in \mathbb{Z}_l^{(2^{d-1})\times(2^{d-1})} \tag{5.11}$$

the matrix associated to L_α with respect to the standard basis.

The idea behind the "collapsing" process starts with the following fact (which can be directly derived from Corollary 3.1.5): for $i \in \{0, \ldots, 2^d - 1\}$, the vectors associated to the powers of x in $R = \mathbb{Z}[x]/\left(x^{2^{d-1}} + 1\right)$

$$\overrightarrow{x^i} = \begin{cases} \mathbf{e}_{i+1} & \text{if } i \in \{0, \ldots, 2^{d-1} - 1\}, \\ \mathbf{e}_{i-2^{d-1}+1} & \text{if } i \in \{2^{d-1}, \ldots, 2^d - 1\}. \end{cases}$$

Moreover, the sum of the entries of $\overrightarrow{x^i}$ is constant on each of these intervals, and can be easily recovered by computing the inner product with the MSB-*test vector* given by

$$\mathbf{t}_{\mathrm{MSB}} = -(1, 1, \ldots, 1) \in \mathbb{Z}_l^{2^{d-1}}. \tag{5.12}$$

Then we have that

$$\left\langle \mathbf{t}_{\mathrm{MSB}}, \overrightarrow{x^i} \right\rangle + 1 = \begin{cases} 0 & \text{if } i \in \{0, \ldots, 2^{d-1} - 1\}, \\ 2 & \text{if } i \in \{2^{d-1}, \ldots, 2^d - 1\}, \end{cases} \tag{5.13}$$

in other words

$$\left\langle \mathbf{t}_{\mathrm{MSB}}, \overrightarrow{x^i} \right\rangle + 1 = 2\,\mathrm{MSB}(i). \tag{5.14}$$

The second observation is that if $\mathbf{C} \in R^{2d_g \times 2}$ is an $\mathcal{E}$-encryption of $v \in \mathbb{Z}_q$, then the second row of $\mathbf{C}$ is of the form

$$[\alpha, \alpha\sigma + hy^v + \varepsilon]. \tag{5.15}$$

This follows directly from Definition 5.3.4.

Finally, observe that

$$\begin{aligned} \langle \mathbf{t}_{\mathrm{MSB}}, \overrightarrow{\alpha\sigma} \rangle &= \left\langle \mathbf{t}_{\mathrm{MSB}}, \ \overrightarrow{\overrightarrow{\alpha}\,\overrightarrow{\sigma}^T} \right\rangle \\ &= \mathbf{t}_{\mathrm{MSB}} \left(\overrightarrow{\overrightarrow{\alpha}} \cdot \overrightarrow{\sigma}^T \right) \\ &= \left(\mathbf{t}_{\mathrm{MSB}} \cdot \overrightarrow{\overrightarrow{\alpha}} \right) \overrightarrow{\sigma}^T \\ &= \left\langle \mathbf{t}_{\mathrm{MSB}} \overrightarrow{\overrightarrow{\alpha}}, \ \overrightarrow{\sigma} \right\rangle. \end{aligned}$$

Note that this equality is independent of the form of $\mathbf{t}_{\mathrm{MSB}}$. This proves the following remark.

Remark 5.4.2. For all $\mathbf{v} \in \mathbb{Z}_l^{2^{d-1}}$ and for all $\alpha, \sigma \in R_l$,

$$\left\langle \mathbf{v}, \ \overrightarrow{\alpha\sigma} \right\rangle = \left\langle \mathbf{v}\overrightarrow{\overrightarrow{\alpha}}, \ \sigma \right\rangle. \tag{5.16}$$

Definition 5.4.3 (MSB Test)**.** Let $n, l \in \mathbb{Z}^+$ and let h as in Definition 5.3.4. The MSB-*test* is the algorithm that takes as input a matrix $\mathbf{C} \in R^{2d_g \times 2}$ and the test vector $\mathbf{t}_{\text{MSB}}$, and computes the following.

Algorithm 4: The MSB test

Input : Matrix $\mathbf{C} \in R^{2d_g \times 2}$, test vector $\mathbf{t}_{\text{MSB}}$.
Output: Vector $\tilde{\mathbf{c}} = (\tilde{a}_0, \ldots, \tilde{a}_{n-1}, b) \in \mathbb{Z}_l^{n+1}$.

