INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Homomorphic encryption

Eduardo Morais

Ricardo Dahab

Technical Report - IC-2016-2 - Relatório Técnico

April - 202016 - Abril

Homomorphic encryption

Eduardo Morais

Ricardo Dahab

April, 2016

Abstract

In 2009, Gentry [Gen09b] constructed the first ***fully homomorphic encryption*** (FHE) scheme, solving a conjecture that remained open since 1978 when it was proposed by Dertouzos et al [RAD78]. The cryptographic construction is important because it allows to compute arbitrary algorithms over encrypted data. It is based on hard problems over ideal lattices and hence is part of the ***post-quantum*** cryptography. In this document we describe the construction under the assumption that the ***approximate GCD problem*** is hard, following the same blueprint originally described in Gentry's work. We think the AGCD-based scheme allows an easier understanding of the concepts involved in the construction of FHE cryptosystems. Even though no FHE proposal till now turned out to be a practical scheme, we are going to show how to build ***somewhat homomorphic encryption*** (SHE) based on the ***learning with errors*** (LWE) problem, which can be used to evaluate a restricted class of algorithms over encrypted data. We also discuss lattice attacks to homomorphic encryption schemes and how the choice of parameters is determined based on the best-attack effort estimation. We compare variants of the main construction and show that each one has advantages and disadvantages depending on the application and on the concrete scenario under which the cryptosystem is implemented.

1 Introduction

After Edward Snowden revelations in 2013, concernment should be growing about how private the internet is and how it will be in the future. It became clear that cryptographic standards where influenced by governments and that companies cooperated in this process of implementing systems that we suppose are secure and private, but in practice turn out to be severely flawed. Privacy is essencial to grant the well-functioning of a democratic state and hence everyone should be worried with the way the internet is being built. Rogaway [Rog15] recently argued that the cryptographic community should be paying more attention to problems related to the privacy of users, rather than to the security of companies.

In the cloud computing scenario, conventional encryption schemes can be used to provide privacy to sensitive information, but normally at the cost of losing basically all the functionality, because data must be in the clear in order to do something interesting with it, as for example computing a function or evaluating an algorithm that receives this data as input. In some sense, cryptography seems to be orthogonal to functionality. However, we are going to show that ***fully homomorphic encryption*** (FHE) allows to compute ***any*** algorithm over encrypted data and hence it is a candidate to solve this problem, since it reaches both privacy and ***maximal*** functionality. Unfortunatelly, the solution has a problem, because it is too expensive in terms of computational resources. In particular, it would be interesting to have the capacity to design a scheme providing limited functionality (instead of maximal), but such that it is enough for an specific purpose. It would allow us to know the minimal overhead in the resources cost when compared to computing over the plaintext data. Then we could decide if this resource cost is affordable or not.

Thus, the construction of FHE is a theoretical big and important advance to cryptology, because many interesting techniques were used and many beautiful ideas and concepts were introduced. Nevertheless, we must know how it could be directly used to increase privacy in practice. Thus, we first need to investigate how to transform FHE into practical cryptosystems. In order to do that we have to understand the inherent tradeoffs among security, efficiency and functionality of homomorphic encryption schemes, where the last one is the measure of how much homomorphic computation can be done over encrypted data. This comprehension allows us to design more efficient systems, but with less functionality. For instance, we will show how to construct schemes that have an upper bound on the number of homomorphic operations that they can deal with. Such schemes are called ***somewhat homomorphic encryption*** (SHE). Therefore, the motivation to study homomorphic encryption is twofold: firstly, by considering the theoretical contributions and new ideas involved in the construction of FHE, we can comprehend how expensive it is to achieve such kind of functionality in the design of cryptographic primitives; secondly, by understanding the hardness of the underlying problems and the algorithms that solve them, we can decide how to choose parameters in order to obtain feasible solutions to be used in practical scenarios.

Before Gentry's work [Gen09b], many SHE proposals appeared in the literature, many were proven to be insecure and others achieved very limited functionality. Interestingly, there were a continuous progress in the construction of homomorphic encryption since the definition of ***privacy homomorphisms*** by Dertouzos et al in 1978 [RAD78]. Hence, we begin the paper describing previous constructions in order to gradually introduce notation and the semantics of our algorithms and problems. We also give some mathematical background and general definitions. Afterwards, we describe the construction of homomorphic encryption over the integers, using the ***approximate GCD problem*** (AGCD), and also over lattices, with focus on the BGV scheme, which is based on the ***LWE problem***. For each construction we are going to describe attacks that are used to derive concrete parameters to achieve a certain security level.

1.1 Organization

This document is organized as follows. In Section 2, we describe constructions proposed before Gentry's work. We also give the main definitions that compose Gentry's blueprint. In Section 3, we show how to achieve fully homomorphic encryption under the hyphotesis that the AGCD problem is hard. In Section 4 we present the BGV scheme in detail, giving concrete parameters to instantiate the construction.

2 From 1976 till 2009

If Alice and Bob want to communicate over an insecure channel, they should use a symmetric cryptographic scheme to protect exchanged messages against an eavesdropper. In order to do that, they must use a shared secret k, generated by an algorithm called KEYGEN. We can define the domains $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{C}$, respectively as the key space, from where algorithm KEYGEN computes its outputs, the plaintext space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. Furthermore, we can define the encryption algorithm $\textsc{Enc} : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ and the decryption algorithm $\textsc{Dec} : \mathcal{C} \times \mathcal{K} \to \mathcal{M} \cup \{\perp\}$, such that $\textsc{Dec}_k(\textsc{Enc}_k(m)) = m \cup \{\perp\}$, where the symbol $\perp$ is used to denote the case when the ciphertext is an invalid input to the decryption algorithm.

In 1976, Diffie and Hellman [DH76] published the famous article ***New directions in cryptography***, introducing the concept of public key cryptography (asymmetric cryptography). In this model, Alice uses the algorithms KEYGEN to generate a key pair $(\texttt{sk}_A, \texttt{pk}_A) \in \mathcal{K}_{\text{priv}} \times \mathcal{K}_{\text{pub}}$. The private key $\texttt{sk}_A$ must be maintained secret while the public key $\texttt{pk}_A$ may be published. Encryption and decryption algorithms are defined respectively by $\textsc{Enc} : \mathcal{M} \times \mathcal{K}_{\text{pub}} \to \mathcal{C}$ and $\textsc{Dec} : \mathcal{C} \times \mathcal{K}_{\text{priv}} \to \mathcal{M} \cup \{\perp\}$, where $\perp$ is used to represent the invalid ciphertext tag, and such that $\textsc{Dec}_{\texttt{sk}}(\textsc{Enc}_{\texttt{pk}}(m)) = m$, for $(\texttt{sk}, \texttt{pk})$ a valid key pair. Namely, it is the output of algorithm KEYGEN. The scheme $\mathcal{E} = \{\textsc{KeyGen}, \textsc{Enc}, \textsc{Dec}\}$ is denominated ***asymmetric encryption scheme***.

In the same article, Diffie and Hellman proposed an algorithm to use Alice and

Bob key pairs to establish a secret key for conventional (symmetric) cryptography. Given a group G, such that $|G| = n$ and a group generator g. The algorithm KEYGEN randomly chooses $a \in [0, n)$, computes $A \equiv g^a \pmod{n}$ and returns $(\mathtt{sk}_A, \mathtt{pk}_A) = (a, A)$ to Alice. Analogously, Bob obtain the key pair (b, B), with $b \in [0, n)$ randomly chosen and $B \equiv g^b \mod n$. Alice uses Bob's public key, B and her own private key a to compute

$$B^a = (g^b)^a = g^{ab} \pmod{n}.$$

Similarly, Bob uses Alice's public key, A, and his own private key b to compute

$$A^b = (g^a)^b = g^{ab} \pmod{n}.$$

In this manner Alice and Bob compute the same value, that shall be used as secret key. Ironically, they suggested a new form of cryptography, without saying how to construct it, at the same time that they abstractly solved symmetric cryptography most important problem.

Two years later, in 1978, Rivest, Shamir and Adleman [RSA83] constructed a public key cryptosystem, named RSA, using a similar idea. Shortly, given $n = p.q$, where p and q are big prime numbers. The algorithm KEYGEN returns the pair (d, e), such that $d.e \equiv 1 \pmod{\phi(n)}$. The encryption algorithm computes $c = \text{ENC}_e(m) = m^e \pmod{n}$, while the decryption algorithm computes $\text{DEC}_d(c) = c^d \pmod{n}$. Correctness is guaranteed because $\text{DEC}_d(\text{ENC}_e(m)) = \text{DEC}_d(m^e \pmod{n}) = m^{e.d} \pmod{n} \equiv m \pmod{n}$.

In special, given two ciphertexts $c_1 = \text{ENC}_e(m_1)$ and $c_2 = \text{ENC}_e(m_2)$, we have that $c_1.c_2 = m_1^e.m_2^e = (m_1.m_2)^e \pmod{n}$. In general, given k ciphertexts $c_1, \ldots, c_k$, we have that $\prod c_i = \text{ENC}_e(\prod m_i)$. Thus, RSA preserves the structure of multiplication and a natural question that emerges is whether it is possible to obtain a scheme that preserves both multiplications and additions. Mathematically, such a map is called a ***ring homomorphism***.

Still in 1978, Rivest, Adleman and Dertouzos [RAD78] defined the concept of ***secret homomorphisms*** as being a mapping between algebraic systems, composed by operations, predicates and constants (preserved by the mapping). In other words, it is a cryptographic scheme $\mathcal{E} = \{\textsc{KeyGen}, \textsc{Enc}, \textsc{Dec}, \textsc{Eval}\}$, where the algorithm $\textsc{Eval}$ is able to evaluate algebraic circuits that belong to a permitted domain, denoted by $\mathbf{S_C}$, composed by additions and multiplications over ciphertexts. Namely, $\textsc{Eval} : \mathcal{K}_{\text{pub}} \times \mathbf{S_C} \times \mathcal{C}^k \to \mathcal{C}$, such that for each circuit $\mathbf{C} \in \mathbf{S_C}$, if $\overline{c} = \langle c_1, \ldots, c_n \rangle$ is a vector of ciphertexts such that $c_i = \textsc{Enc}_{\texttt{pk}}(m_i)$, then we have that $m = \mathbf{C}(m_1, \ldots, m_k)$ and $m = \textsc{Dec}_{\texttt{sk}}(\textsc{Eval}_{\texttt{pk}}(\mathbf{C}, c))$. The set of algorithms $\mathcal{E} = \{\textsc{KeyGen}, \textsc{Enc}, \textsc{Dec}, \textsc{Eval}\}$ is called ***fully homomorphic encryption (FHE)***, if $\mathbf{S_C}$ if equivalent to the set of all Boolean circuits. Formally it is necessary to establish conditions in order to the obtain a practical cryptosystem. For example, the ciphertext must not grow too much with respect to the size of the circuit that we want to evaluate. Furthermore, key generation, encryption, decryption and evaluation algorithms must have polynomial complexity with respect to the security parameter. In the same article, the authors proposed some concrete secret homomorphisms, but they were all proved to be insecure.

