NTNU
Norwegian University of
Science and Technology

Fully Homomorphic Encryption

Harald Elias Bjerke Sperre

Master of Science in Physics and Mathematics
Submission date: July 2017
Supervisor: Kristian Gjøsteen, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

NTNU
Department of Mathematical Sciences

Homomorphic Encryption

Harald Sperre

July 22, 2017

MASTER THESIS
Department of Mathematical Sciences
Norwegian University of Science and Technology

Supervisor: Professor Kristian Gjøsteen

Abstract

In this paper we consider fully homomorphic encryption based on the learning with errors problem. We present the problem as introduced by Oded Regev in 2009 and explain a simple public key cryptosystem based on it. We show how the scheme can be modified to be more suitable for homomorphic operations, and introduce bootstrapping, using the ideas proposed by Craig Gentry in 2009. Finally we examine bootstrapping techniques proposed by Chilotti et al. in 2016, which is the main topic of this paper.

Contents

Chapter 1

Introduction

In most of the history of cryptography, users have been concerned only with sending and receiving messages securely. Then, during the late 20th century, computers became prevalent and storing data encrypted became just as important as sending it encrypted. This new usage created, in turn, a desire for a new property of the encryption schemes: *homomorphism*. We say that an encryption scheme is homomorphic if it allows some procedure Eval satisfying:

- it takes a function f and some ciphertexts $c_1, \ldots, c_k$ as input
- it outputs a valid ciphertext c
- if each c_i is a fresh encryption of μ_i and the function f belongs to some set of permitted functions, then c is an encryption of $f(\mu_1, \ldots, \mu_k)$.

The two assumptions in the last line requires some explaining. The limitation on which functions are permitted is required because each ciphertext contains a certain amount of noise, which masks the plaintext. Then, homomorphic operations increase this noise, so evaluating a very complex function could increase the noise by so much that a decryption would give the wrong plaintext. Hence, the set of permitted functions would only contain functions simple enough to not increase the noise by too much. A fresh ciphertext is one that is output from the encryption function rather than a result of a homomorphic operation. This requirement exists because if one uses the result of an evaluation in a new evaluation then the noise may already be so large that even simple functions cannot be evaluated homomorphically and give a decryptable result.

Naturally, a lot of interest lies in removing the two limitations described above. In *fully homomorphic encryption* (FHE) one can evaluate arbitrarily complex functions and use the resulting ciphertexts as input in new evaluations, in an arbitrarily deep circuit.

The first fully homomorphic encryption scheme was introduced by Craig Gentry in 2009[1]. This was accomplished by employing a technique he called *bootstrapping*, namely evaluating the decryption algorithm homomorphically. The result is a new encryption of the same plaintext, which "looks more like" a fresh encryption.

More precicely, the bootstrapping procedure is used to reset the noise to some fixed level, such that more operations can be performed without pushing the noise outside the decryptable range. In order for this to work, the decryption algorithm clearly needs to be simple enough to be able to evaluate homomorphically without introducing too much noise. Because of this, Gentry looked for encryption schemes with simple decryption algorithms, and ended up with lattice-based encryption. A lot of work have been done after Gentry's first publication to make more efficient schemes, and many of the attempts also focus on lattices.

Another mathematical problem, called *learning with errors* (LWE) was introduced in 2009 by Oded Regev[2]. This problem is concerned with finding solutions to so-called *equations with errors*, where the error in each equation is unknown, but satisfies a known probability distribution. I turns out that the problem shares some properties with some hard-to-solve lattice problems, and is suitable for use in encryption.

After its introduction, a lot of work has been done with LWE to design homomorphic encryption schemes, both with and without bootstrapping. The scheme we consider in this paper was introduced by Gentry, Sahai and Waters[3], and we look at a bootstrapping procedure called FHEW[4] with improvements by Chillotti, Gama, Georgieva and Izabachène[5].

1.1 Overview

We start out in the next chapter by introducing some notation and definitions needed to follow the paper. These are mainly from basic algebra as well as probability theory and some specific lattice problems. Chapter 3 follows by introducing the LWE problem as well as details on the GSW encryption scheme and a short discussion on the security of LWE based schemes. Chapter 4 gives a brief overview of the bootstrapping procedure of FHEW to serve as an appetizer for the improvements whic we present in chapter 5.

Chapter 2

Preliminaries

This chapter introduces some notation and some important concepts for the rest of the paper. We include some important concepts of algebra and probability theory, as well as some specific probability distributions and lattice problems.

2.1 Basic notation and definitions

The real torus $\mathbb{R}/\mathbb{Z}$ will be denoted $\mathbb{T}$. This is the set of real numbers modulo 1. Further, we define $\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1) \bmod 1$, and $\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$. Denote by $\mathbb{B}$ the set $\{0, 1\}$. For any ring R we let $\mathcal{M}_{p,q}(R)$ denote the set of $(p \times q)$-matrices with entries in R.

Over $\mathbb{Z}$ and $\mathbb{R}$ we use the standard norms, $\|\cdot\|_p$ and $\|\cdot\|_\infty$. For $P \in \mathbb{T}[X]$, we let $\|P(X)\|_p$ mean the norm of the coefficient vector of P. Similarily, for $P \in \mathbb{T}_N[X]$, $\|P\|_p$ is the norm of the unique representative of degree less than N.

For $\mathbf{x} \in \mathbb{T}^k$, we write $\|\mathbf{x}\|_p = \min_{\mathbf{u} \in \mathbf{x} + \mathbb{Z}^k}(\|\mathbf{u}\|_p)$, although this is not an actual norm on $\mathbb{T}^k$. However, for $\mathbf{x} \in \mathbb{T}^k$, $\|\mathbf{x}\|_p$ represents the norm of the representative of $\mathbf{x}$ in which all coefficients are in $(-1/2, 1/2]$. Similarily for a polynomial $a \in \mathbb{T}_N[X]$, we define $\|a\|_p$ as the p-norm of a's representative in $\mathbb{T}[X]$ with degree less than N and coefficients in $(-1/2, 1/2]$.