$\alpha_1 \leftarrow \mathbf{C}[0,1]$;
$\alpha_2 \leftarrow \mathbf{C}[1,1]$;
$(\tilde{a}_0, \ldots, \tilde{a}_{n-1}) \leftarrow \mathbf{t}_{\text{MSB}} \cdot \overrightarrow{\overrightarrow{\alpha_1}}$;
$b \leftarrow \langle \mathbf{t}_{\text{MSB}}, \overrightarrow{\alpha_2} \rangle + h$;
return $(\tilde{a}_0, \ldots, \tilde{a}_{n-1}, b)$;

Proposition 5.4.4. *Let $v \in \mathbb{Z}_q$ and let $\mathbf{C} \in R^{2d_g \times 2}$ be an encryption of v under the bootstrapping encryption scheme (see Definition 5.3.4). Then the* MSB *test, on input $\mathbf{C}$ and the test vector $\mathbf{t}_{\text{MSB}}$ outputs an $\mathsf{LWE}_{4:l}^{\psi'}$ encryption of the most significant bit of v.*

Proof. From the Remark 5.4.2 we have that $\langle \mathbf{t}_{\text{MSB}}, \overrightarrow{\alpha} \rangle = \left\langle \mathbf{t}_{\text{MSB}} \cdot \overrightarrow{\overrightarrow{\alpha}},\ \sigma \right\rangle$. From this and equations 5.14 and 5.15 we have that the output of Algorithm 4 is of the form

$$\Big(\mathbf{a}, \langle \mathbf{a}, \sigma \rangle + h\big(2\,\text{MSB}(v) - 1\big) + h + \langle \mathbf{t}_{\text{MSB}}, \varepsilon \rangle \Big)$$

where $\mathbf{a} = \mathbf{t}_{\text{MSB}} \cdot \overrightarrow{\overrightarrow{\alpha}}$. Then, since $h \approx \frac{l}{2t}$, this output is an LWE encryption of $\text{MSB}(v)$. □

Proposition 5.4.5. *Assuming that the* Ring-LWE *problem is hard on R_l, then the homomorphic accumulator scheme defined above is ϵ-correct, where*

$$\epsilon(k) = \sqrt{\frac{Nq^2}{l^2}\Big(\theta_g^2 \cdot k \cdot q \cdot B_g^2 \cdot d_g + \theta_{ks}^2 \cdot d_{ks}\Big) + |bs|^2} \cdot \omega\sqrt{\log n}.$$

The proof of this proposition can be found in [11]. It requires to first study the evolution of the error after every step of the homomorphic accumulator to bound the final noise after the MSB test.

5.5 Other Gates

In Section 5.1 we described a new way to homomorphically compute the NAND gate introduced by Ducas and Micciancio [11]. Since the NAND gate is universal, this is enough

for the cryptosystem to be fully-homomorphic (see Definition 4.1.1). However, other gates can be efficiently computed and refreshed by the homomorphic accumulator using a similar technique.

We skip the proof of the following theorem since it is analogous to the proof of Theorem 5.1.1.

Theorem 5.5.1. *Let $r \geq 4$ be a positive integer and let $q = 2^r$. Let $n \in \mathbb{Z}^+$ and let $\mathbf{s} \in \mathbb{Z}_q^n$. For $i \in \{0, 1\}$ let $\mu_i \in \{0, 1\}$ and $\mathbf{c}_i = (\mathbf{a}_i, b_i) \in \mathsf{LWE}_{4:q}^{\frac{q}{16}}(\mathbf{s}, \mu_i)$. Then the pair*

$$\mathsf{homAND}(\mathbf{c}_0, \mathbf{c}_1) = \left(\mathbf{a}_0 + \mathbf{a}_1, \frac{q}{8} - b_0 - b_1\right) \tag{5.17}$$

is an element of $\mathsf{LWE}_{2:q}^{\frac{q}{4}}(\mathbf{s}, \mu_1 \wedge \mu_2)$. Similarly, the pair

$$\mathsf{homNOR}(\mathbf{c}_0, \mathbf{c}_1) = \left(-\mathbf{a}_0 - \mathbf{a}_1, \frac{3q}{8} - b_0 - b_1\right) \tag{5.18}$$

is an element of $\mathsf{LWE}_{2:q}^{\frac{q}{4}}(\mathbf{s}, \mu_1 \veebar \mu_2)$.

One interesting circuit to implement is the add-and-carry circuit (which will be computed with a single gate in the next section). Since the NAND gate is universal it is possible to construct the add-and-carry circuit using only NAND gates, but it would take at least 9 of them. Using NOT, NOR, OR and AND it is possible to construct a circuit using only 7 gates, which significantly reduces the running time.