An important property to the construction of secret homomorphisms is known as ***semantic security***. If we know a set of plaintexts $M = \{m_1, m_2, \ldots, m_k\}$ and we wish to determine if a ciphertext c corresponds to some m_i and the encryption scheme is ***deterministic***, then it is sufficient to encrypt each m_i and compare the result with c. In order to have semantic security, a cryptographic scheme must be protected against such a trivial attack, and it implies that the encryption algorithm must be randomized, that is, each time the algorithm executes, it computes a different ciphertext (with high probability). RSA is a deterministic cryptosystem, then it not semantically secure. Thus, ***ElGamal*** is an immediate alternative, because it is not deterministic and also have multiplicative homomorphism like RSA. Specifically, given a big prime number p, a generator g of the multiplicative group $\mathbb{Z}_p$, Alice's secret key is a value a, randomly chosen between 0 and $p-1$. The public key is given by $A = g^a \pmod{p}$. Given the message $m \in \mathbb{Z}_p$, and a random number k, between 0 and $p-1$, we compute the ciphertext as $(c_1, c_2) = (g^k, A^k.m)$. To decrypt, Alice calculates $m = (c_1^a)^{-1}.c_2 \pmod{p}$. Given two ciphertexts, (c_1, c_2) and (c_1', c_2'), we define multiplication component-wisely, namely, $(c_1, c_2).(c_1', c_2') = (c_1.c_1', c_2.c_2')$. Hence, it is easy to see that ElGamal scheme is a homomorphism, because $(c_1.c_1', c_2.c_2') = (g^{k_1+k_1'}, g^{a(k_1+k_1')}.m)$. Indeed, this homomorphism maps multiplications to additions.

Interestingly, the first semantically secure homomorphic cryptosystem was proposed by Goldwasser and Micali in the same paper that defined the concept of semantic security [GM82]. They used the quadratic residuosity problem as the basic element of the construction. Let $N = p.q$, with p and q big prime numbers. Computing quadratic residues in $\mathbb{Z}_p$ or $\mathbb{Z}_q$ is easy, but computing it in $\mathbb{Z}_N$ is hard if we don't know the factorization of N. Therefore, private key is given by (p, q), the factorization of N, while the public key is given by (N, z), where z is an element of $\mathbb{Z}_N$ such that $z^{p-1/2} \equiv 1 \pmod{N}$ and z is not a quadratic residue in $\mathbb{Z}_N$. Given a message $m \in \{0, 1\}$, if $m = 0$, encryption algorithm returns a random quadratic residue in $\mathbb{Z}_N$, otherwise, if $m = 1$, it returns a non-quadratic residue c such that $c^{p-1/2} \equiv 1 \pmod{N}$. Decryption only can be done knowing the factorization of N, because we can compute separately the residues in $\mathbb{Z}_p$ and $\mathbb{Z}_q$, and use the Chinese remainder theorem to calculate corresponding quadratic residue in $\mathbb{Z}_N$. In special, given two quadratic residues, we know that its multiplication is again a quadratic residue. We also have that multiplication of elements c_1 and c_2, such that $c_i^{p-1/2} \equiv 1 \pmod{N}$, results in an element $c = \mathbb{Z}_N$, such that $c^{p-1/2} \equiv 1 \pmod{N}$ and if c_1 or c_2 is a non-quadratic residue, then c is also a non-quadratic residue. Therefore, the scheme is homomorphic with respect to multiplication and can be used as a secret homomorphism.

Particularly important is the cryptosystem ***Paillier***, whose security also is based (however there is no proof of equivalency) on the factorization of $N = p.q$. This scheme is constructed over the group $\mathbb{Z}_{N^2}^*$, which is isomorphic to $\mathbb{Z}_N \times \mathbb{Z}_N^*$. Indeed, the isomorphism is given by the relation $f : \mathbb{Z}_N \times \mathbb{Z}_N^*$, such that:

$$f(a,b) = (1+N)^a.b^N \pmod{N^2}.$$

The public key is given by the value N, while the private key is given by the pair (p,q). To encrypt the message $m \in \mathbb{Z}_N^*$, the value $c = (1+N)^m r^N \pmod{N^2}$ is computed, for a randomly chosen $r \in \mathbb{Z}_N^*$. On the other hand, the decryption algorithm computes

$$m = \frac{[c^{\phi(N)} \pmod{N^2}] - 1}{N}.\phi(N)^{-1} \pmod{N}.$$

Encryption is homomorphic with respect to addition, because

$$\begin{aligned}\textsc{Enc}_N(m_1).\textsc{Enc}_N(m_2) &= ((1+N)^{m_1} r_1^N).((1+N)^{m_2} r_2^N), \\ &= (1+N)^{m_1+m_2 \pmod{N}} (r_1 r_2)^N \pmod{N^2}.\end{aligned}$$

Paillier's cryptosystem homomorphism is distinct from others, not only because f has two components, but also because operation in each component is different. Namely,

$$f(a_1,b_1).f(a_2,b_2) = f(a_1+a_2, b_1.b_2).$$

Another cryptosystem that allows the construction of secret homomorphisms is called ***Polly Cracker***, proposed by Fellow and Koblitz [FK94], where the polynomial ring $R = \mathbb{F}_q[x_1,\ldots,x_n]$ contains the ideal I generated by a public set of polynomials, $\{p_1(x_1,\ldots,x_n),\ldots,p_k(x_1,\ldots,x_n)\}$, with a common root $\alpha = (\alpha_1,\ldots,\alpha_n)$, kept secret. Given a message $m \in \mathbb{F}_q$, the encryption algorithm computes the polynomial $c(\mathbf{x}) = \sum p_i(\mathbf{x}).r_i(\mathbf{x})$, where r_i are random polynomials, in order to obtain a random element from I. To decrypt, it is enough to evaluate the polynomial $c(\mathbf{x})$ in α. The security of Polly Cracker is an open problem, because although attacks were found, the scheme was adapted to resist against them.

The cryptosystem ***BGN*** [BGN05] is a practical scheme that allows the evaluation of quadratic formulas, in other words it allows the evaluation of circuits with just one level of multiplications and an arbitrary number of additions. Let $N = p.q$ and consider the groups $\mathbb{G}$ and $\mathbb{G}_1$, of order N, and a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. Given a generator $g \in \mathbb{G}$, compute $h = g^p$ and the public key is given by (N, h, g), while the private key is the factorization (p, q) of N. The plaintext space $\mathcal{M}$ is given by $\mathbb{Z}_p$ and the encryption algorithm computes $c = g^{m+kp}$, with a random k and $m \in \mathbb{Z}_p$. The decryption algorithm firstly computes $c^q \equiv g^{mq} \pmod N$ and later solves the discrete logarithm problem (DLP) with basis g^q. In order to solve the DLP efficiently, m must correspond to an element that belongs to a set with polynomial size, instead of any element from $\mathbb{Z}_p$. We have that a ciphertext multiplication corresponds to the addition of plaintexts. Moreover, given the ciphertexts $c_1 = g^{m_1+k_1p}$ and $c_2 = g^{m_2+k_2p}$, the bilinear pairing $e(c_1, c_2)$ is equal to $e(g, g)^{m_1m_2+dp}$, for an integer d, and corresponds to the multiplication of plaintexts.

The security of previous homomorphic constructions is related to complexity of the ***ideal membership problem***. In 2009 [Gen09b], Gentry utilized ideal lattices to construct the first FHE scheme, solving a problem that remained open for more than three decades. Because multiplications are too expensive (in a sense that we will make clear later) and also because of the size of the public key, such a proposal is not practical. However, optimizations were proposed in the following years [CMNT11, CNT11, SV09, BGV11, SS10], improving FHE performance. If FHE becomes a reality, new cryptographic primitives would emerge, providing a new kind of protection to cloud computing environment. Thus, we must understand what is the essencial contribution of Gentry's work. In his seminal paper [Gen09b], we can observe that the underlying problem that was used to construct the FHE scheme is related to a slightly different abstract problem when compared to the previous constructions. This abstraction is called ***ideal coset problem***.

Shortly, it is possible to describe Gentry's generic scheme as follows:

Key generation. The algorithm KeyGen chooses ideals J prime with I and generates a basis B_I for ideal I and two distinct basis for ideal J, namely B_J^{sk} and B_J^{pk}. Furthermore, the distribution $\mathcal{D}_{B_I}(m)$, that outputs random elements from coset

$m + I$, is determined.

Encryption. Given the message $m \in R \pmod{B_I}$, utilize the distribution $\mathcal{D}_{B_I}(m)$ to compute m' and then compute its reduction modulo B_J^{pk}, as follows

$$c = m' \pmod{B_J^{\mathrm{pk}}}.$$

Decryption. To decrypt, we compute the value

$$m = [c \pmod{B_{\mathrm{sk}}}] \pmod{B_I}.$$

Gentry used ideals over polynomial rings to construct a ***somewhat homomorphic scheme*** (SHE), in the sense that there is a limit in the number of operations that it can homomorphically evaluate. The proposal is able to evaluate circuits composed by many additions, but just a few number of multiplications. As operations are executed, a noise that is inserted during the encryption algorithm grows according to the homomorphic operations that are computed. Putting it differently, addition increases the noise linearly and multiplication increases it quadratically. The decryption algorithm works only if this noise do not exceed certain public threshold. Gentry used a concept he called ***bootstrapping*** to reduce such a noise homomorphically, by computing a special kind of circuit over a ciphertext whose noise is close to achieve this threshold. Unfortunately, this transformation is resource-consuming as we are going to see.

2.1 Gentry's blueprint

Next we show the common definitions and theorems, upon which the FHE scheme that we are going to present in this work is based.

Definition 2.1. ***Correctness.*** A scheme $\mathcal{E}(\textsc{KeyGen}, \textsc{Dec}, \textsc{Enc}, \textsc{Eval})$ is ***correct*** if, for a determined circuit $\mathbf{C}$ and every key pair $(\mathtt{sk}, \mathtt{pk})$, generated by $\textsc{KeyGen}$, any message tuple $(m_1, \ldots, m_t)$ and corresponding ciphertexts $\bar{c} = \langle c_1, \ldots, c_t \rangle$, that is, $c_i = \textsc{Enc}_{\mathtt{pk}}(m_i)$ for $1 \leq i \leq t$, then we have that

$$\textsc{Dec}_{\mathtt{sk}}(\textsc{Eval}_{\mathtt{pk}}(\mathbf{C}, \bar{c})) = \mathbf{C}(m_1, \ldots, m_t).$$

Furthermore, the algorithms $\textsc{KeyGen}$, $\textsc{Dec}$, $\textsc{Enc}$ and $\textsc{Eval}$ must have polynomial complexity.