2.2 Some probability theory

A probability distribution on a set A is a function $\phi : A \to [0, 1]$ such that $\sum_{\mathbf{x} \in A} \phi(\mathbf{x}) = 1$ or $\int_A \phi(\mathbf{x})\, \mathrm{d}\mathbf{x} = 1$. A probability space is a pair (A, ϕ) where A is a set and ϕ is a probability distribution on A. When the meaning is clear, we may use probability distributions and probability spaces interchangibly.

Let ϕ_1, ϕ_2 be probability distributions defined on some space A. The *statistical distance* between ϕ_1 and ϕ_2 is given by

$$\Delta(\phi_1, \phi_2) = \frac{1}{2} \int_{\mathbb{A}} |\phi_1(\mathbf{x}) - \phi_2(\mathbf{x})| \, \mathrm{d}\mathbf{x}$$

if A is continuous and

$$\Delta(\phi_1, \phi_2) = \frac{1}{2} \sum_{\mathbf{x} \in A} |\phi_1(\mathbf{x}) - \phi_2(\mathbf{x})|$$

if A is discrete.

This distance function satisfies the triangle inequality, that is

$$\Delta(\phi_1, \phi_3) \leq \Delta(\phi_1, \phi_2) + \Delta(\phi_2, \phi_3)$$

for any distributions ϕ_1, ϕ_2, ϕ_3 defined on the same domain. Another important fact about the statistical distance, is that it cannot increase by applying any function to the two probability distributions. That is,

$$\Delta\left(f(\phi_1), f(\phi_2)\right) \leq \Delta(\phi_1, \phi_2).$$

We say that a distribution ϕ on the torus $\mathbb{T}$ is *concentrated* if its support is contained in a ball of radius $\frac{1}{4}$, except with negligible probability. For such a distribution, we define its variance as $\mathsf{Var}(\phi) = \min_{\bar{x} \in \mathbb{T}} \int_{\mathbb{T}} \phi(x)|x - \bar{x}|^2 \, \mathrm{d}x$ and its expectation $\mathsf{E}(\phi)$ as the value of $\bar{x}$ obtaining this minimum. This definition and the following lemma are analog to those presented by Chilotti et al. in [5].

Lemma 2.1. *Let ϕ_1, ϕ_2 be independent, concentrated distributions on either $\mathbb{T}, \mathbb{T}^k$, or $\mathbb{T}_N[X]^k$ and $e_1, e_2 \in \mathbb{Z}$, such that $\phi = e_1\phi_1 + e_2\phi_2$ is also concentrated. Then, $\mathsf{E}(\phi) = e_1\mathsf{E}(\phi_1) + e_2\mathsf{E}(\phi_2)$ and $\mathsf{Var}(\phi) \leq e_1^2\mathsf{Var}(\phi_1) + e_2^2\mathsf{Var}(\phi_2)$.*

2.3 The Gaussian distribution

Next follow some definitions and results regarding the Gaussian distribution, which is commonly used to sample the error terms in the LWE problem. Most of the content in this section is found in [2].

Let ρ_σ denote the Gaussian function, given by

$$\rho_\sigma \, : \, \mathbb{R}^n \to (0, 1) \quad ; \quad \mathbf{x} \mapsto e^{-\pi \|\mathbf{x}/\sigma\|^2}.$$

Then $\nu_\sigma = \rho_\sigma / \sigma^n$ is a probability density function on $\mathbb{R}^n$ and can be sampled by sampling n independent standard normal variables and multiplying each by σ. For a countable set $A \subset$, define

$$\rho_\sigma(A) = \sum_{\mathbf{x} \in A} \rho_\sigma(\mathbf{x}),$$

and similarly for ν_σ. Then, the *discrete Gaussian probability density function* is given by

$$D_{A,\sigma} : A \to \mathbb{R} \quad ; \quad \mathbf{x} \mapsto \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(A)}.$$

Note that this is actually a density function, since the division normalizes the function to sum to 1.

For any vector $\mathbf{c} \in \mathbb{R}^n$ we can define the shifted Gaussian function $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \rho_\sigma(\mathbf{x} - \mathbf{c})$. For sufficiently small $\mathbf{c}$, the following lemma limits the amount by which $\rho_{\sigma,\mathbf{c}}$ differ from ρ_σ.

Lemma 2.2. *For $\sigma, t, l > 0$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $\|\mathbf{x}\| \leq t$, $\|\mathbf{x} - \mathbf{y}\| \leq l$,*

$$\rho_\sigma(\mathbf{y}) \geq (1 - \pi(2lt + l^2)/\sigma^2)\rho_\sigma(\mathbf{x})$$

Proof. Using the fact that $e^{-z} \geq 1 - z$, we get

$$\rho_\sigma(\mathbf{y}) = e^{-\pi\|\mathbf{y}/\sigma\|^2} \geq e^{-\pi(\|\mathbf{x}\|/\sigma + l/\sigma)^2} = e^{-\pi(\|\mathbf{x}\|/\sigma^2 + (l/\sigma)^2)}\rho_\sigma(\mathbf{x}) \geq (1 - \pi(2lt + l^2)/\sigma^2)\rho_\sigma(\mathbf{x}).$$

□

We say that a distribution ϕ is *σ-subgaussian* if its tails are bounded by the Gaussian function with parameter σ. That is, $\mathsf{P}(|\phi| \geq x) \leq 2\rho_\sigma(x)$. The following result from [5], here presented without proof, tells us that a subgaussian distribution with sufficiently small parameter must be concentrated.

Lemma 2.3. *Let ϕ be a distribution on $\mathbb{T}, \mathbb{T}^k$, or $\mathbb{T}_N[X]^k$ where each coefficient is σ-subgaussian. If $\sigma \leq 1/\sqrt{32\log(2)(\lambda + 1)}$, then at most a fraction $2^{-\lambda}$ of the distribution's mass is outside the interval $[-1/4, 1/4]$.*

For $\beta > 0$, let $\Psi_\beta : \mathbb{T} \to [0, 1]$ be the distribution on the torus generated by sampling ν_σ (in one dimension) and reducing modulo 1, with $\sigma = \beta/\sqrt{2\pi}$. This distribution has probability density function given by

$$\Psi_\beta(r) = \nu_\sigma(A), \quad \text{where } A = \bigcup_{k=-\infty}^{\infty} \{r + k\}.$$

An important property of this distribution is that a small change in the parameter β changes the distribution by a very small amount.