Chapter 6

Homomorphic Accumulator for Multi-Bit Bootstrapping

In Chapter 5 we described in detail a fully-homomorphic cryptosystem based on LWE. The construction was given by Ducas and Micciancio [11] in 2015, and it has the novelty of having a very low-cost bootstrapping procedure that is applied after every gate. It offers the best possible level of granularity and it has the advantage that it frees the user from having to calculate where to perform the bootstrapping operation. The cryptosystem has, however, some disadvantages: it provides a very limited family of gates that can be bootstrapped by their method called the homomorphic accumulator. In [11] the authors only describe a way to perform the operation after the NAND gate. However, we showed in Section 5.5 it can be easily modified to refresh the ciphertext after other gates are applied, such as NOT, AND and NOR.

In this chapter we introduce a generalization of the homomorphic accumulator used by Ducas and Micciancio in [11] to perform bootstrapping. We use the special representation of the cyclotomic polynomial $\Phi_p(x)$, where p is a prime, given in Proposition 3.1.4, to compute a generalization of the MSB test described in Section 5.4, which can be seen as a "set-membership" test. This generalization allows us to homomorphically compute the bits corresponding to a message from $\mathbb{Z}_p$. Using the computation of the separate bits of a message and the homomorphic properties of LWE described in Section 3.2, we can perform the bootstrapping operation after any arbitrary circuit.

To understand the content of this chapter we assume that the reader has basic knowledge of Chapter 5 or [11]. However, we warn the reader of a change of notation between [11] and this work. The results in this chapter previously appeared in a paper by Biasse

and Ruiz [4].

6.1 General Set-Membership Test

The core of our construction is a generalization of the MSB test (see Definition 5.4.3). The MSB test is used as the msbExtract procedure in the GSW construction to collapse an $\mathcal{E}$-encryption of an element $v \in \mathbb{Z}_q$ into an LWE-encryption of $\mathrm{MSB}(v)$. For our generalization we use a wider family of functions that allows us to refresh LWE-ciphertext encrypting messages in $\mathbb{Z}_p$.

Definition 6.1.1 (Characteristic Function)**.** Let S be a set and $S' \subseteq S$. The *characteristic function* $\chi_{S'} \colon S \to \{0,1\}$ is the function defined by

$$\chi_{S'}(s) = \begin{cases} 1 & \text{if } s \in S' \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that n is even. Then the MSB function can be seen as the characteristic function

$$\mathrm{MSB}(v) = \chi_{\{\frac{n}{2},\ldots,n-1\}}(v)$$

for $v \in \{0,\ldots,n-1\}$. This family of functions is enough to recover the message given the homomorphic properties of the LWE encryption: suppose that we want to refresh an LWE ciphertext $(\mathbf{a}, b)$ encrypting a message $\mu \in \mathbb{Z}_p$, and assume we can obtain, after the "increasing" loop in the refreshing procedure (Algorithm 3) the LWE encryptions $c_0,\ldots,c_{p-1}$ of $\chi_0(\mu),\ldots,\chi_{p-1}(\mu)$, respectively. Then using the fact that

$$\mu = \sum_{i=1}^{p-1} i\chi_i(\mu)$$

and the homomorphic properties of LWE (Proposition 3.2.6), we have that $\sum_{i=1}^{p-1} ic_i$ is an LWE encryption of μ, provided that some bound on the noise is satisfied.

Recall that the encryption function of LWE maps elements from $\mathbb{Z}_p$ to $\mathbb{Z}_q$ using a (randomized) scaling function and adding some noise. Then the elements of $\mathbb{Z}_q$ can be seen as "scaled versions of $\mathbb{Z}_p$", and the characteristic function χ_i over $\mathbb{Z}_p$ corresponds to the characteristic function χ_S, for some $S \subseteq \mathbb{Z}_q$. To collapse an $\mathcal{E}$-encryption of an element $v \in \mathbb{Z}_q$ into an LWE-encryption of $\chi_S(v)$, we use some special algebraic properties of the cyclotomic ring R_l.

Let $m = p^d$, then the cyclotomic polynomial Φ_m is of the form

$$\Phi_{p^d}(x) = \sum_{j=0}^{p-1} x^{jp^{d-1}}.$$

This implies that the powers of x in R_l "cycle in two different intervals". More precisely, if $exp\colon \mathbb{Z} \to R$ maps $i \mapsto x^i$, we have beforehand that this function has period m, since x is an mth primitive root of unity. However, the set $\{0, \dots, m-1\}$ is mapped as follows

$$\begin{array}{ccc}
0 & \mapsto & 1 \\
1 & \mapsto & x \\
2 & \mapsto & x^2 \\
\vdots & \vdots & \vdots \\
\varphi(m)-1 & \mapsto & x^{\varphi(m)-1} \\
\varphi(m) & \mapsto & -1 - x^{\frac{m}{p}} - \ldots - x^{(p-2)\frac{m}{p}} \\
\vdots & \vdots & \vdots \\
m-1 = \varphi(m)+p-1 & \mapsto & -x^{\frac{m}{p}-1} - \ldots - x^{(p-1)\frac{m}{p}-1}.
\end{array}$$