Definition 2.2. ***Fully Homomorphic Encryption.*** A scheme $\mathcal{E}$ is ***correct for a class*** $\mathbf{S_C}$ of circuits, if it is correct for each $\mathbf{C} \in \mathbf{S_C}$. Moreover, $\mathcal{E}$ is denominated ***fully homomorphic*** if it is correct for every algebraic circuit, or, equivalently, if it is correct for every Boolean circuit.

Circuit privacy. We say that a scheme $\mathcal{E}$ has ***circuit privacy*** if the following functions are indistinguishable:

$$\textsc{Enc}_{\mathtt{pk}}(\mathbf{C}(m_1, \ldots, m_t)) \approx \textsc{Eval}_{\mathtt{pk}}(\mathbf{C}, \bar{c}).$$

Compact homomorphic encryption. A scheme $\mathcal{E}$ is ***compact*** if for every circuit $\mathbf{C}$, every ciphertext vector $\bar{c}$, considering any valid key pair (generated by $\textsc{KeyGen}$), then the ciphertext size generated by $\textsc{Eval}$ is polynomial in relation to the security parameter λ and independent from circuit size.

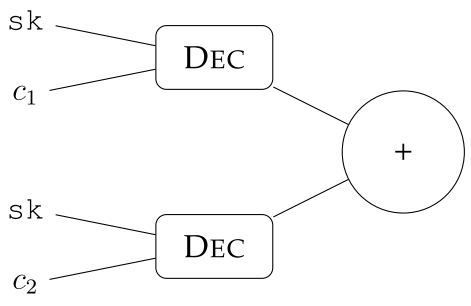


Figure 2.1: $D_{\mathcal{E}}^{+}$

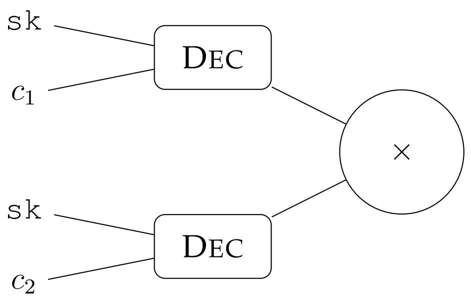


Figure 2.2: $D_{\mathcal{E}}^{\times}$

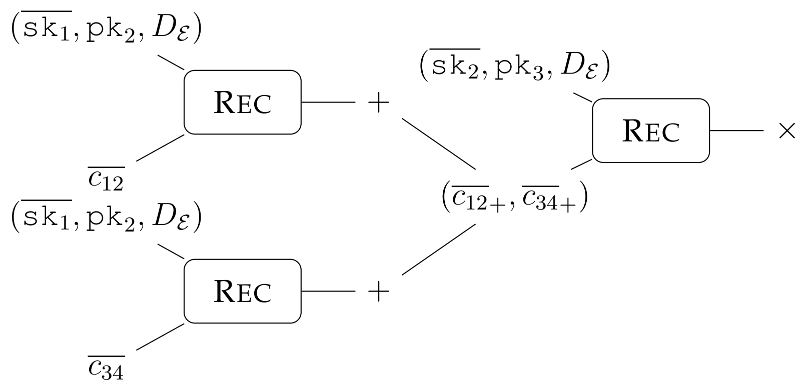
Definition 2.3. ***Augmented decryption circuit.*** Let $\mathcal{E}$ be a scheme such that decryption is implemented by a circuit that depends just on the security parameter λ. We define the ***augmented decryption circuit set*** as the set formed by two circuits, both receive as input the private key and two ciphertexts. The first circuit, $D_{\mathcal{E}}^{(+)}$, decrypts the ciphertexts of $\overline{c}$ and adds the results, while the second, $D_{\mathcal{E}}^{\times}$, decrypts its inputs and multiplies the results. Finally, we denote by $D_{\mathcal{E}}$ the decryption circuit by itself and by $\mathbf{S}_{\mathcal{D}}$ the set $\{D_{\mathcal{E}}^{+}, D_{\mathcal{E}}^{\times}\}$.

Bootstrappable encryption. Let $\mathcal{E}$ be a homomorphic encryption scheme. If $\mathbf{S}_{\mathbf{C}}$ represents the circuit set for which $\mathcal{E}$ is correct, and if $\mathbf{S}_{\mathcal{D}} \subseteq \mathbf{S}_{\mathbf{C}}$, then $\mathcal{E}$ is called ***bootstrappable***.

Theorem 2.1. ***Leveled homomorphic encryption.*** Let $\mathcal{E}$ be a scheme correct for the augmented decryption circuits, that is, $\mathcal{E}$ is ***bootstrappable***, then it is possible to construct a new scheme $\mathcal{E}^{(d)}$, correct, compact and homomorphic for all Boolean circuits of depth d. Moreover, $\mathcal{E}^{(d)}$ is semantically secure if $\mathcal{E}$ also is.

Proof can be found in Gentry's paper that describes the construction over ideal lattices [Gen09b].

This new scheme has the same $D_{\mathcal{E}}$ and has the same ciphertext length. The key pair consists of $d+1$ keys pairs from $\mathcal{E}$, together with the encryption of each bit of $\mathtt{sk}_i$ using $\mathtt{pk}_{i+1}$, what we are going to denote by $\overline{\mathtt{sk}_i}$. Namely, the private key is given by the vector $(\mathtt{sk}_1, \ldots, \mathtt{sk}_{d+1})$ and the public key is composed by



$(\overline{\mathtt{sk}_1}, \mathtt{pk}_2, D_{\mathcal{E}})$
REC
$+$
$\overline{c_{12}}$
$(\overline{\mathtt{sk}_1}, \mathtt{pk}_2, D_{\mathcal{E}})$
REC
$+$
$\overline{c_{34}}$
$(\overline{\mathtt{sk}_2}, \mathtt{pk}_3, D_{\mathcal{E}})$
REC
$\times$
$(\overline{c_{12+}}, \overline{c_{34+}})$

Figure 2.3: $\textsc{Enc}_{\mathtt{pk}_3}((m_1 + m_2) \times (m_3 + m_4))$

$$((\mathtt{pk}_1, \ldots, \mathtt{pk}_{d+1}), (\overline{\mathtt{sk}_1}, \ldots, \overline{\mathtt{sk}_d})).$$

Given a circuit **C** that we want to evaluate homomorphically, for each level i of the circuit C, ciphertexts are reencrypted, using $\mathtt{pk}_{i+1}$ and each addition or multiplication from the original circuit is substituted by an equivalent augmented decryption circuit. Thus, there is an algorithm, denoted by REC, that reencrypts the message replacing the public key $\mathtt{pk}_i$ by $\mathtt{pk}_{i+1}$, such that the message is always protected by at least one encryption level.

Algorithm 2.1 Reencryption

```
INPUT  $\mathtt{pk}_{i+1}$, $D_{\mathcal{E}}$, $\overline{\mathtt{sk}_i} = \textsc{Enc}_{\mathtt{pk}_{i+1}}(\mathtt{sk}_i)$ and the ciphertext tuple $\overline{c_i}$.
OUTPUT  A ciphertext tuple $\overline{c_{i+1}}$ computed using $\mathtt{pk}_{i+1}$.
  $\overline{c^\star_{i+1}} = \textsc{Enc}_{\mathtt{pk}_{i+1}}(\overline{c_i})$.
  $\overline{c_{i+1}} = \textsc{Eval}_{\mathtt{pk}_{i+1}}(D_{\mathcal{E}}, (\overline{\mathtt{sk}_i}, \overline{c^\star_{i+1}}))$.
  return $\overline{c_{i+1}}$.
```

Figure 2.3 shows an example of a depth 2 circuit, with 4 messages, m_1, m_2, m_3 and m_4, such that we can define the variables $\overline{c_{12}} = (\textsc{Enc}_{\mathtt{pk}_1}(m_1), \textsc{Enc}_{\mathtt{pk}_1}(m_2))$ and $\overline{c_{34}} = (\textsc{Enc}_{\mathtt{pk}_1}(m_3), \textsc{Enc}_{\mathtt{pk}_1}(m_4))$. Furthermore, we define $\overline{c_{12+}} = \textsc{Enc}_{\mathtt{pk}_2}(m_1 + m_2)$ and $\overline{c_{34+}} = \textsc{Enc}_{\mathtt{pk}_2}(m_3 + m_4)$. To simplify notation, we are going to use $\overline{\mathtt{sk}_i}$ to denote the vector composed by the encryption of each bit of $\mathtt{sk}_i$ using the public key $\mathtt{pk}_{i+1}$.

Definition 2.4. ***Circular security.*** Given a scheme $\mathcal{E}$, we say that $\mathcal{E}$ has circular security if it is secure to encrypt each private key bit using its own public key.

If a scheme $\mathcal{E}$ has circular security, we can use the same key pair all along the circuit homomorphic computation, and therefore it is not necessary to use $d + 1$ key pairs.

2.2 Security model

A cryptosystem is secure against ***chosen ciphertext attack*** (CCA2) if there is no polynomial time adversary that can win the following game with non negligible probability.

Setup. The challenger obtains $(\mathtt{sk}, \mathtt{pk}) = \textsc{KeyGen}(\lambda)$ and sends $\mathtt{pk}$ to the adversary $\mathcal{A}$.

Queries. $\mathcal{A}$ sends ciphertexts to the challenger, before or after the challenge, who returns the corresponding plaintexts.

Challenge. The adversary randomly generates two plaintexts $m_0, m_1 \in \mathcal{M}$ and sends to the challenger, that chooses randomly a bit $b \in \{0, 1\}$ and computes the ciphertext $c = \textsc{Enc}_{\mathtt{pk}}(m_b)$. The challenger sends c to $\mathcal{A}$.

Answer. $\mathcal{A}$ sends a bit b' to the challenger and wins the game if $b' = b$.

If we allow queries only before the challenge, we say that the cryptosystem is secure against CCA1 adversaries (lunchtime attacks). As previously described, queries can be interpreted as an access to a ***decryption oracle***. If instead we only allow access to an ***encryption oracle***, i. e. the adversary can choose any message to be encrypted under the same key pair, then we say that the cryptosystem is secure against ***chosen plaintext attacks*** (CPA).

In homomorphic encryption, it is impossible to achieve CCA2 security, because the adversary can add an encryption of zero to the encrypted message, or multiply it by the encryption of one, and send it back to the decryption oracle. Many FHE schemes have as public value an encryption of the private key bits, which can be sent to the decryption oracle before the challenge, what make such schemes insecure against CCA1 adversaries. Indeed, a ***key recovery*** attack is stronger than an CCA1 attack and Loftus et al [LMSV11] showed that Gentry's construction over ideal lattices is vulnerable to it and presented the only somewhat homomorphic encryption proposal that is known to be CCA1 secure. From now on, when we say that an scheme is ***semantically secure***, we mean CPA security.