Lemma 2.4. *Let $0 < \alpha < \beta \leq 2\alpha$. Then*

$$\Delta(\Psi_\alpha, \Psi_\beta) \leq \frac{5}{2}\left(\frac{\beta}{\alpha} - 1\right).$$

Proof. Recall that $\Delta(f(\phi_1), f(\phi_2)) \leq \Delta(\phi_1, \phi_2)$ for any function f and pair ϕ_1, ϕ_2 of distributions. With $f(x) = x \bmod 1$, we get

$$\Delta\left(\Psi_\alpha, \Psi_\beta\right) = \Delta\left(f\left(\nu_{\alpha/\sqrt{2\pi}}\right), f\left(\nu_{\beta/\sqrt{2\pi}}\right)\right) \leq \Delta\left(\nu_{\alpha/\sqrt{2\pi}}, \nu_{\beta/\sqrt{2\pi}}\right).$$

We can scale the parameters, setting $\alpha = 1$, $\beta = 1 + \epsilon$ for some $0 < \epsilon \leq 1$. The statistical difference between these normal distributions is

$$\begin{aligned}
\Delta\left(\nu_{\frac{1}{\sqrt{2\pi}}}, \nu_{\frac{1+\epsilon}{\sqrt{2\pi}}}\right) &= \frac{1}{2}\int_{-\infty}^{\infty}\left|\nu_{\frac{1}{\sqrt{2\pi}}}(x) - \nu_{\frac{1+\epsilon}{\sqrt{2\pi}}}(x)\right| \mathrm{d}x = \frac{1}{2}\int_{-\infty}^{\infty}\left|e^{-\pi x^2} - \frac{1}{1+\epsilon}e^{-\frac{\pi x^2}{(1+\epsilon)^2}}\right| \mathrm{d}x \\
&\leq \frac{1}{2}\left(\int_{-\infty}^{\infty}\left|e^{\pi x^2} - e^{-\frac{\pi x^2}{(1+\epsilon)^2}}\right| \mathrm{d}x + \int_{-\infty}^{\infty}\left|1 - \frac{1}{1+\epsilon}e^{-\frac{\pi x^2}{(1+\epsilon)^2}}\right| \mathrm{d}x\right) \\
&= \frac{1}{2}\int_{-\infty}^{\infty}\left|e^{-\pi x^2} - e^{-\frac{\pi x^2}{(1+\epsilon)^2}}\right| \mathrm{d}x + \frac{\epsilon}{2} \\
&= \frac{1}{2}\int_{-\infty}^{\infty}\left|e^{-\pi(1-1/(1+\epsilon)^2)x^2} - 1\right| e^{-\frac{\pi x^2}{(1+\epsilon)^2}} \mathrm{d}x + \frac{\epsilon}{2}
\end{aligned}$$

Further, since $1 \geq e^{-z} \geq 1 - z$ for any $z \geq 0$,

$$\begin{aligned}
\left|e^{-\pi x^2\left(1-1/(1+\epsilon)^2\right)} - 1\right| &= 1 - e^{-\pi x^2\left(1-1/(1+\epsilon)^2\right)} \\
&\leq 1 - \left(1 - \pi x^2\left(1 - \frac{1}{(1+\epsilon)^2}\right)\right) = \pi x^2\left(1 - \frac{1}{(1+\epsilon)^2}\right) \\
&\leq 2\epsilon\pi x^2
\end{aligned}$$

where the last inequality follows from the Macclaurin expansion. Thus,

$$\begin{aligned}
\Delta\left(\Psi_1, \Psi_{1+\epsilon}\right) &\leq \frac{\epsilon}{2} + \frac{1}{2}\int_{-\infty}^{\infty}\left|e^{-\pi x^2\left(1-1/(1+\epsilon)^2\right)} - 1\right| e^{-\frac{\pi x^2}{(1+\epsilon)^2}} \mathrm{d}x \\
&\leq \frac{\epsilon}{2} + \int_{-\infty}^{\infty}\epsilon\pi x^2 e^{-\frac{\pi x^2}{(1+\epsilon)^2}} \mathrm{d}x = \frac{\epsilon}{2} + \frac{\epsilon(\epsilon+1)^3}{2} \\
&\leq \frac{5}{2}\epsilon = \frac{5}{2}\left(\frac{\beta}{\alpha} - 1\right).
\end{aligned}$$

□

2.4 Some (hard) problems on lattices

Since lattice problems have been extensively studied for a long time, we are more confident in the hardness of some well-known lattice problems than the LWE problem, which was introduced comparatively recently. We therefore base security of encryption schemes using LWE on a reduction from some of these lattice problems to LWE. Before we introduce these, we need basic definitions concerning lattices.

A lattice Λ in $\mathbb{R}^n$ is the set of all integer combinations of some set of n linearly independent vectors. This set of vectors is a *basis* for the lattice, although any lattice has infinitely many basis sets. We let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in Λ. Further, λ_k is the infinum of numbers λ such that there exists a set of k non-zero, linearly independent vectors with length less than λ.

The main lattice problem on which we choose to base security is the shortest independent vectors problem (SIVP), which is concerned with finding short, linearly independent vectors in a given lattice.

Definition 2.5 (SIVP_γ)**.** An instance of SIVP_γ is given by an n-dimensional lattice Λ and an approximation factor γ. The solution is any set of n linearly independent vectors of length at most $\gamma\lambda_n(\Lambda)$.

This problem has been studied extensively and is generally considered to be hard to solve[6], for high enough lattice dimensions.

Another problem we consider is the Discrete Gaussian Sampling problem (DGS). To solve the problem, given a lattice, one would have to sample the discrete gaussian distribution on the lattice, with parameter determined by some function φ of the lattice. This problem is useful for us because we will base security of LWE on the hardness of this problem, and then reduce solving SIVP to solving DGS.