We then have as a corollary that the vectors associated to the powers of x over R_l are of the form

$$\overrightarrow{x^i} = \begin{cases} \mathbf{e}_i & \text{if } i \in \left\{0, \dots, \varphi(p^d) - 1 = (p-1)p^{d-1} - 1\right\}, \\ \sum_{j=0}^{p-2} \mathbf{e}_{j \cdot p^{d-1}+i} & \text{if } i \in \left\{\varphi(p^d) = (p-1)p^{d-1}, \dots, p^d\right\}. \end{cases}$$

The goal is to recover some information about the exponent of x. We can try to do so by using Remark 5.4.2 as we did for the MSB test in Chapter 5. It is thus natural to look at the inner product of $\overrightarrow{x^i}$ with the vectors

$$\begin{array}{ccccc}
\mathbf{v}_0 & = & \overbrace{(\underbrace{1, \dots, 1}_{\frac{m}{p}}, 0, \dots, 0)}^{\varphi(n)=(p-1)\frac{m}{p}} & = & \sum_{k=1}^{\frac{m}{p}} \mathbf{e}_k, \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{v}_i & = & \overbrace{(\underbrace{0, \dots, 0}_{i\frac{m}{p}}, \underbrace{1, \dots, 1}_{\frac{m}{p}}, 0, \dots, 0)}^{\varphi(m)=(p-1)\frac{m}{p}} & = & \sum_{k=i\frac{m}{p}+1}^{(i+1)\frac{m}{p}} \mathbf{e}_k, \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{v}_{p-2} & = & (0, \dots, 0, 1, \dots, 1) & = & \sum_{k=(p-2)\frac{m}{p}+1}^{(p-1)\frac{m}{p}} \mathbf{e}_k.
\end{array} \tag{6.1}$$

For $i \in \{0, \ldots, p-2\}$ let

$$S_i = \left\{ i\frac{m}{p} + 1, \ldots, (i+1)\frac{m}{p} \right\} \tag{6.2}$$

be the set of non-zero entries of $\mathbf{v}_i$. We then have

$$\left\langle \overrightarrow{x^i}, \mathbf{v}_j \right\rangle = \begin{cases} 1 & \text{if } i \in S_j, \\ 0 & \text{if } i < \varphi(m) \text{ and } i \notin S_j, \\ -1 & \text{if } i \in \{\varphi(m), \ldots, m-1\}. \end{cases} \tag{6.3}$$

This calculation would allow us to recover the right information for $i \in \{0, \ldots, \varphi(m)-1\}$, but it still fails for $i \in \{\varphi(m), \ldots, m-1\}$. To fix this shortcoming, note that if

$$\mathbf{v} = (1, \ldots, 1) \in \mathbb{Z}_l^{\varphi(m)}, \tag{6.4}$$

then

$$\left\langle \overrightarrow{x^i}, \mathbf{v} \right\rangle = \begin{cases} 1 & \text{if } i \in \{0, \ldots, \varphi(m)-1\}, \\ -p+1 & \text{if } i \in \{\varphi(m), \ldots, m-1\}. \end{cases} \tag{6.5}$$

Putting the equations 6.3 and 6.5 together we have that

$$\left\langle \overrightarrow{x^i}, p\mathbf{v}_j - \mathbf{v} \right\rangle = p\left\langle \overrightarrow{x^i}, \mathbf{v}_j \right\rangle - \left\langle \overrightarrow{x^i}, \mathbf{v} \right\rangle = \begin{cases} p-1 & \text{if } i \in S_j, \\ -1 & \text{if } i < \varphi(m) \text{ and } i \notin S_j, \\ -1 & \text{if } i \in \{\varphi(m), \ldots, m-1\}. \end{cases} \tag{6.6}$$

Moreover, using $-\mathbf{v}$, by Equation 6.5 we can identify whether or not $i \in \{\varphi(m), \ldots, m-1\}$ in the same way since $\left\langle \overrightarrow{x^i}, -\mathbf{v} \right\rangle = -\left\langle \overrightarrow{x^i}, \mathbf{v} \right\rangle$. This motivates the following definition.

Definition 6.1.2 (Test Vectors)**.** For $i \in \{0, \ldots, p-2\}$ we define the *ith test vector* to be

$$\mathbf{t}_i = p\left(\sum_{j=ip^{k-1}}^{(i+1)p^{k-1}-1} \mathbf{e}_j \right) - (1, \ldots, 1)\,.$$

Using these test vectors, the following definition is a generalization of Algorithm 4.