3 Construction over the integers

In this section we will describe the construction of fully homomorphic encryption over the integers. This version is simpler than the lattice-based version and we are going to start viewing how to construct the framework described in last section over the ring of integers $\mathbb{Z}$. The extension to ideal lattices has many similarities with the integer version, then it will be simpler, in terms of mathematical abstraction, to describe the integer version.

3.1 Symmetric somewhat homomorphic scheme

Definition 3.1. Let λ be the security parameter. The algorithm KEYGEN randomly generates the private key as an odd integer p with bit-length λ^2. To encrypt a bit m, the algorithm ENC randomly chooses r with λ bits and r is called the ***noise***. An integer q with λ^5 bits is randomly chosen in order to compute the ciphertext:

$$c = m + 2r + pq.$$

The decryption algorithm computes the bit $m = \text{DEC}_p(c) = [c]_p \pmod 2$. It is easy to see that the encryption is homomorphic with respect to addition and also multiplication. However, the decryption only works if $|m + 2r|$ is less than or equal to $p/2$, because modular reduction does not change m's parity.

Given the ciphertexts $c_1, c_2, \ldots, c_k$, such that $c_i = m_i + 2r_i + pq_i$ and $m_i + 2r_i \ll p$, finding p is a problem called ***approximate greatest common divisor (AGCD)***, studied by Howgrave-Graham in the context of cryptanalysis [HG01]. The sizes of q_i, p and r_i are chosen to resist against attacks described in the literature.

3.2 Asymmetric somewhat homomorphic scheme

In order to obtain an asymmetric scheme we can use encryptions of zero as the public key, since they can be used to randomize a ciphertext by homomorphically adding to it random elements from the public key.

The construction utilizes the following parameters, which are polynomial with respect to the security parameter λ:

- γ is the bit-length of ciphertexts. This parameter must satisfy $\gamma = \omega(\eta^2 \log \lambda)$, in order to avoid attacks against approximate GCD problem [HG01,CH11,Lag85, NS01];

- η is the bit-length of private key p, respecting the inequality $\eta \geq \rho\Theta(\lambda \log^2 \lambda)$, because this is enough to allow the homomorphic evaluation of the augmented decryption circuit;

- ρ is the bit-length of the noise r_i. This parameter must be chosen such that the scheme resists against brute force attack in the noise [CN12];

- τ is the number of approximate multiples of p in the public key, chosen according to the relation $\tau \geq \gamma + \omega(\log \lambda)$, allowing us to use the leftover hash lemma to show that the public key looks like random values with respect to the private key, and therefore we can show that the scheme is secure;

- $\rho' = \rho + \omega(\log \lambda)$ is an auxiliary parameter used in the decryption algorithm.

Halevi [vDGHV10] suggests the following parameter instantiation: $\rho = \lambda$, $\rho' = 2\lambda$, $\eta = \tilde{O}(\lambda^2)$, $\gamma = \tilde{O}(\lambda^5)$ and $\tau = \gamma + \lambda$. Considering such a choice, we can define the following distribution:

$$\mathcal{D}_{\gamma,\rho}(p) = \{pq + r \mid q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)\}.$$

Definition 3.2. **Key Generation.** Obtain a random odd integer p with η bits. For $0 \leq i \leq \tau$, compute $x_i = \mathcal{D}_{\gamma,\rho}(p)$. Rename the indexes such that x_0 corresponds to the largest element. Repeat until x_0 is odd and $x_0 \pmod p$ is even. The public key is given by $\texttt{pk} = (x_0, \ldots, x_\tau)$ and the private key is given by $\texttt{sk} = p$.

Encryption. Choose randomly a subset $S \subset \{0, 1, \ldots, \tau\}$ and a random integer r in the interval $(-2^{\rho'}, 2^{\rho'})$ and compute $c = [m + 2r + \sum_{i \in S} x_i]_{x_0}$.

Decryption. Return $m = [c]_p \pmod 2$.

Evaluation. Given the circuit C and t ciphertexts, execute the operations of C modulo x_0 over the ciphertexts, that are given by big integers, and return the result.

It is important to remark that the fact that p is odd allows us to decrypt as follows:

$$m' = [c - \lfloor c/p \rceil]_2 = (c \pmod 2) \oplus (\lfloor c/p \rceil \pmod 2).$$

3.3 Correctness

Now we are going to show the correctness of the scheme from definition 3.2. However, before that, we will give some important definitions, as for example the ***generalized*** circuit and ***permitted circuit***, that will make our work easier later.

Definition 3.3. ***Generalized circuit.*** Considering a circuit $\mathbf{C}$ whose gates are given by additions and multiplications modulo 2, the generalized circuit $g(\mathbf{C})$ is formed by equivalent operations, but instead of computing over bits it computes over larger integers.

Definition 3.4. ***Permitted circuits.*** Considering a circuit $\mathbf{C}$ and the generalized circuit $g(\mathbf{C})$, we define the ***permitted circuits class***, denoted by $\mathbf{S_C}$, as being composed by the circuits whose inputs have absolute value at most $2^{\alpha(\rho'+2)}$ and whose output has absolute value at most $2^{\alpha(\eta-4)}$.

In fact, definition 3.4 is technically important because it allows us to prove security in a straightforward manner, because the parameter can be chosen in order to satisfy the constraints. Indeed, a fresh ciphertext has noise whose length is at most $2^{\rho'+2}$, while the decryption algorithm expects noise limited above by $p/2$. Therefore, regarding an appropriate choice of parameters, η ensures that the scheme can evaluate its own decryption circuit, because it forces the augmented decryption circuit set to be contained in the permitted circuit class. But as we will see later, correctness and security establishes a trade-off with minimum requirements for both sides, what makes constants and polynomial complexity degrees high enough to turn FHE impractical.

Nevertheless, this definition doesn't captures directly what kind of circuit can be homomorphically evaluated. In order to understand that, we need to comprehend how addition and multiplication influences the noise growth. But this question has an easier solution when we work with integers. In special, multiplication noise is squared for each circuit level, while addition maintains just a linear growth. Thus, we can interpret the circuit operations as a multivariate polynomial over the integers, and the its degree d is related to the circuit ***multiplicative depth***, which is given by $\log_2 d$.

Lemma 3.1. Given a circuit $\mathbf{C}$ and the corresponding generalized circuit $g(\mathbf{C})$, we can construct the polynomial $f(x_1, \ldots, x_t)$, that is equivalent to the circuit $\mathbf{C}$. Let d be the degree of f, then if we have that $|f|(2^{\rho'+2})^d \leq 2^{\eta-4}$, where $|f|$ represents the a summation of coefficients of f, we have that $\mathbf{C}$ is a permitted circuit, i. e. $\mathbf{C} \in \mathbf{S_C}$.

Proof. According to definition 3.2, we have that $c = [m + 2r + \sum_{i \in S} x_i]_{x_0}$. Furthermore, x_0 is the maximum among x_i, for $0 \leq i \leq \tau$, then there exists an integer k, such that $|k| < \tau$, satisfying the following equation

$$c = (m + 2r + \sum_{i \in S} x_i) + kx_0.$$

By definition of x_i, we have that $x_i = q_i p + 2r_i$, for $|r_i| \leq 2^\rho$. Hence, we have that

$$c = k(q_0 p + 2r_0) + (m + 2r + \sum_{i \in S}(q_i p + 2r_i)),$$

$$c = p(kq_0 + \sum q_i) + (m + 2r + 2kr_0 + \sum_{i \in S} 2r_i),$$

$$c = p(kq_0 + \sum q_i) + (m + 2(r + kr_0 + \sum_{i \in S} r_i)).$$

Considering that $\rho' \geq 2\rho$, $\tau \leq 2^\rho$ and $\tau = \lambda^5 + \lambda \leq 2^\lambda$ ($\lambda > 23$ is enough to ensure this condition), the rightmost term has absolute value at most,

$$\begin{aligned}|1+2(2^{\rho'+1}+\tau 2^{\rho+1}+\tau 2^{\rho+1})| &\leq |1+2(2^{\rho'+1}+\tau 2^{\rho+2})|,\\ &\leq |1+2^{\rho'+2}+\tau 2^{\rho+3}|,\\ &\leq |2^{\rho'+3}|.\end{aligned}$$

Therefore, the scheme can evaluate polynomials whose degree respects the following inequality:

$$d \leq \frac{\eta - 4 - \log|f|}{\rho' + 2}.\square$$

Polynomials satisfying this condition are called ***permitted polynomials***.

Lemma 3.2. Let $(\mathtt{sk}, \mathtt{pk})$ be a key pair generated by KeyGen. Let $c = \text{Enc}_{\mathtt{pk}}(m)$, with $m \in \{0, 1\}$. Then we have that $c \pmod p$ is of the form $2a+m$, that is, $c \pmod p$ has the same parity of m. Moreover, $|2a+m| < 2^{\rho'+2}$.

Proof. According to definition 3.2, we have that $c = [m + 2r + \sum_{i \in S} x_i]_{x_0}$. In addition, as before we have that x_0 maximum among x_i, for $0 \leq i \leq \tau$, then modular reduction of each x_i by x_0 results in a negative integer. If we take the absolute value we have that $2r$ by definition is at most $2^{\rho'+2}$. $\square$

Lemma 3.3. Considering a permitted circuit $\mathbf{C}$, the output of $\text{Eval}_{\mathtt{pk}}(\mathbf{C}, c_1, \ldots, c_t)$, where c_i are valid ciphertexts, is a new ciphertext with noise at most $p/8$.

Proof. The noise of $\text{Eval}_{\mathtt{pk}}(\mathbf{C}, c_1, \ldots, c_t)$ is given by the evaluation of $\mathbf{C}$ over c_i individual noises, that is, we can separate the evaluation of circuit C in two steps, because multiples of p can be combined into a unique multiple p, while isolated evaluation of noises results in the final noise. We have that each c_i noise is limited to $2^{\rho'+2}$ according lemma 3.2, then by definition of permitted circuit, we have that the final noise is at most $2^{\eta-4} = p/8$. $\square$

Unfortunately, the scheme $\mathcal{E}$ have not decryption circuit depth sufficiently short as required and some adjustments will be necessary to make it possible.

3.4 Security

The details of the security proof for the scheme from last section can be found in the paper [vDGHV10], where the authors show that the existence of an attack to the proposed cryptosystem leads to a solution to the approximate GCD (AGCD) problem. Shortly, the problem consists in finding a common divisor to a set of approximate multiples of this divisor. Supposing the existence of an algorithm to compute one bit from the plaintext, given just the ciphertext and public parameters, we can use binary GCD algorithm to construct a solution to the AGCD problem.