Definition 2.6 (DGS_φ)**.** An instance of DGS_φ is given by an n-dimensional lattice Λ and a number $r > \varphi(\Lambda)$. The desired output is a sample from $D_{\Lambda,r}$.

2.5 Logic gates and circuits

A logic gate is essentially an implementation of some boolean function. That is, given one or more boolean values as input, the gate outputs a boolean value corresponding to the result of the function. A logic gate is essentially equivalent to a mathematical operation modulo 2, and these will be used interchangibly. A *circuit* is a tree of logic gates, where the leaves take boolean input, send their output to the gates one level closer to the root and so on, until the root gives output.

A logic gate is said to be *functionally complete* if any boolean expression can be represented using a circuit consisting of only that type of gate. The NAND-gate is the inverted AND-gate, that is

$$\mathsf{NAND}(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 1 \\ 1 & \text{otherwise} \end{cases}$$

The NAND-gate is paticularly useful since it can be used in circuits to emulate any combination of boolean gates. We state this more formally in a lemma.

Lemma 2.7. *The* NAND*-gate is functionally complete.*

Chapter 3

Learning with errors

In this chapter we introduce the encryption scheme on which the bootstrapping procedure we want to examine is used. To this end, we start by introducing the LWE problem, before explaining how it can be used for encryption. Finally, we present a brief overview on security of LWE based encryption.

3.1 The problem

Learning with errors is a problem concerning a system of "equations with errors", introduced by Oded Regev in 2009[2]. Given vectors $\mathbf{a}_i$ and $\mathbf{b} = (b_1, \ldots, b_k)$, the LWE problems are about the existence of some vector $\mathbf{s}$ such that

$$\mathbf{a}_i \cdot \mathbf{s} = b_i + \varepsilon_i,$$

where each ε_i is chosen according to some known probability distribution χ, centered around 0.

Definition 3.1. Let n be a positive integer and let $\alpha > 0$ be a noise parameter. Further, let $\mathbf{s}$ be a secret vector chosen uniformly at random from some bounded set $\mathcal{S} \subset \mathbb{Z}^n$. We denote by $\mathcal{D}^{\text{LWE}}_{\mathbf{s},\alpha}$ the probability distribution on $\mathbb{T}^n \times \mathbb{T}$ obtained by choosing $\mathbf{a}$ uniformly at random from $\mathbb{T}^n$, choosing ε according to a probability distribution with mean 0 and standard deviation α, and returning $(\mathbf{a}, b) = (\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \varepsilon)$.

The definition is focused on LWE samples over the real torus $\mathbb{T}$, since this is the focus of the presentation in [5], to which we give the greatest attention in this thesis. However, the problem can be formulated over other algebraic structures, such as $\mathbb{Z}_q$ or any polynomial

ring. When the problem is formulated over polynomial rings, it is often referred to as ring-LWE.

Building on this, we consider two different LWE problems:

Search LWE Given a number of pairs $(\mathbf{a}, b)$ chosen according to $\mathcal{D}^{\text{LWE}}_{\mathbf{s},\alpha}$, find $\mathbf{s}$.

Decision LWE Given a number of pairs $(\mathbf{a}, b)$ chosen according to either $\mathcal{D}^{\text{LWE}}_{\mathbf{s},\alpha}$ or the uniform distribution on $\mathbb{T}^n \times \mathbb{T}$, decide from which distribution it is drawn.

It is the hardness of these problems which will be fundamental for the security of the encryption schemes we consider later. Clearly, with $\alpha = 0$ (and thus $\varepsilon = 0$) both problems are trivially solvable by Gaussian elimination. In addition to the parameter α, the algebraic structure on which which we work affects the hardness. Naturally, the hardness increases with the dimension n.

3.2 Designing a simple encryption scheme

In the simple encryption scheme which will be presented shortly, we make use of *TLWE samples*, which can vary between regular- and ring-LWE. We use the definition presented in [5], based on generalizations of LWE introduced in [7].

Definition 3.2 (TLWE samples)**.** Let $k \geq 1$ be an integer, N a power of 2, and $\alpha \geq 0$ be a parameter. A TLWE secret key $\mathbf{s} \in \mathbb{B}_N[X]^k$ is a vector of k polynomials in $\mathbb{Z}_N[X]$ with binary coefficients. The message space of TLWE samples is $\mathbb{T}_N[X]$. A (fresh) sample of a message μ with noise parameter α under key $\mathbf{s}$ is an element $(\mathbf{a}, b) \in \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$, where $b \in \mathbb{T}_N[X]$ has Gaussian distribution $\mathcal{D}_{\mathbb{T}_N[X],\alpha}$ around $\mu + \mathbf{s} \cdot \mathbf{a}$.

Based on TLWE, we introduce a simple public key encryption scheme.

Key Generation $\text{sk} : \mathbf{s} \xleftarrow{\text{R}} \mathbb{B}_N[X]^k; \text{pk} : (\mathbf{a}_i, b_i)_{i=1}^m \leftarrow \mathcal{D}^{\text{LWE}}_{\mathbf{s},\alpha}$.

Encryption For each bit μ_i in a message μ, choose a subset $S \subset \{1, \ldots, m\}$ uniformly at random. Then set $c_i \leftarrow (\sum_{j \in S} \mathbf{a}_j, \frac{1}{2}\mu_i + \sum_{j \in S} b_j)$. The ciphertext is $c = (c_1, \ldots, c_M)$.

Decryption For each pair $c_i = (\mathbf{a}_i, b_i)$ in a ciphertext, calculate $b_i - \mathbf{s} \cdot \mathbf{a}_i$. If this is closer to 0 than to $1/2$, set $\mu_i = 0$. Otherwise, $\mu_i = 1$. The decryption is $\mu = \mu_1, \ldots, \mu_M$.

Solving the search LWE problem is equivalent to finding the secret key of the scheme, given a number of ciphertexts. Semantic security of the scheme is equivalent to desicion LWE. To investigate correctness of decryption, we have to consider the statistical distribution of $\mathbf{a}$ and b.