Definition 6.1.3 (Membership Test)**.** Let $n = p^d, l \in \mathbb{Z}^+$ and let $h \in \mathbb{Z}_l^* \cap \{ \left\lfloor \frac{l}{p \cdot t} \right\rfloor, \left\lceil \frac{l}{p \cdot t} \right\rceil$. The *ith-membership test* is the algorithm that takes as input a matrix $\mathbf{C} \in R^{2d_g \times 2}$ and the

test vector $\mathbf{t}_i$, and computes the following.

Algorithm 5: The membership test

Input : Matrix $\mathbf{C} \in R^{2d_g \times 2}$, test vector $\mathbf{t}_i$.
Output: Vector $\tilde{\mathbf{c}} = (\tilde{a}_0, \ldots, \tilde{a}_{n-1}, b) \in \mathbb{Z}_l^{n+1}$.

$\alpha_1 \leftarrow \mathbf{C}[0,1]$;
$\alpha_2 \leftarrow \mathbf{C}[1,1]$;
$(\tilde{a}_0, \ldots, \tilde{a}_{n-1}) \leftarrow \mathbf{t}_{\text{MSB}} \cdot \overrightarrow{\overrightarrow{\alpha_1}}$;
$b \leftarrow \langle \mathbf{t}_{\text{MSB}}, \overrightarrow{\alpha} \rangle + h$;
return $(\tilde{a}_0, \ldots, \tilde{a}_{n-1}, b)$;

Note that the inner product given in equations 6.5 and 6.6 is not quite the result we need. Ideally we would like a test to output 1 if $i \in S_j$ and 0 otherwise (as in the characteristic function). However we can use a trick similar to the one used for the MSB test in Chapter 5, introducing an invertible factor $h \in \mathbb{Z}_l$ that is close to $\frac{l}{p \cdot t}$. Observe that $\left\{ \left\lfloor \frac{l}{p \cdot t} \right\rfloor, \left\lceil \frac{l}{p \cdot t} \right\rceil \right\} \cap \mathbb{Z}_l^*$ is non empty, since no consecutive numbers can have a common divisor.

Proposition 6.1.4. *Let $v \in \mathbb{Z}_q$ and let $\mathbf{C} \in R^{2d_g \times 2}$ an encryption of v under the bootstrapping encryption scheme (see Definition 5.3.4). Let $h \in \left\{ \left\lfloor \frac{l}{p \cdot t} \right\rfloor, \left\lceil \frac{l}{p \cdot t} \right\rceil \right\} \cap \mathbb{Z}_l^*$ and let $i \in \{0, \ldots, p-1\}$. Then the membership test, on input an $\mathbf{C}$ and the ith test vector $\mathbf{t}_i$ outputs an* LWE*-encryption of $\chi_{S_i}(v)$.*

Proof. Using Equation 6.6, the proof is completely analogous to the proof of Proposition 5.4.4. □

In the previous proposition, the error is given by $\langle \mathbf{t}_i, \varepsilon \rangle$. However, in order to bound it we first have to describe the evolution of ε on every step of the accumulator.

Lemma 6.1.5 (Norm of Decomposed Matrices)**.** *Let $\mathbf{D}^{(i)} = \left[\mathbf{D}_1 | \cdot | \mathbf{D}_{d_g} \right]$ where $h^{-1}\text{Acc}^{(i)} = \sum_{i=1}^{d_g} B_g^j \mathbf{D}_j$. Assume that for $i \in \{1, \ldots, d_g\}$, the ciphertext $\mathbf{C}^{(i)}$ is computationally indistinguishable from random. Then the bound*

$$S_1 \left[\mathbf{D}^{(0)} | \ldots | \mathbf{D}^{(k)} \right] = O\left(B_g \sqrt{N \cdot d_g \cdot k} \right) \tag{6.7}$$

on the singular norm holds with overwhelming probability for any function $k \geq \omega(\sqrt{\log n})$.

Proof. The proof is analogous to the proof of Fact *2* of [11]. □

Table 6.1: Truth table of the one-bit full adder

b_1	b_2	c_i	s	c_o
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Lemma 6.1.6 (Intermediate Error of the Collapsing Procedure)**.** *Assume the hardness of* Ring-LWE *and let* Acc *be a k-encryption of $v \in \mathbb{Z}_q$, where $q \geq \omega\left(\sqrt{\log n}\right)$. Then the ciphertext* $\mathbf{c}$ *defined by the collapsing procedure has an error*

$$\epsilon(\mathbf{c}) = O\left(\theta_g B_g \sqrt{q \cdot n \cdot d_g \cdot k}\right). \tag{6.8}$$