Now we can explain better why the parameter ρ' was required. Basically, choosing a noise with $\rho' = 2\rho$ bits, we have that the ciphertext is protected by ***high noise*** ρ', while the public key contains zero encryptions, achieved using ***low noise*** ρ. This distinction is important to the security proof, because we can reduce the security of the cryptosystem with high noise to the GCD problem over low noise.

Definition 3.5. Given parameters (ρ, η, γ) and a polynomial number of elements from the distribution $\mathcal{D}_{\gamma,\rho}(p)$, for a randomly chosen odd integer p, the ***AGCD problem***, consists in revealing p.

Theorem 3.1. The parameter choice described in definition 3.2 and the security parameter λ, any attack $\mathcal{A}$ with advantage ϵ over the scheme $\mathcal{E}$ can be transformed into an algorithm $\mathcal{B}$ to solve the approximate GCD problem with advatage at least $\epsilon/2$. The complexity of $\mathcal{B}$ is polynomial over the complexity of $\mathcal{A}$, λ and $1/\epsilon$.

3.5 Attacks

The first strategy one could imagine to solve the AGCD problem is using brute force to find the noise of an arbitrary pair of samples. Considering that the noise has ρ bits, we must find two arbitrary noises r_1 and r_2 from samples $x_1 = q_1p + r_1$ and $x_2 = q_2p + r_2$, respectively, compute the GCD or $x_1 - r_1$ and $x_2 - r_2$ and verify if the result has η bits. The complexity of this algorithm is $2^{2\rho}$.

Another immediate attack would be factoring noise-free term $x_0 = pq$ to find p. The best algorithm available to solve this problem is Lenstra's factoring algorithm [Len87], which has complexity $O(\sqrt{\eta})$. Therefore, choosing $\eta = \lambda^2$ we avoid this kind of attack.

Lagarias attacks. The AGCD problem can be interpreted as the ***simultaneous diophantine equations*** problem and we can use the Lagarias algorithm to solve it. Shortly, the idea of the attack is very simple. We must use samples contained in the public key, namely x_i, for $0 \leq i \leq t$, to compute the lattice given by the following matrix:

$$M = \begin{pmatrix} 2^{\rho} & x_1 & x_2 & \dots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & -x_0 & \\ & & & & -x_0 \end{pmatrix} \tag{1}$$

Then we reduce the lattice basis using for example the LLL algorithm as described in the Magma code in Table 1. If the reduction is good enough we get a vector $v = \langle q_0 2^{\rho}, q_0 x_1 - q_1 x_0, \dots \rangle$. Thus we compute the quotient q_0, from which it is possible to obtain the private key p even in the case that x_0 has a non-zero noise r_0.

Other two attacks are important to be considered. For instance, the ***Cohn-Heninger attack*** and the ***orthogonal lattice attack***. Both are described in detail in Lepoint PhD thesis [Lep14], where a Sage routine is presented, such that it is possible to derive parameters using estimations for the number of operations required to run each attack and thus obtaining the security level of the cryptosystem by making this attacks run in at least 2^{λ} operations.

```
lagarias := function(X, X0, rho)
    Z = IntegerRing();
    t := #X;
    XX := Matrix( Z, t, 1, [x : x in X] );
    M := ZeroMatrix( Z, t, t );
    M[1,1] := rho;
    for i:=1 to t-1 do
        M[1,i+1] := XX[i,1];
        M[i+1,i+1] := -X0;
    end for;
    L := LLL(M);
    q0 := (EuclideanNorm(L[1,1]))/(rho);
    if (q0 ne 0) then
        r0 := (X0 mod q0);
        p := (X0-r0) div q0;
        return p;
    end if;
    return -1;
end function;
```

Table 1: Lagarias attack

3.6 Squashing the decryption circuit

In this section we present modifications that are essential to construct a bootstrappable encryption scheme, because to construct a fully homomorphic encryption scheme we must be able to evaluate its own decryption circuit. This condition is satisfied if decryption has multiplicative depth sufficiently low. The decryption is computed using the equation $m = [c - [c/p]]_2$ and division by p can not be computed using a sufficiently shallow circuit. To solve this problem we can add information to the ciphertext, such that the decryption algorithm has its job facilitated, but without losing in security. Next we are going to present a scheme that is able to evaluate its own decryption circuit.

Definition 3.6. This construction uses three new parameters: $\kappa = \gamma\eta/\rho'$, $\theta = \lambda$ and $\Theta = \omega(\kappa \log \lambda)$, that is, they are all polynomial with respect to the security parameter λ.

Key Generation. Compute `sk` and `pk` as in definition 3.2. Compute $x_p = \lfloor 2^\kappa/p \rceil$, choose randomly a vector $s = \langle s_1, \ldots, s_\Theta \rangle$, with Θ bits and ***Hamming weight*** θ. The set S is defined as

$$S = \{i \mid s_i = 1\}.$$

Choose randomly integers u_i, where $1 \leq i \leq \Theta$, with at most κ bits, such that $\sum_{i \in S} u_i = x_p \pmod{2^{\kappa+1}}$. Compute $y_i = u_i/2^\kappa$, such that each y_i is a positive integer less than or equal to 2, with κ bits of precision after the fractional part. Thus, we have that $[\sum_{i \in S} y_i]_2 = (1/p) - \Delta_p$, for $\Delta_p < 2^{-\kappa}$.

The private key is given by the vector $(s_1, \ldots, s_\kappa)$ and the public key is given by `pk` and the vector $(y_1, \ldots, y_\Theta)$.

Encryption. Compute c as in the original scheme. For $1 \leq i \leq \Theta$, compute $z_i = [cy_i]_2$, maintaining just $\lceil \log \theta \rceil + 3$ of precision bits for each z_i. Return c and the vector $(z_1, \ldots, z_\Theta)$.

Decryption. Return $m' = [c - \lfloor \sum_{i \in S} s_i z_i \rceil]_2$.

Evaluation. Addition and multiplication are computed using canonical operations over the rationals.

Lemma 3.4. The modified scheme is correct for permitted circuits $C \in \mathbf{S_C}$. Moreover, given an ciphertext $(z_1, \ldots, z_\Theta)$, generated by the evaluation of any permitted circuit, we have that $s_i z_i - \lfloor \sum s_i z_i \rceil \leq 1/4$.

Proof. Given that the public key contains the vector $(y_1, \ldots, y_\Theta)$, we know that the values of y_i were chosen respecting $[\sum s_i y_i]_2 = 1/p + \Delta_p$, with $\Delta_p \leq 2^{-\kappa}$.

Given a permitted circuit C, such that $c^* = \textsc{Eval}_{\mathrm{pk}}(C, c_1, \ldots, c_t)$, for valid ciphertexts c_i, we have that $[c^* y_i]_2 = z_i - \Delta_i$, with $\Delta_i \leq 1/16\theta$, because just $\lceil \log \theta \rceil + 3$ of precision is maintained in relation to z_i. Hence, we have that

$$\begin{aligned}
[(c^*/p) - \sum s_i z_i]_2 &= [(c^*/p) - \sum s_i [c^* y_i]_2 + \sum s_i \Delta_i]_2, \\
&= [(c^*/p) - c^* [\sum s_i y_i]_2 + \sum s_i \Delta_i]_2, \\
&= [(c^*/p) - c^*(1/p - \Delta_p) + \sum s_i \Delta_i]_2, \\
&= [(c^* \Delta_p + \sum s_i \Delta_i]_2.
\end{aligned}$$

Considering the last term, we have that $|c^* \Delta_p| \leq 1/16$, because c^* is an ciphertext computed by the evaluation algorithm, whose input is formed by ciphertexts of length at most $2^{\alpha(\rho'+2)}$. Thus, the algorithm EVAL returns a value with legth at most $2^{\alpha(\eta-4)}$. In particular, ciphertext sizes are limited above by 2^γ, then we have that c^* has magnitude of at most $2^\gamma(\eta-4)/(\rho'+2) < 2^{\kappa-4}$. Hence, as $\Delta_p < 2^{-\kappa}$, we have that $|c^* \Delta_p| < 1/16$. With respect to $|\sum s_i \Delta_i|$, as $|\Delta_i| < 1/16\theta$ and there is θ values of i for which $i \in S$, then we have that $|\sum s_i \Delta_i| < 1/16$. Therefore, we have that

$$\left| \left[c^* \Delta_p + \sum s_i \Delta_i \right]_2 \right| < 1/8. \square$$

3.7 Bootstrappability

In the last section, we constructed an asymmetric scheme that has better decryption circuit, in the sense that it has shorter multiplicative depth when compared to the symmetric scheme. Now we can prove that this scheme is bootstrappable, completing the construction of a fully homomorphic encryption scheme over the integers.

Theorem 3.2. Definition 3.6 gives us an scheme whose augmented decryption circuit set, denoted by $\mathbf{S}_{\mathcal{D}}$, is an element of the set formed by the permitted circuits, i. e. $\mathbf{S}_{\mathcal{D}} \in \mathbf{S}_{\mathbf{C}}$.

Proof. The goal is to find an adequate circuit to compute the following equation:

$$m = c - \left\lfloor \sum s_i z_i \right\rceil \pmod 2.$$

We can define a new variable $a_i = s_i . z_i$, with $1 \leq i \leq \Theta$. Thus, $a_i = z_i$ when $s_i = 1$ and $a_i = 0$ when $s_i = 0$. By definition, a_i has $n = \lceil \log \theta \rceil + 3$ bits of precision and

$a_{1,0},$	$a_{1,-1}$	$a_{1,-2}$	$\ldots$	$a_{1,-n}$
$a_{2,0},$	$a_{2,-1}$	$a_{2,-2}$	$\ldots$	$a_{2,-n}$
$a_{3,0},$	$a_{3,-1}$	$a_{3,-2}$	$\ldots$	$a_{3,-n}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$a_{\theta,0},$	$a_{\theta,-1}$	$a_{\theta,-2}$	$\ldots$	$a_{\theta,-n}$
W_0	W_{-1}	W_{-2}	$\ldots$	W_{-n}

Table 2: Achieving bootstrappability

there is θ non-zero values among a_i. The last statement is crucial to construct this circuit, because it allows us to reduce the number of variables we are dealing with. This reduction is achieved by finding $n + 1 = \lceil \log \theta \rceil + 4$ rational numbers w_j, such that $\sum w_j = \sum a_i \pmod 2$.

Each a_i is a rational number between 0 and 2. Hence, the binary representation of a_i can be expressed as follows:

$$a_i = a_{i,0}, a_{i,-1} a_{i,-2} \ldots a_{i,-n}.$$

Negative indexes are used to reinforce the fact that these bits represent binary expansion of a rational number, that is,

$$a_i = 2^{-j} \sum_{j=0}^{n} a_{i,-j}.$$

Before we can compute w_j, we are going to define W_{-j} as being the ***Hamming weight*** of the vector $\{a_{i,j}\}_{i=1}^{\theta}$, as shown in Table 2.