Definition 3.3. Let $\mathbf{c} = (\mathbf{a}, b)$ be a TLWE sample under secret key $\mathbf{s}$. We define the *phase* of the sample as $\varphi_{\mathbf{s}}(\mathbf{c}) = b - \mathbf{s} \cdot \mathbf{a}$.

We will sometimes omit the key and write $\varphi(\mathbf{c})$. Note that the decryption calculation is exactly fnding the phase of each TLWE sample in the ciphertext. If the noise parameter α was 0, the phase would be exactly equal to 0 or 1/2. To show that decryption still works when $\alpha > 0$, we show some more properties of the phase.

Lemma 3.4. *The phase function is linear over* $\mathbb{T}_N[X]^{k+1}$.

Lemma 3.5. *The phase function is (kN+1)-lipschitzian in the* ∞*-norm. That is,* $\|\varphi_{\mathbf{s}}(\mathbf{c}_1) - \varphi_{\mathbf{s}}(\mathbf{c}_2)\|_\infty \leq (kN+1)\|\mathbf{c}_1 - \mathbf{c}_2\|_\infty$

The lipschitzianity is especially useful for allowing approximations, since two samples close to each other will also have phases that are close to each other.

We continue by defining some statistics on a TLWE sample. Note that while these functions are well-defined, they are in general not possible to compute or approximate in practice.

Definition 3.6. Let $\mathbf{c} \in \mathbb{T}_N[X]^{k+1}$ be a random variable. We say that $\mathbf{c}$ is a *valid* TLWE sample if there exists $\mathbf{s} \in \mathbb{B}_N[X]^k$ such that the distribution of the phase $\varphi_{\mathbf{s}}(\mathbf{c})$ is concentrated. Further, define:

- the *message* $\mathsf{msg}(\mathbf{c}) = \mathsf{E}[\varphi_{\mathbf{s}}(\mathbf{c})]$,
- the *error* $\mathsf{Err}(\mathbf{c}) = \varphi_{\mathbf{s}}(\mathbf{c}) - \mathsf{msg}(\mathbf{c})$,
- the variance of the error, $\mathsf{Var}(\mathsf{Err}(\mathbf{c})) = \mathsf{Var}(\varphi(\mathbf{c}))$,
- $\|\mathsf{Err}(\mathbf{c})\|_\infty$, the maximum amplitude of the error.

Note that as long as the error term ε is taken from a distribution with mean 0, the message of a TLWE sample is exactly the message that is encrypted.

3.3 A more homomorphic scheme

We now wish to present a tweaked scheme, which will allow a fairly deep circuit to be evaluated homomorphically. The scheme was introduced by Gentry, Sahai, and Waters[3]. In order to do so, however, we need to first define a few helpful operations, which will make homomorphic operations easier. Note that in this scheme we use LWE over $\mathbb{Z}_q$.

Let $\mathbf{a}, \mathbf{b}$ be vectors of dimension k over $\mathbb{Z}_q$, and let $l = \lfloor \log_2 q \rfloor + 1$ and $N = k \cdot l$. Then, $\mathsf{BitDecomp}(\mathbf{a}) = (a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{k,0}, \ldots, a_{k,l-1})$ is the binary decomposition of

$\mathbf{a}$. That is, $a_{i,j}$ is the jth bit of the binary representation of the ith element of $\mathbf{a}$. Its inverse is given by

$$\mathsf{BitDecomp}^{-1}(a_{1,0},\ldots,a_{1,l-1},\ldots,a_{k,0},\ldots,a_{k,l-1}) = \left(\sum_{j=0}^{l-1} 2^j a_{1,j},\ldots,\sum_{j=0}^{l-1} 2^j a_{k,j}\right).$$

Note that this function is also well-defined on non-binary vectors. Further, let $\mathsf{Flatten}(\mathbf{a}') = \mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\mathbf{a}'))$, for N-dimensional $\mathbf{a}'$. Clearly, when $\mathbf{a}'$ is a binary vector, $\mathsf{Flatten}(\mathbf{a}') = \mathbf{a}'$. In a sense, $\mathsf{Flatten}(\mathbf{a}')$ and $\mathbf{a}'$ encode the same information, even when $\mathbf{a}'$ is non-binary. This can be demonstrated after introducing a new operation, $\mathsf{Powersof2}(\cdot)$:

$$\mathsf{Powersof2}(\mathbf{b}) = (b_1, 2b_1, \ldots, 2^{l-1}b_1, \ldots, b_k, \ldots, 2^{l-1}b_k),$$

where $\mathbf{b} = (b_1,\ldots,b_k)$. Now, $\langle \mathsf{Flatten}(\mathbf{a}'), \mathsf{Powersof2}(\mathbf{b})\rangle = \langle \mathsf{BitDecomp}^{-1}(\mathbf{a}'), \mathbf{b}\rangle = \langle \mathbf{a}', \mathsf{Powersof2}(\mathbf{b})\rangle$.

For each of these functions, define its operation when applied to a matrix to be the matrix formed by operating on each row. Using these operations, we are able to take an LWE secret key and ciphertext and make sure they have some useful properties. In order to make operations on ciphertexts less costly, we want the ciphertext vectors to have binary entries. We need to make sure decryption still works, so we need a corresponding transformation of the secret key. Hence, we use $\mathsf{Flatten}$ and $\mathsf{Powersof2}$, respectively, on the ciphertext and secret key. The full description of the scheme is thus:

$\mathsf{Setup}(1^\lambda, 1^L)$**:** Choose a modulus q of $\kappa(\lambda, L)$ bits, lattice dimension parameter $n(\lambda, L)$ and error distribution $\chi(\lambda, L)$. Also choose a parameter $m(\lambda, L) = O(n \log q)$. Let $\mathsf{params} = (n, q, \chi, m)$ and $l = \lfloor \log q \rfloor + 1$ and $N = (n+1)l$.

$\mathsf{SecretKeyGen}(\mathsf{params})$**:** Sample $\mathbf{t} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. Let $\mathbf{s} \leftarrow (1, -t_1, \ldots, -t_n) \in \mathbb{Z}_q^{n+1}$ and output $\mathsf{sk} = \mathbf{v} = \mathsf{Powersof2}(\mathbf{s})$.