6.2 Full Adder Gate

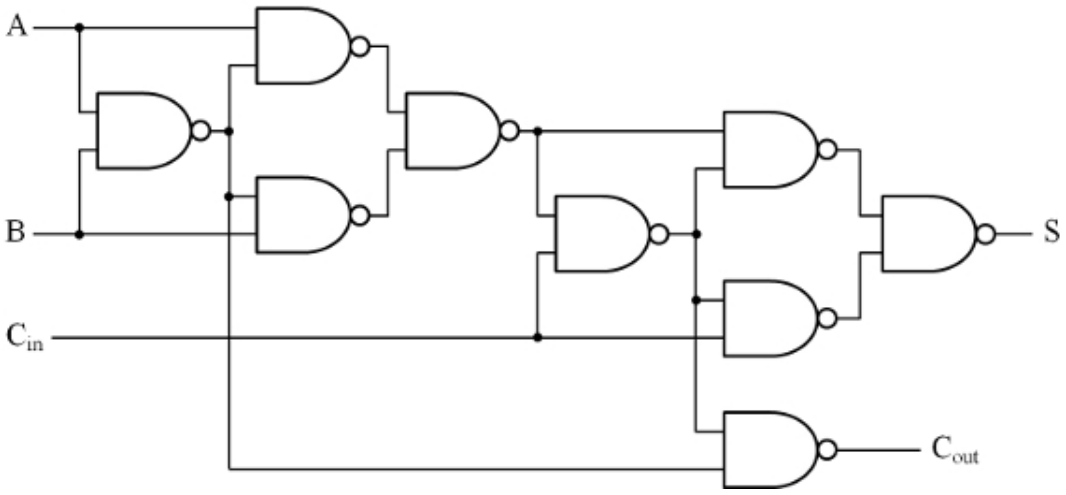
Addition of integers is one of the fundamental operations performed by computers. An *adder* is a circuit that performs addition of numbers. We are particularly interested in the one-bit full adder, that computes the sum of two one-bit numbers $b_1, b_2 \in \{0, 1\}$ and an input carry c_i, outputting bits $s, c_o \in \{0, 1\}$ such that $b_1 + b_2 + c_i = 2c_o + s$.

Definition 6.2.1 (One-Bit Full Adder)**.** The *one-bit full adder* is the circuit that takes 3 bits as input b_1, b_2, c_i and outputs two bits s, c_o, according to Table 6.2.1.

There are several possible implementations of the full-adder circuit using boolean gates. Moreover, since the NAND gate is universal, it is possible to implement the full-adder using only NAND gates. The minimal possible implementation of such a circuit is described in Figure 6.2.

By using a large enough modulus together with the membership tests, it is possible to implement the full adder circuit in one single gate running the homomorphic accumulator just once, thus bootstrapping more than one bit at the time and (potentially) reducing the

Figure 6.1: Full adder with NAND gates



running time compared to an implementation using only NAND gates.

Algorithm 6: Homomorphic full adder gate

Input : $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_1)$, $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_2)$, $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(c)$

Output: $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_3)$, $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(c')$, digit and carry of $b_1 + b_2 + c$

1 $(\mathbf{a}, b) \leftarrow \mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_1) + \mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_2) + \mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(c)$;

2 Call Algorithm 3 on $(\mathbf{a}, b) = \mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(\mu)$;

3 $\mathsf{LWE}^{\psi}_{\mathbf{z}:t/Q}(c) \leftarrow \mathsf{LWE}^{\psi}_{\mathbf{s}:t/Q}(\chi_{\{2,3\}}(\mu))$;

4 $\mathsf{LWE}^{\psi}_{\mathbf{z}:t/Q}(b_3) \leftarrow \mathsf{LWE}^{\psi}_{\mathbf{z}:t/Q}(\chi_{\{0,1\}}(\mu))$;

5 Switch key and modulus on $\mathsf{LWE}^{\psi}_{\mathbf{z}:t/Q}(b_3), \mathsf{LWE}^{\psi}_{\mathbf{z}:t/Q}(c)$;

6 **return** $\mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(b_3), \mathsf{LWE}^{\psi}_{\mathbf{s}:t/q}(c)$;

Chapter 7

Implementation and Benchmarks

In [11], Ducas and Micciancio reported on an implementation of their encryption scheme. This implementation is special for two main reasons: it is simple and short (taking just a few hundred lines of code), and it performs a complete bootstrapping procedure in less than a second. Although the amortized cost-per-operation is not yet faster than that reached by other implementations [19], the theoretical and practical importance of the cryptosystem is undeniable. The code of Ducas and Micciancio is a C++ implementation available online. Most of the routines are original, using only the help of the FFTW 3 library to carry out the polynomial multiplications.