There is at most θ non-zero values of a_i, then the value of W_{-j} is at most θ and defining $w_j = 2^{-j} W_{-j} \pmod 2$, we have that w_j can be represented by $\lceil \log \theta \rceil + 1$ bits of precision.

Lemma 3.5. Considering the sequence of bits $\overrightarrow{b} = (b_1, \ldots, b_k)$, the ***Hamming weight*** of $\overrightarrow{b}$, denoted by $H_{\overrightarrow{b}}$ can be computed by looking at each bit b_i. If the binary representation of $H_{\overrightarrow{b}}$ is given by $(h_n, \ldots, h_0)$, such that $H_{\overrightarrow{b}} = \sum 2^i h_i$, then h_i can be writen as a degree 2^i polynomial on the variables $\{b_i\}_1^k$. Furthermore, there is a circuit of size $k2^i$ that computes all the values of h_i simultaneously.

Proof. The i-th bit of $H_{\overrightarrow{b}}$ can be computed by δ_{2^i}, where δ_i represents the i-th ***elementary symmetric polynomial***. The degree of δ_{2^i} is exactly 2^i and to calculate simultaneously all the values of h_i it is enough to compute the polynomial $p(z) = \prod(z - b_i)$, because h_i corresponds to the coefficient of the term z^{k-i} in $p(z)$. The following algorithm computes the bits $h_0, \ldots, h_n$ and can easily be transformed into a circuit:

Algorithm 3.1 Elementary symmetric polynomials

```
INPUT $b_1, \ldots, b_k$.
OUTPUT $\delta_1(b_1, \ldots, b_k), \ldots, \delta_n(b_1, \ldots, b_k)$.
  Initialize $e_{0,0} = 1$ and $e_{i,0} = 0$ for $i = 1, 2, 3, \ldots, 2^n$.
  for $j = 1, 2, \ldots, k$ do
    for $i = 2^\ell, 2^{\ell-1}, \ldots, 1$ do
      Compute $e_{i,j} b_j e_{i-1,j-1} + e_{i,j-1}$ (polynomial arithmetic).
  return $e_{1,k}, \ldots, e_{2^n,k}$.
```

Polynomial multiplications are computed using the Fast Fourier Transform (FFT).

□

3.8 Security of the new scheme

In order to show that the presented construction is secure it is necessary to ensure that the information included in the public key, $(y_1, \ldots, y_\theta)$, can not be used to reconstruct the private key. This problem was studied by Gentry in 2009 [Gen09a] and is known as ***sparse subset sum problem*** (SSSP). If we choose θ sufficiently big to avoid brute force attack this problem is hard to solve. Moreover, it is necessary to have Θ greater than $\omega(\kappa \log \lambda)$, where κ is the bit-length of the numbers included in the public key.

3.9 Further work

Some optimizations were proposed to improve DGHV performance [CNT12,CNT11, CMNT11]. In 2013, Kim et al [KLYC13] used the ***Chinese remainder theorem*** (CRT) to show how to construct a CRT-based fully homomorphic encryption, allowing to encode more information inside each ciphertext. Recently, Cheon and Stehlé [CS15a] showed that the AGCD problem can be reduced to the LWE problem and vice-versa, demonstrating that both problems are in some sense equivalent.

It is possible to construct FHE using the LWE problem following roughly the same steps describe in this section. Such kind of construction is more efficient than AGCD-based schemes, for reasons that we will expose in next section. However, we are not going to give the details of how to achieve bootstrapability, since we will be concerned in the construction of practical cryptosystems. But it is important to remark that LWE-based FHE schemes corresponds to the ***state-of-the-art*** and indeed it is possible to bootstrap in less than 1 second using new techniques proposed by Ducas and Micciancio [DM15].

4 Construction based on the LWE problem

Untill now, our goal was to obtain fully homomorphic encryption, i. e. an scheme that allows an ***arbitrary*** number of additions and multiplications over encrypted data, and for that reason the plaintext size was restricted to the binary set $\{0, 1\}$. On the other hand the BGV proposal allows to compute for example using plaintext equal to $\{0, 1\}^{32}$, what is relevant if you want to use homomorphic encryption in practice. Hence, we will be interested in how to choose parameters to allow multiplicate depth $L \leq 30$. Such an scheme is called ***somewhat homomomorphic encryption*** (SHE).

In 2011, Brakerski and Vaikuntanathan [BV11a] proposed two new ideas to help the noise management for homomorphic schemes: ***dimension reduction*** (also known as ***relinearization*** or ***key switching***) and ***modulus reduction***. These new techniques provided a better alternative than the one used to squash the decryption circuit in sections 3.6, namely the utilization of SSSP problem.

These concepts are fundamental to the construction proposed by Brakerski, Gentry and Vaikuntanathan, called ***BGV*** [BGV11], which is a construction with better performance in practice, as we are going to show. Previous constructions have a worse method to deal with multiplications, because the noise grows too fast. Indeed, BGV's noise management avoids the exponential growth inherent in other proposals.

The security of LWE-based cryptosystems follows originally from a quantum reduction to $\textsc{Gap}\mathbf{SVP}_\gamma$ in the worst case [Reg05a], where γ is a polynomial function on the LWE parameters. A classical reduction was shown by Lindner and Peikert [LP11], but imposing a condition on the size of the modulus, namely an exponential modulus, which was shown to be not necessary by Brakerski et al in 2013 [BLP$^+$13].

Next we are going to define the LWE problem and present the BGV scheme, which can be easily implemented using a library like for example NTL [Sho], for number theoretic calculations, over GMP [pro], for efficient arbitrary precision arithmetic.

4.1 LWE problem

Definition 4.1. The ***LWE problem*** consists in finding the vector $s \in \mathbb{Z}_q^n$, given the equations

$$\begin{aligned} \langle s, a_1 \rangle &\approx_{\mathcal{D}} b_1 \pmod q \\ \langle s, a_2 \rangle &\approx_{\mathcal{D}} b_2 \pmod q \\ &\vdots \end{aligned}$$

Notation $\approx_{\mathcal{D}}$ means a tolerance in the equality, according to a distribution $\mathcal{D}_{n,\sigma}$. Namely, $\langle s, a_i \rangle$ has a distance to b_i and this distance is determined by the distribution $\mathcal{D}_{n,\sigma}$, generally an n-dimensional Gaussian distribution with standard deviation given by σ. Alternatively, we can write $\langle s, a_i \rangle = b_i + e_i \pmod q$, where $e_i \in \mathcal{D}_{n,\sigma}$.

Gaussian distribution plays a central role, because Micciancio and Regev [MR07] presented in 2004 a concept called ***smoothing parameter***. This parameter permits a different way to obtain pseudorandomness from lattices. Figure 4.4 shows a simplification of the idea. It shows centered Gaussians, with increasing standard deviation, reduced modulo the trivial one-dimensional lattice $\mathcal{L} = \mathbb{Z}$.

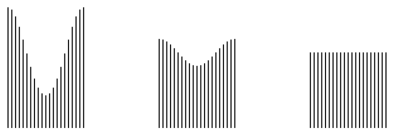


Figure 4.4: Gaussian distributions modulo 1

The number of equations do not contribute a lot to the solution. There is a trade-off between the number of equations and the execution time to find a solution to the problem, even with an arbitrary number of equations the complexity is at least subexponential [BKW03].

In 2005, Regev [Reg05b] presented a quantum reduction from LWE problem to worst case problems over lattices. Moreover, this work provided what was necessary to construct a new cryptosystem, whose performance is considerable better than other schemes based on lattices. Lindner and Peikert [LP11] showed a classical reduction, proposing the parameter analysis that are going to be adopted here.

Lyubaskevsky, Peikert and Regev defined a similar version to the LWE problem, but using polynomial rings [LPR10]. This construction will be denoted by ***ring LWE***. Concretely, let $f(x) = x^n + 1$, where n is a power of 2. Given an integer q and an element $s \in R = \mathbb{Z}_q[x]/f(x)$, the **ring LWE problem** over R, with respect to the distribution $\mathcal{D}_{n,\sigma}$, is defined equivalently, that is, it is necessary to find s that satisfies the following equations:

$$\begin{aligned} s.a_1 &\approx_{\mathcal{D}} b_1 \pmod{R_q} \\ s.a_2 &\approx_{\mathcal{D}} b_2 \pmod{R_q} \\ &\vdots \end{aligned}$$

where a_i and b_i are elements of the ring R_q. Modular reduction in R_q is the same as reducing modulo $f(x)$ and reducing the coefficients modulo q.

4.2 Somewhat homomorphic encryption

Let $\mathcal{D}_{n,\sigma,q}$ be a n-dimensional spherical discrete Gaussian distribution [LPR13,LPR10] with standard deviation σ and over the ring $\mathbb{Z}_q$.

Definition 4.2. Given the security parameter λ and a secondary parameter μ, we choose an integer q with μ bits and $N = \lceil 3 \log q \rceil$. The scheme $\mathcal{E}$ parameters are given by $\lambda, \mu, n, \sigma, q$ and the ring $R = \mathbb{Z}_q[x]/f(x)$ and $f(x)$ a cyclotomic polynomial.

Key Generation. Utilize the distribution $\mathcal{D}_{n,\sigma,q}$ to compute the polynomial $s^\star \in R$. Denote by s the vector formed by the polynomials 1 and $s^\star$. The private key is given by $\mathtt{sk} = s$. Generate randomly a matrix A' with N rows and one column, whose elements are polynomials with coefficients uniformly chosen in $\mathbb{Z}_q$. Utilize the distribution $\mathcal{D}_{n,\sigma,q}$ to generate N polynomials e_i and compute $b = A's^\star + te$. Compute the matrix A of two columns, the first one equal to b and the second one equal to $-A'$. The public key is given by $\mathtt{pk} = A$. By construction, we have that $As = te$.

Encryption. Given a message $m \in \{0,1\}^t$ and the public key $\mathtt{pk}$, we define the matrix m' with two rows, where the first one is the value m and the second one is equal to zero. Generate randomly a column matrix denoted by r with N rows composed by binary polynomials. Finally, output the ciphertext

$$c = \textsc{Enc}_{\mathtt{pk}}(m) = m' + A^T r \pmod{q}.$$

Decryption. Compute

$$m = \textsc{Dec}_{\mathtt{sk}}(c) = [[\langle c, \mathtt{sk} \rangle]_q]_t.$$

The correctness of this scheme is easily verified using the relation $As = te$ and the fact that q is chosen sufficiently big such that the error do not get above $q/4$, similarly to the case over the integers. Also, ciphertext addition is executed component-wisely, while ciphertext multiplication is done by computing the tensor technique, which will be explained in Section 4.6.