$\mathsf{PublicKeyGen}(\mathsf{params}, \mathsf{sk})$**:** Generate a matrix $B \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{m \times n}$ and vector $\mathbf{e} \leftarrow \chi^m$. Set $\mathbf{b} = B \cdot \mathbf{t} + \mathbf{e}$. Set $A = [\mathbf{b}|B]$ and publish the public key $\mathsf{pk} = A$.

$\mathsf{Enc}(\mathsf{params}, \mathsf{pk}, \mu)$**:** To encrypt a message $\mu \in \mathbb{Z}_q$, sample $R \xleftarrow{\mathrm{R}} \mathbb{B}^{N \times m}$ and output ciphertext

$$C = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A)) \in \mathbb{Z}_q^{N \times N}.$$

$\mathsf{Dec}(\mathsf{params}, \mathsf{sk}, C)$**:** Observe that the first l coefficients of $\mathbf{v}$ are $2^0, \ldots, 2^{l-1}$, exactly one of which is in $(q/4, q/2]$. Let i be such that $v_i \in (q/4, q/2]$ and consider C_i, the ith row of C. Compute $x_i \leftarrow \langle C_i, \mathbf{v}\rangle$ and output $\mu' \leftarrow \lfloor x_i / v_i \rceil$.

$\mathsf{MultConst}(C, \alpha)$**:** To multiply ciphertext $C \in \mathbb{Z}_q^{N \times N}$ by a constant $\alpha \in \mathbb{Z}_q$, set $M_\alpha \leftarrow \mathsf{Flatten}(\alpha I_N)$ and output $\mathsf{Flatten}(M_\alpha \cdot C)$.

$\mathsf{Add}(C_1, C_2)$**:** Output $\mathsf{Flatten}(C_1 + C_2)$.

$\mathsf{Mult}(C_1, C_2)$**:** Output $\mathsf{Flatten}(C_1 \cdot C_2)$.

Consider first the decryption, defined by the matrix-vector product

$$\begin{aligned} C \cdot \mathbf{v} &= \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A)) \cdot \mathbf{v} = (\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A)) \cdot \mathbf{v} \\ &= \mu \cdot \mathbf{v} + R \cdot A \cdot \mathbf{s} = \mu \cdot \mathbf{v} + R \cdot [\mathbf{b}|B] \cdot \begin{bmatrix} 1 \\ -\mathbf{t} \end{bmatrix} = \mu \cdot \mathbf{v} + R \cdot (B \cdot \mathbf{t} + \mathbf{e} - B \cdot \mathbf{t}) \\ &= \mu \cdot \mathbf{v} + R \cdot \mathbf{e}. \end{aligned}$$

When actually performing the decryption, we use only the ith row of C and compute $x_i = C_i \cdot \mathbf{v}$, which is the ith entry in $C \cdot \mathbf{v}$, i.e $x_i = \mu v_i + R_i \cdot \mathbf{e}$. Recall that R has only binary entries, so the error has magnitude

$$|R_i \cdot \mathbf{e}| \leq \|\mathbf{e}\|_1.$$

Thus, when computing $\mu' = \frac{x_i}{v_i}$, the distance from the original plaintext μ i at most $\frac{\|\mathbf{e}\|_1}{v_i}$. Recall that i was chosen such that $v_i \in \left(\frac{q}{4}, \frac{q}{2}\right]$. Then, by chosing χ such that $\|\mathbf{e}\|_1 < \frac{q}{8}$ with overwhelming probability, we have

$$|\mu - \mu'| < \frac{q/8}{q/4} = \frac{1}{2}.$$

Thus, rounding μ' to the closest element in $\mathbb{Z}_q$ yields the correct decryption. However, this result depends on the distribution χ from which we draw the error vector $\mathbf{e}$. Clearly, this choice also impacts security, which we will examine closer in the next section. Note also that in order to allow homomorphic operations, we need to find a tighter bound. Otherwise, any non-trivial operation would risk increasing the error by too much.

Knowing that encryption and decryption works is a good first step, but we also want to be able to do homomorphic operations. To that end, consider what happens to the decryption of the result of the operations listed above:

$$\begin{aligned} \mathsf{MultConst}(C, \alpha) \cdot \mathbf{v} = \mathsf{Flatten}(\alpha I_N \cdot C) \cdot \mathbf{v} &= \alpha I_N \cdot \mathbf{v} \cdot \mu + \alpha I_N \cdot R \cdot \mathbf{e} \\ &= \alpha\mu + \alpha R \cdot \mathbf{e}. \end{aligned}$$

Here, the error is multiplied at most by N.

For adding two ciphertexts, the scheme obviously works since $(C_1 + C_2) \cdot \mathbf{v} = C_1 \cdot \mathbf{v} + C_2 \cdot \mathbf{v}$. The error term will also become the sum of the error terms.

The multiplication of cipertexts is given by

$$\begin{aligned} \mathsf{Mult}(C_1, C_2) \cdot \mathbf{v} = \mathsf{Flatten}(C_1 \cdot C_2) \cdot \mathbf{v} &= C_1 \cdot C_2 \cdot \mathbf{v} \\ &= C_1 \cdot (\mu_2 \mathbf{v} + \mathbf{e}_2) + \mu_2(\mu_1 \mathbf{v} + \mathbf{e}_1) + C_1 \cdot \mathbf{e}_2 \\ &= \mu_1\mu_2 \cdot \mathbf{v} + \mu_2 \mathbf{e}_1 + C_1 \cdot \mathbf{e}_2. \end{aligned}$$

Here we see that the error of the product depends not only on the old errors, but also on the encrypted version of one message, and the plaintext of the other. The ciphertext C_1 contributes at most a factor of N, since it is a binary matrix. The plaintext μ_2 on the other hand, could potentially lead to an increase of factor up to q. This observation leads to a wish to restrict the plaintext space to a much smaller one. One could for example restrict the plaintext space to $\{0, 1\}$.