We modified their code to improve its performance and extend its capabilities. Using the OpenMP library, we parallelize the most expensive operation, improving the running time of the algorithm by roughly 4 times, allowing further parallelization by the user when constructing circuits and giving the possibility of running time improvements by modifying the parameters of the homomorphic accumulator.

In this chapter we explain the main details of the original implementation given by Ducas and Micciancio, and the improvement that we gave which also has the ability to refresh LWE ciphertext encrypting messages in $\mathbb{Z}_p$ using the techniques explained in Chapter 6. We conclude the chapter with comparative benchmarks of both cryptosystems.

7.1 Fast Fourier Transform and Polynomial Multiplication

The homomorphic accumulator is a bootstrapping procedure used to refresh LWE ciphertexts. In Chapters 5 and 6 we described a realization of this procedure using an encryption scheme based on Ring-LWE. In Algorithm 3 several polynomial multiplications are required when implemented with the homomorphic accumulator defined in Section 5.3. However, carrying out multiplications in the naive way requires a quadratic number of operations. An optimized way to multiply polynomials is to use a fast Fourier transform algorithm. To understand how this algorithm helps to compute polynomial multiplications, we first explain what the discrete Fourier transform is.

Definition 7.1.1 (Discrete Fourier Transform)**.** Let $\omega_1, \ldots, \omega_m \in \mathbb{C}$ be a sequence of complex numbers. The *discrete Fourier transform* is the sequence

$$\left\{ X_k = \sum_{j=0}^{n-1} \omega_j \cdot e^{-2\pi i \frac{jk}{n}} : \text{ for } k \in \mathbb{Z} \right\}.$$

It is easy to see that the sequence is periodic, and in fact actually determined by the first n terms. Hence, the Fourier transform can be seen as an (invertible) linear transformation $\mathcal{F}_n : \mathbb{C}^n \longrightarrow \mathbb{C}^n$. In fact, the matrix associated to this linear function is the Vandermonde matrix

$$\mathbf{M}_{\mathcal{F}_n} = \begin{pmatrix} 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^{n-1} \\ 1 & \zeta_2 & \zeta_2^2 & \cdots & \zeta_2^{n-1} \\ 1 & \zeta_3 & \zeta_3^2 & \cdots & \zeta_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta_n & \zeta_n^2 & \cdots & \zeta_n^{n-1} \end{pmatrix}.$$

Note that if $\alpha(x) = a_0 + a_1 x + \ldots + a_n x^n$, then multiplying the vector $(a_0, \ldots, a_n)^T$ on the left by the matrix $\mathbf{M}_{\mathcal{F}_n}$ is equivalent to compute the image of α under the canonical embedding (see Equation 3.6). Therefore, once the Fourier transform is computed, polynomial multiplication is transformed to component-wise multiplication (see Equation 3.7)

A *fast Fourier transform* is an algorithm that computes the discrete Fourier transform, the most common being the Cooley-Tukey algorithm, whose running time is $O(n \ln n)$. The details of the Cooley-Tukey algorithm are out of the scope this thesis, but they can be found in [8].

The fast Fourier transform implementation that is used by Ducas and Micciancio in the implementation of their cryptosystem, as well as in our code, is the FFTW3 library, which is specially optimized for powers of 2 (due to the nature of the Cooley-Tukey algorithm). However, the performance of this implementation remains acceptable if the dimension of the problem is of the form $2^a 3^b 5^c 7^d 11^e 13^f$, where $e + f \in \{0, 1\}$ [13].

7.2 Use of OpenMP

The Ducas-Micciancio cryptosystem [11] was the first construction to report a complete bootstrapping procedure in less than a second. However, since this operation has to be performed right after (almost) every gate is computed, the amortized cost makes the implementation impractical to use in most real-world situations. After profiling of the given code, it is clear that most of the computing time is spent on the bootstrapping step (which is expected), particularly, more than 99% of the computing time is spent on the "increasing loop" (step 2 of Algorithm 3) that calls the incr function every time $a_{i,j}$ in the decomposition

$$-a_i = \sum_{j=0}^{d_g - 1} a_{i,j} B_g^j$$

is non zero. Recall that the incr function constructed in Section 5.3 consists of decomposing the current value of Acc and multiplying by the bootstrapping key corresponding to $a_{i,j}$ (see definitions 5.2.1 and 5.3.2). The matrix multiplication step can be parallelized into as many threads as the dimension of the square matrix (the decomposed for of the accumulator), which is $2d_g$ (see Definition 5.3.1).

To parallelize the incr routine we used OpenMP. After several experiments we realized that the most efficient tool for this purpose is `parallel sections`. We assigned each row multiplication to a different section, whose computation is carried out by one individual thread.