4.3 LWE security

Gentry, Halevi and Smart [GHS12b] used the BGV scheme to homomorphically evaluate the AES symmetric encryption system. They based the security analysis on Lindner and Peikert's [LP11] work, showing that an LWE-based cryptosystem is secure as long as

$$d > \log\left(\frac{q}{\sigma}\right)\frac{\lambda + 110}{7.2}. \tag{2}$$

When applied with homomorphic schemes, this relation acquires a challenging aspect, because as the standard deviation increases, less homomorphic operations can be evaluated, since a larger initial noise would be rapidly propagated, what would require a larger modulus q, depending on the circuit multiplicative depth. Thus, as the ratio q/σ determines the LWE-based cryptography security, in order to avoid managing the growth of such weakly related functions, for instance the mutually dependent values of d, q and σ, we can fix a sufficiently large minimum value for σ, such that attacks that explore small standard deviations are mitigated [AG11]. Then we can focus on the choice of parameters d and q for a fixed standard deviation, for example $\sigma \approx 4$, what constitutes a good starting point for homomorphic encryption and, as we are going to see later, q just needs to be chosen big enough to accommodate the noise growth along the computation.

The Magma code in Table 3 implements a simplified version of the attack and can be used to study the security of LWE-based cryptography for small instances.

4.4 Dimension reduction

The decryption algorithm described in Definition 4.2 is similar when compared to ElGamal cryptosystem, because the ciphertext is formed by two polynomials, $c = [c_0, c_1]$, while the private key is given by $s = [1, s^\star]$. Hence, decryption can be represented by

```
lwe_attack := function(n,m,q,A,b)
        qI := ZeroMatrix(Z, m, m);
        for i := 1 to m do qI[i,i] := q; end for;
        A1 := VerticalJoin(Transpose(A),qI);
        X, U := HermiteForm(A1);
        B := RowSubmatrixRange(X,1,m);
        BB := LLL(B: TimeLimit:=1);
        L := Lattice(BB);
        tb := Transpose(b)[1];
        w := ClosestVectors(L,tb : TimeLimit:=1);
        e := Matrix(Z,m,1,[b[i,1]-w[1,i] : i in [1..m]]);
        return e;
end function;
```

Table 3: LWE attack

$$m = [c_0 + c_1 s^\star]_q \pmod 2.$$

If we interpret s^* symbolically, the expression $c_0 + c_1 s^\star$ represents a polynomial of degree 1. To multiply two ciphertexts, $c = \textsc{Enc}_{\mathrm{pk}}(m)$ and $c' = \textsc{Enc}_{\mathrm{pk}}(m') = c'_0 + c'_1 s^\star$, we can compute

$$(c_0 + c_1 s^\star)(c'_0 + c'_1 s^\star) = c_0 c'_0 + (c_0 c'_1 + c'_0 c_1) s^\star + c_1 c'_1 (s^\star)^2.$$

If q is sufficiently big, as we replace $s^\star$ by the private key in last expression, we obtain a polynomial that can be used to recover $m.m'$. However, multiplication gives us back an element in higher dimension and a compact cryptosystem can not have such an increase in the ciphertext size, then we must provide an algorithm to reduce the dimension of the ciphertext. This task is achieved by using an algorithm called $\mathrm{SwitchKey}$, that based on public parameters, returns a ciphertext that can usually be decrypted.

Definition 4.3. Given a polynomial x, we define the algorithm $\mathrm{BitDecomp}$, that returns $\log q$ binary polynomials x_i, where each coefficient of x is writen in binary representation and x_i coefficients are the corresponding i-th bit of this representation. Namely, we have that

$$\mathrm{BitDecomp}(x) = [x_0, \ldots, x_{\log q}],$$

such that

$$x = \sum 2^i x_i.$$

Furthermore, consider the algorithm $\mathrm{PowerOf2}$, that returns $\log q$ polynomials in the form $2^i x$, as follows:

$$\mathrm{PowerOf2}(x) = [x, 2x, \ldots, 2^{\lfloor \log q \rfloor} x].$$

By construction, we have that

$$\langle \mathrm{BitDecomp}(c), \mathrm{PowerOf2}(s) \rangle = \langle c, s \rangle \pmod{q}.$$

Definition 4.4. Also, we define the algorithm $\mathrm{SwitchKeyGen}$ as follows:

1. given a vector of polynomials s'', derived from private key $\texttt{sk}$, generate a random matrix $\overline{A'}$ with $\overline{N}$ rows and 2 columns, where $\overline{N} = 3\log^2 q$, compute $\overline{A} = \overline{A'}\texttt{sk} + e'$, where e' is vector of N rows whose elements are generated using distribution $\mathcal{D}_{d,\sigma,q}$;

2. return $\overline{B} = \overline{A} + \mathrm{PowerOf2}(s'')$, where $\mathrm{PowerOf2}(s'')$ is added to the first column of $\overline{B}$.

Thus, given an expanded ciphertext $\bar{c}$, the algorithm $\mathrm{SwitchKey}$ can simply be defined as

$$\mathrm{SwitchKey}(\bar{c}) = \mathrm{BitDecomp}(\bar{c})^T\overline{B}.$$

Matrix $\overline{B}$ works as an alternative to the SSSP problem, described in previous constructions. In other words, it is analogous to the encryption of the private key using its own public key, such that we are again assuming circular security. Brakerski and Vaikuntanathan [BV11b] proposed an alternative to make circular assumption not necessary. They transformed the basic scheme in order to show that ciphertexts encryptying functions of the secret key are indistinguishable from ciphertexts encryptying zero.

4.5 Modulus reduction

Cryptosystems defined before BGV have a common problem: the noise grows quadratically with multiplications. In order to transpose this barrier, Brakerski and Vaikuntanathan [BV11a] proposed a new technique to manage the noise. Basically, if the initial noise is proportional to r, after k levels of multiplications this noise would be proportional to r^{2^k}. The proposed solution was to use a decreasing moduli chain $q_i \approx q/r^i$. After the first multiplication, we adjust the ciphertext c, multiplying it by $1/r$ and fixing parity if necessary, and replacing modulus q by q/r. This change seems to bring no gain and you can not repeat this procedure arbitrarily, because the chain decreases fastly (linearly with respect to the circuit depth) to a minimum value. However, it is easy to show that the noise is reduced in the same proportion $1/r$, that is, after the k-th multiplication, we obtain a noise proportional to r^k, instead of r^{2^k}. Therefore, there is an exponential gain involved in this transformation. When this chain reaches its end, it is necessary to use bootstrapping to continue the computation, but this subject is outside the scope of this paper.

Definition 4.5. Given a vector of polynomials x, the algorithm $\mathrm{Scale}(x, q_i, q_{i+1})$ computes the vector of polynomials x' closest to $(q_{i+1}/q_i)x$, such that

$$x' = \mathrm{Scale}(x, q_i, q_{i+1}) = \lfloor (q_{i+1}/q_i)x \rceil \equiv x \pmod t.$$

Brakerski proposed a construction that avoids the scaling technique, called ***scale invariant*** [Bra12]. In this construction, ciphertexts are composed by elements in the interval $(-1/2, 1/2]$, i. e., they are maintained in fractional form, then there is no need for modulus switching. Moreover, the construction have a classical reduction to lattice hard problems without using an exponential (in the degree d) modulus q. This reduction works only for the standard LWE and the scheme presents better performance only for $L \geq 20$. Therefore, determining if this technique is preferable will depend on the parameters. For some choices it will, for othe choices the modulus switching approach will be the best option.

4.6 BGV

In this section we describe BGV scheme [BGV11], that can homomorphically deal with circuits of multiplicative depth at most L. Hence, if we know the maximum L necessary for a determined application, then we can derive optimal parameters for the utilization of the BGV scheme. Moreover, we avoid the usage of the expensive bootstrapping method.

Definition 4.6. **Setup.** Given the security parameter λ and the multiplicative depth L, compute $\mu = \theta(\log\lambda + \log L)$. For i varying from L to 0, run the $\textsc{Setup}(\lambda, (i{+}1)\mu)$ algorithm of scheme $\mathcal{E}_{R,i}$, obtaining a decreasing chain of moduli q_i.

Key Generation. Run $(\mathtt{sk}_i, \mathtt{pk}_i) = \mathcal{E}_{R,i}.\textsc{KeyGen}(\lambda)$ for each level i of the circuit. Compute $s'_i = \mathtt{sk}_i \otimes \mathtt{sk}_i$ and $s''_i = \mathrm{BitDecomp}(s'_i, q_i)$. Finally, compute $\overline{B_i} = \mathrm{SwitchKeyGen}(s''_i, \mathtt{sk}_{i-1})$, for $i > 0$. Output the key pair $(\mathtt{sk}, \mathtt{pk})$, where the private key $\mathtt{sk}$ is formed by the values of $\mathtt{sk}_i$, while public key $\mathtt{pk}$ corresponds to the public keys $\mathtt{pk}_i$ together with $\overline{B_i}$.

Encription. Given the message $m \in \{0,1\}^t$, return $c = \textsc{Enc}_{\mathtt{pk}_L}(m)$.

Decription. Given the ciphertext c, at level i of the circuit, utilize the private key $\mathtt{sk}_i$ to compute $m = \textsc{Dec}_{\mathtt{sk}_i}(c)$.

Consider the ciphertexts $c = c_0 + c_1 s^\star$ and $c' = c'_0 + c'_1 s^\star$. We have that addition $c + c'$ can be computed component-wisely by

$$c + c' = (c_0 + c'_0) + (c_1 + c'_1)s^\star,$$

while multiplication is done by tensor product of the ciphertexts, obtaining

$$c \times c' = c_0c'_0 + (c_0c'_1 + c'_0c_1)s^\star + c_1c'_1(s^\star)^2,$$

where each of the coefficients is a polynomial and the vector composed by these three coefficients is called ***expanded ciphertext***. The algorithm $\mathrm{Recrypt}$ maps expanded ciphertexts to regular ciphertexts and is defined in Algorithm 4.1.

Algorithm 4.1 Recrypt

```
INPUT The expanded ciphertext $\overline{c}$, the moduli $q_i$ and $q_{i+1}$.
OUTPUT The ciphertext $c$.
  $c = \mathrm{PowerOf2}(\overline{c}, q_i)$ (the following condition is valid: $\langle c_1, s_i'' \rangle = \langle \overline{c}, s_i' \rangle$).
  $c = \mathrm{Scale}(c, q_{i+1}, q_i)$.
  $c = \mathrm{SwitchKey}(c, q_i, \overline{B_i})$.
  return $c$.
```

4.6.1 Batch operations

In this section we will describe an important optimization to the scheme previously presented. The idea consists in using the Chinese remainder theorem (CRT) to allow simultaneous operations over a vector of messages. In the literature, this concept is associated with the SIMD model, because we have the capacity of parallel computation over vectors [SV11]. This parallelism allows to encode more information inside each ciphertext and therefore can be used to reduce the overhead of homomorphic encryption schemes. Concretely, it is possible to reduce the computation complexity per operation to a polylog overhead, achieving a big improvement with respect to the previous constructions. However, the circuit that will be homomorphically evaluated must have average width in $\Omega(\lambda)$ [GHS12a].