When restricting the plaintext space to $\{0, 1\}$, the operations one would like to evaluate can be expressed as boolean circuits. Recall that any boolean operation can be described using only NAND-gates. Hence, we only need a homomorphic NAND-operation to evaluate any circuit. We define it as $\mathsf{NAND}(C_1, C_2) = \mathsf{Flatten}(I_N - C_1 \cdot C_2)$. The decryption is then

$$\mathsf{NAND}(C_1, C_2) \cdot \mathbf{v} = (I_N - C_1 \cdot C_2) \cdot \mathbf{v} = (1 - \mu_1\mu_2) \cdot \mathbf{v} - \mu_2 \cdot \mathbf{e}_1 - C_1 \cdot \mathbf{e}_2.$$

3.4 Security of LWE based encryption

When we are now going to examine the security of these LWE based encryption schemes, we must first note that applying transformations such as BitDecomp and its inverse does not affect security. Indeed, if an adversary has access to a ciphertext C then he can trivially comput $\mathsf{BitDecomp}(C)$ and vice versa. Hence, we can assume without loss of generality that only matrices A and C, but without usage of these operations, are made public.

Then, observe that the matrix C is exactly a matrix of LWE ciphertexts under secret vector $\mathbf{s}$. Hence, finding $\mathbf{s}$ from knowledge of C and A is equivalent to solving search LWE, and deciding if C is a ciphertext or random matrix is equivalent to decision LWE. Consequently, the encryption scheme is secure if the LWE problems are computationally infeasible.

Since LWE is a relatively new and unstudied problem, and since its hardness is difficult to prove directly, we would like to establish a security reduction from some other problem, with an established computational infeasibility. Due to the rather close relationship between LWE and lattices, it is natural to base security on the hardness of certain lattice problems.

In this paper, we look at a *quantum* reduction first published by Regev[2]. For more discussion on the security of LWE based encryption, the interested reader is encouraged to look at [8],[9],[10].

The security reduction by Regev is supported by the following two lemmas, here presented without proofs.

Lemma 3.7. *There exists an efficient algorithm that, given any n-dimensional lattice Λ and $r > 2^{2n}\lambda_n(\Lambda)$, outputs a sample from a distribution within statistical distance $2^{-\Omega(n)}$*

of $D_{\Lambda,r}$.

Lemma 3.8. *Let* $\varepsilon = \varepsilon(n)$ *be a negligible function,* $\alpha = \alpha(n) \in (0,1)$ *be a real number and* $p = p(n) \geq 2$ *be an integer. Assume we have access to an oracle* W *that solves* $\mathsf{LWE}_{p,\Psi_\alpha}$ *given a polynomial number of samples. Then, there exists a constant* $c > 0$ *and an efficient quantum algorithm that, given any* n*-dimensional lattice* Λ*, a number* $r > \sqrt{2}p\eta_\varepsilon(\Lambda)$*, and* n^c *samples from* $D_{\Lambda,r}$*, produces a sample from* $D_{L,r\sqrt{n}/(\alpha p)}$.

With these lemmas, we are finally ready to prove the main security reduction from LWE to DGS.

Theorem 3.9. *Let* $\varepsilon = \varepsilon(n)$ *be a negligible function of* n*,* $p = p(n)$ *some integer, and* $\alpha = \alpha(n) \in (0,1)$ *be such that* $\alpha p > 2\sqrt{n}$*. If there exists an oracle* W *which solves* $\mathsf{LWE}_{p,\Psi_\alpha}$ *given a polynomial number of samples, then there exists an efficient quantum algorithm for solving* $\mathsf{DGS}_{\sqrt{2n}\eta_\varepsilon(\Lambda)/\alpha}$.

Proof. The input to the algorithm is an n-dimensional lattice Λ and a number $r > \sqrt{2n}\eta_\varepsilon(\lambda)\alpha$. Let $r_i = r \cdot (\alpha p/\sqrt{n})^i$. The algorithm starts by producing n^c samples from $D_{\Lambda,r_{3n}}$ where c is the same as in Lemma 3.8. By Lemma 3.7, these samples can be efficiently generated. Now, by Lemma 3.8, we can use the n^c samples from D_{Λ,r_i} to produce samples from $D_{\Lambda,r_{i-1}}$, for $i = 3n, 3n-1, \ldots, 1$. Having done this, we output one of the samples from D_{Λ,r_1} and we are done. □

Assuming this DGS problem is infeasible for the lattice dimension and statistical parameter we use, the theorem tells us that LWE is hard. In order to convince ourselves that this is the case, we could use the following lemma, also by Regev.

Lemma 3.10. *Under the assumptions of theorem 3.9, there exists a polynomial time reduction from* $\mathsf{SIVP}_{\tilde{O}(n/\alpha)}$ *to* $\mathsf{DGS}_{\sqrt{2n}\eta_\varepsilon(\Lambda)/\alpha}$.

The hardness of SIVP_γ depends on the parameter γ. For large values, the problem is easy since the bound on the length of the vectors to be found is very loose. For $\gamma = \tilde{O}(n/\alpha)$, the problem is naturally harder for larger α. This is in agreement with intuition: LWE gets harder when the errors in the equations are allowed to be large. As long as γ is sub-exponential, however, the problem seems to be unsolvable in polynomial time[6].

Chapter 4

Previous bootstrapping techniques

The idea of *bootstrapping* a homomorphic encryption scheme was introduced by Gentry in his doctoral thesis in 2009[1]. The tecnique is used to transform a somewhat homomorphic scheme into a fully homomorphic one by evaluating the decryption algorithm homomorphically, thereby reducing the error. Naturally, the procedure requires the evaluator to have access to an encryption of the secret key, so security of the scheme relies on the assumption that knowing such an encryption isn't enough to break the scheme. This seems like a reasonable assumption in most cases, as the encrypted secret key is just another ciphertext, but there exist situations where this is not the case[11]. We can avoid this potential weakness, however, by encrypting the secret key in a slightly different encryption scheme than the one it is itself a key for.

4.1 Homomorphic NAND and refreshing

In order to get a sense of the bootstrapping of LWE-based encryption, we take a look at FHEW, a homomorphic encryption scheme proposed by Ducas and Micciancio[4]. For the interested reader, information on other ideas for bootstrapping of LWE based schemes are availible in [3],[12],[10].