7.3 Benchmarks

In this section we present comparisons of the efficiency of the two encryption schemes in the case of the full adder circuit, and the influence that parallelization has on the implementation.

In Table 7.1 we present the computation time for the implementation of the Ducas-Micciancio cryptosystem [11] described in Chapter 5 when the incr step of the homomorphic accumulator is parallelized into several threads. In our experiments, a single matrix multiplication is roughly 4 times faster when the computation is divided into 6 threads. In the table we can observe a similar speedup on the overall computation time of the cryptosystem (that includes encryption, homomorphic computation of NAND, bootstrapping and decryption).

Table 7.1: Influence of parallel computation on the NAND gate

Number of threads	1	2	3	4	5	6
CPU time (sec)	1.5446	0.7504	0.6913	0.6761	0.5941	0.4176

In Table 7.2 we present the computation time for our cryptosystem when the full-adder is evaluated, and how the performance is improved when the same increasing procedure is divided into several threads and how it is compared to the implementation of the same circuit using only NAND gates as showed in Figure 6.2.

Table 7.2: Benchmark on the homomorphic full adder

Number of threads	1	2	3	4	5	6
full adder with 9 NAND gates	13.901	6.754	6.222	6.05	5.347	3.758
Full adder gate	4.374	2.421	1.691	1.725	1.697	0.993

Chapter 8

Conclusions and Future Work

In the last seven years, fully-homomorphic encryption has evolved from a merely theoretical construction to constructions that are better understood and allow implementations that can even be considered for some real-life applications. However, the best existing constructions and implementations are still far from being considered practical in most scenarios, such as cloud computing, and the application of fully-homomorphic encryption in other cryptographic primitives such as obfuscation, fully-homomorphic signatures, etc.

As it was mentioned before in Chapter 4 the bottleneck of all existing cryptosystems is the bootstrapping step. A long standing question is the possibility of a fully-homomorphic cryptosystem that does not require bootstrapping. However, a more realistic approach is the improvement of the existing bootstrapping techniques.

From the theoretical point of view, an outstanding problem is the extension of the construction given by Ducas and Micciancio [11] to the entire set of positive integers. From the practical point of view, the implementation we provide may as well be improved by using different algorithms to carry out polynomial multiplications, that can possibly result in a more efficient implementation of the homomorphic accumulator.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[2] Miklós Ajtai. The shortest vector problem in $L2$ is NP-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 10–19, New York, NY, USA, 1998. ACM.

[3] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 284–293, New York, NY, USA, 1997. ACM.

[4] Jean Francois Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In *Advances in Cryptology - LATINCRYPT 2015*, pages 119–135, Berlin, Heidelberg, 2015. Springer-Verlag.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[7] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[8] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965. URL: `http://cr.yp.to/bib/entries.html#1965/cooley`.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[10] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In JuanA. Garay and Rosario Gennaro, editors, *Advances in Cryptology CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 335–352. Springer Berlin Heidelberg, 2014.

[11] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 617–640, 2015.

[12] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[13] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. 93(2):216–231, February 2005.

[14] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 31–51, Berlin, Heidelberg, 2008. Springer-Verlag.

[15] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[16] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[17] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK, 1997. Springer-Verlag.

[18] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[19] Shai Halevi and Victor Shoup. Bootstrapping for HElib. Cryptology ePrint Archive, Report 2014/873, 2014. http://eprint.iacr.org/.

[20] Guillaume Hanrot, Xavier Pujol, and Damien Stehl. Analyzing blockwise lattice algorithms using dynamical systems. In *CRYPTO'11*, pages 447–464, 2011.

[21] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[22] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *In TCC 2007*, 2007.

[23] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 193–206, New York, NY, USA, 1983. ACM.

[24] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[25] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 144–155, Berlin, Heidelberg, 2006. Springer-Verlag.

[26] Daniele Micciancio. Generalized compact knapsaks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, December 2007. Prelim. in FOCS 2002.

[27] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[28] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In D. J. Bernstein and J. Buchmann, editors, *Post-quantum Cryptography*. Springer, 2008.

[29] Phong Q. Nguyen and Jacques Stern. Cryptanalysis of the ajtai-dwork cryptosystem. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 1998.

[30] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[31] Michael Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull.*, 15(1):37–44, February 1981.

[32] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.

[33] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[34] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations on Secure Computation, Academia Press*, pages 169–179, 1978.

[35] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In Yuval Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234. Springer Berlin Heidelberg, 2011.

[36] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66(2):181–199, September 1994.

[37] Adi Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 145–152, Washington, DC, USA, 1982. IEEE Computer Society.

[38] Peter Van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, University of Amsterdam, 1981.