Furthermore, the capacity to perform additions and multiplications over vectors is not a complete computational model. For instance, it lacks the ability to permute vector elements. To solve this problem it is possible to use the Frobenius automorphisms over the ciphertext, obtaining circular rotations of the underlying vector elements. However, it is not possible to calculate every permutation using only circular rotations. In order to get any permutation, it is used a permutation network, that allows us to combine left and right rotations to achieve more complicated permutations. More details can be found in the work of Halevi and Shoup [HS14].

Consider the integers n (usually a power of 2) and p, such that $p^d \equiv 1 \pmod{m}$. Thus $\mathbb{Z}_p$ contain an n-th primitive root of unity $\zeta_n \in \mathbb{Z}_p$, then the n-th cyclotomic polynomial $\phi_n(x)$ is such that its degree is equal to Euler's totient function $\varphi(n)$ and it can be factored into $\ell = \varphi(n)/d$ degree-d terms modulo p

$$\phi_n(x) = \prod_{i \in \mathbb{Z}_m^\star} (x - \zeta_n^i) \pmod p.$$

A polynomial $a(x) \in \mathbb{Z}_p[x]/\Phi_n(x)$ can be represented by a vector containing the coefficients of the polynomial for each power of x, or it can be represented by a vector containing the evaluations of $a(x)$ over the n-th primitive roots of unity. Thus, there are two possible representations: ***by coefficient*** or ***by evaluation***. The later is denoted by $\textsc{Coefs}(a(x))$, while the second one is denoted by $\textsc{Evals}(a(x))$ and both representations are related by the Vandermonde matrix V_d as follows

$$\textsc{Evals}(a(x)) = V_d \textsc{Coefs}(a(x)).$$

If the LWE problem modulus q itself can be a product of primes $p_i \equiv 1 \pmod n$ (and $p_i \equiv 1 \pmod p$ to obtain better noise growth), we can use the so-called ***double CRT***, because the elements of $\mathbb{Z}_q$ can be decomposed with respect to the factors p_i and the factors of $\phi_n(x)$.

Recent utilization of cyclotomic rings is concentrated to the case where n is a power-of-two. As argumented by Lyubashevsky, Peikert and Regev [LPR13], this choice for n have both advantages and disadvantages. For instance, such a choice allows us to easily adapt the classical FFT in order to transform from one representation to another, allowing faster arithmetic over the evaluation representation, because the operations can be computed component-wisely in linear time. In general, choosing a power-of-two makes things simpler to understand and implement, what explains why many constructions adopted this choice. On the other hand, it would be better to choose n as the minimal integer satisfying Equation 2, while the next power-of-two may be twice as far as the best possible value, what constitutes an argument against such possibility. Nevertheless, this choice of n leads to larger ***expansion factors***. A work that shows many contributions related to the ring-LWE for non-power-of-two cyclotomic rings was proposed by Gentry, Halevi, Peikert and

Smart [GHPS12].

4.7 Setting parameters

To provide parameters for the BGV cryptosystem, we must grant that the LWE problem is hard for all the moduli q_i, for $0 \leq i < L$. Also, we must garantee that the ciphertext is decryptable for every i. These two conditions have a circular dependency, but although it is not simple to satisfy all the requirements, we will show how to find parameters that respect both of them. As previously defined, we must choose q_i with $(i+1)\mu$ bits, for $\mu = \theta(\log L + \log \lambda)$, but the hidden constant in this expression may assume different values depending on the SHE variant we are using.

If you previously determined the circuit you want to evaluate, then you can compute the minimum q_{L-1} and the ratio between subsequent value q_i and q_{i-1} in order to allow the correctness of the homomorphic computation. Ana Costache and Nigel Smart [CS15b] studied the parameter choice of many variants of the BGV and NTRU schemes, compared them, concluding that NTRU is better only for very small plaintext size. For $t > 5$, we have that BGV is the better choice. They considered two algorithms for $\mathrm{SwitchKey}$ and the possibility of using the scale invariant scheme. The noise growth depends on the initial noise and the number of additions and multiplications in the circuit. The authors constructed tables of parameters for $t \in \{2, 101, 2^{32}, 2^{64}, 2^{128}, 2^{256}\}$ and $L \in \{2, 5, 10, 20, 30\}$. They also generalized the definition of $\mathrm{PowerOf2}$ and $\mathrm{BitDecomp}$ to allow not only base $w = 2$, but any power of 2, turning it possible to look for optimal values for w. Choosing parameters is not an easy task, because we not only have the mentioned circular dependency, but also we have many details to determine depending on the desired application, like for example the utilization of batch operations. Nevertheless, we can describe an abstract recipe that helps to accomplish this task:

1. stablish a value for the standard deviation σ that avoids attacks described in the literature [AG11]. It is important to remark that this choice may or may not

consider the worst-case connection to lattice problems, since we would need to obey the relation $\sigma \in \omega(\sqrt{n})$ in order to have worst case reduction;

2. compute the minimal modulus q_{L-1} such that the scheme is correct for the highest level. This step depends on the base w in algorithms $\mathrm{PowerOf2}$ and $\mathrm{BitDecomp}$. In special, it is possible to iterate through different values for w to encounter the optimal one;

3. compute the dimension n using Equation 2 to obtain the security of the cryptosystem;

4. calculate the size of the intermediate moduli q_i, for $0 \leq i < L$ and verify if the scheme is correct for all i.

Actually, we could invert our strategy, computing the minimal dimension n to obtain the correctness, and afterwards we determine the value of q_L that provides the security according to Equation 2. If we have an special interest in some specific value for the modulus q_L, then it is possible to fix this parameter and recalculate the others in order to get correctness and security together.

A good measurement for homomorphic encryption perfomance is the comparison of ciphertext size and plaintext size. But a first remark that is important to be done is that many homomorphic encryption schemes allow parallel computation of ℓ slots per encryted message. Table 4 shows the ciphertext size and number of slots of the BGV scheme for different values of L. The scale invariant scheme has smaller ciphertext size only for big L and, for $L \leq 20$, modulus switching technique has smaller ciphertext size. However, bigger ciphertext can encode bigger plaintexts, and depending on the target application it could be better to choose a determined setting or another.

	Modulus Switching		Scale Invariant	
L	size (KBits)	slots	size (KBits)	slots
2	424	1982	792	2714
5	2488	4817	3088	8567
10	10000	9664	10192	18170
20	40832	19542	37472	37742
30	91128	29199	82448	57130

Table 4: Ciphertext size and number of slots

References

[AG11] S. Arora and R. Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Killian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50:506–519, July 2003.

[BLP$^+$13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA, 2013. ACM.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology - Crypto 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of the 31st Annual Conference on Advances in Cryptology*, CRYPTO'11, pages 505–524, Berlin, Heidelberg, 2011. Springer-Verlag.

[CH11] H. Cohn and N. Heninger. Approximate common divisors via lattices. Cryptology ePrint Archive, Report 2011/437, 2011. `http://eprint.iacr.org/`.

[CMNT11] J. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Proceedings of the 31st annual conference on Advances in cryptology*, CRYPTO'11, pages 487–504, Berlin, Heidelberg, 2011. Springer-Verlag.

[CN12] Y. Chen and P. Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519. Springer, 2012.

[CNT11] J. Coron, D. Naccache, and M. Tibouchi. Optimization of fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/440, 2011. `http://eprint.iacr.org/`.

[CNT12] J. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer Berlin Heidelberg, 2012.

[CS15a] J. Cheon and D. Stehlé. Fully homomophic encryption over the integers revisited. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 513–536. Springer Berlin Heidelberg, 2015.

[CS15b] A. Costache and N. P. Smart. Which ring based somewhat homomorphic encryption scheme is best? Cryptology ePrint Archive, Report 2015/889, 2015. `http://eprint.iacr.org/`.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22, 1976.

[DM15] L. Ducas and D. Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In E. Oswald and M. Fischlin, editors, *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, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[FK94] M. Fellows and N. Koblitz. Combinatorial cryptosystems galore! In G. L. Mullen and P. J.-S. Shiue, editors, *Finite Fields: Theory, Applications, and Algorithms*, volume 168 of *Contemporary Mathematics*, pages 51–61. AMS, 1994.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA, 2009. ACM.

[GHPS12] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. Cryptology ePrint Archive, Report 2012/240, 2012. `http://eprint.iacr.org/`.

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482, 2012.

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. *IACR Cryptology ePrint Archive*, 2012:99, 2012.

[GM82] S. Goldwasser and S. Micali. Probabilistic Encryption and How To Play Mental Poker Keeping Secret All Partial Information. In *Proc. $14$th ACM Symp. on Theory of Computing*, pages 270–299. ACM, 1982.

[HG01] N. Howgrave-Graham. Approximate integer common divisors. In *CaLC*, pages 51–66, 2001.

[HS14] S. Halevi and V. Shoup. Algorithms in helib. Cryptology ePrint Archive, Report 2014/106, 2014. `http://eprint.iacr.org/`.

[KLYC13] J. Kim, M. S. Lee, A. Yun, and J. H. Cheon. CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057, 2013. `http://eprint.iacr.org/`.

[Lag85] J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, February 1985.

[Len87] A. K. Lenstra. Factoring multivariate polynomials over algebraic number fields. *SIAM J. Comput*, 16, 1987.

[Lep14] T. Lepoint. *Design and Implementation of Lattice-Based Cryptography*. PhD thesis, École Normale Supérieure and University of Luxembourg, June 2014.

[LMSV11] J. Loftus, A. May, N. P. Smart, and F. Vercauteren. On CCA-secure somewhat homomorphic encryption. In *In Selected Areas in Cryptography*, pages 55–72, 2011.

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Proceedings of the 11th International Conference on Topics in Cryptology: CT-RSA 2011*, CT-RSA'11, pages 319–339, Berlin, Heidelberg, 2011. Springer-Verlag.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Advances in Cryptology EUROCRYPT 2010*, 6110/2010(015848):1?23, 2010.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[NS01] P. Nguyen and J. Stern. The two faces of lattices in cryptology. In J. H. Silverman, editor, *Cryptography and Lattices: International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers*, pages 146–180, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[pro] Gnu project. GMP website. `https://gmplib.org/`. Accessed: 2014-08-22.

[RAD78] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[Reg05a] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, New York, NY, USA, 2005. ACM.

[Reg05b] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[Rog15] P. Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. `http://eprint.iacr.org/`.

[RSA83] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26:96–99, January 1983.

[Sho] V. Shoup. NTL website. `http://www.shoup.net/ntl/`. Accessed: 2014-08-22.

[SS10] D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.

[SV09] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, Report 2009/571, 2009. `http://eprint.iacr.org/`.

[SV11] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.

[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg, 2010. Springer-Verlag.