The idea is to take binary messages $\mu_i \in \{0, 1\}$ and let the error distribution χ be such that the error of a ciphertext is less than $q/16$, except with negligible probability. Recall that this can be achieved by using a σ-subgaussian distribution and scaling the sample by $\frac{q}{4}$, for $\sigma \leq 1/\left(\sqrt{32\log(2)(\lambda+1)}\right)$ and sufficiently large λ.

To accomplish this, we use a slight variation of the scheme presented in the previous chapter. The LWE problem is considered over some ring $\mathbb{Z}_q$, and the message space is still $\mathbb{B}$. The encryption function is then given by

$$E(\mu) = \left(\mathbf{a}, \mathbf{a}\cdot\mathbf{s} + \frac{\mu q}{2} + \varepsilon\right) \in \mathbb{Z}_q^{n+1}.$$

Within this framework, we can compute the NAND of two ciphertexts $(\mathbf{a}_i, b_i) = \mathsf{Enc}(\mu_i)$ as follows:

$$(\mathbf{a}, b) = \mathsf{HomNAND}((\mathbf{a}_1, b_1), (\mathbf{a}_2, b_2)) = \left(-\mathbf{a}_1 - \mathbf{a}_2, \frac{5q}{8} - b_1 - b_2\right).$$

This resulting ciphertext then satisfies

$$\begin{aligned} b - \mathbf{a}\cdot\mathbf{s} &= \frac{5q}{8} - \mathbf{a}_1\cdot\mathbf{s} - \frac{\mu_1 q}{2} - \varepsilon_1 - \mathbf{a}_2\cdot\mathbf{s} - \frac{\mu_2 q}{2} - \varepsilon_2 + (\mathbf{a}_1 + \mathbf{a}_2)\cdot\mathbf{s} \\ &= \frac{5q}{8} - \frac{q}{2}(\mu_1 + \mu_2) - (\varepsilon_1 + \varepsilon_2) \end{aligned}$$

which in turn gives

$$b - \mathbf{a}\cdot\mathbf{s} - \frac{q}{2}(1 - \mu_1\mu_2) = \frac{5q}{8} - \frac{q}{2} - (\varepsilon_1 + \varepsilon_2) = \frac{q}{8} - (\varepsilon_1 + \varepsilon_2).$$

Hence, $(\mathbf{a}, b)$ is an encryption of $1 - \mu_1\mu_2 = \mathsf{NAND}(\mu_1, \mu_2)$ with error bound

$$\mathsf{Err}(\mathbf{a}, b) \leq \left|\frac{q}{8} - (\varepsilon_1 + \varepsilon_2)\right|_\infty \leq \frac{q}{8} + \frac{q}{16} + \frac{q}{16} = \frac{q}{4}.$$

Since we need an error bound of $\frac{q}{16}$ in the input to $\mathsf{HomNAND}$, we need to refresh, or bootstrap, this ciphertext before we can do another NAND-operation. That is, we need another operation which takes a ciphertext with error bounded by $\frac{q}{4}$ and produces a new ciphertext, encrypting the same message, with error bounded by $\frac{q}{16}$.

In order to implement this refreshing function, the same principle as in Gentry's original bootstrapping procedure is used. Namely, one uses an encryption of the secret key $\mathbf{s}$ and homomorphically evaluates the decryption calculation on the noisy ciphertext and the encrypted key. Here, however the key is not generally encrypted under the same encryption scheme as the one in which we perform homomorphic operations.

In this case, we compute

$$\lfloor 2(b - \mathbf{a}\cdot E(\mathbf{s}))/q \rceil \bmod 2$$

where $E(\cdot)$ denotes encryption under this scheme.

We will not go into detail on this, which is just meant to illustrate the idea before we start with the detailed procedure in the next chapter.

Chapter 5

On improving the bootstrapping

In order to make the bootstrapping step more efficient, we will allow approximations in the calculations. We go back to thinking of ciphertexts as TLWE samples on the torus. Naturally, these approximations introduce a greater amount of noise, but they also improve running time and memory requirements of the bootstrapping procedure[5]. In this chapter we take a closer look on TLWE samples and introduce the TGSW sample which is essentially a matrix of TLWE samples. We also introduce an algorithm which decomposes a TLWE sample into a short vector of integer polynomials. Using these tools, we will introduce a new homomorphic product which we will then use between the encrypted secret key and a noisy ciphertext. We finish the chapter with some analysis of this bootstrapping procedure as well as defining the operations of some of the basic boolean gates.

5.1 A new homomorphic product

Before we can look at the improved bootstrapping procedure, we have to define a new, external product between ciphertexts. To facilitate this, we first introduce a *gadget decomposition*. A gadget is a tool we use to make the algorithm work, in this case a matrix on

this form:

$$\mathbf{h} = \left(\begin{array}{c|c|c} 1/B_g & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1/B_g^l & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & 1/B_g \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/B_g^l \end{array}\right) \in \mathcal{M}_{p,k+1}(\mathbb{T}_N[X]). \tag{5.1}$$

When we run a decomposition algorithm with gadget $\mathbf{h}$ and TLWE sample $\mathbf{v}$ as input, the output should be a short vector $\mathbf{u}$ such that $\mathbf{u} \cdot \mathbf{h} \approx \mathbf{v}$. We allow approximation, and important properties of the decomposition algorithm are its *quality* and *precision*. These values limit the output's length and distance from the exact, respectively.

Definition 5.1 (Approximate Gadget Decomposition)**.** Let $\mathbf{h} \in \mathcal{M}_{p,k+1}(\mathbb{T}_N[X])$. We say that $\mathsf{Dec}_{\mathbf{h},\beta,\epsilon}(\mathbf{v})$ is a *valid decomposition algorithm on the gadget* $\mathbf{h}$ with quality β and precision ϵ if, for any TLWE sample $\mathbf{v} \in \mathbb{T}_N[X]^{k+1}$, it outputs a vector $\mathbf{u} \in \mathcal{R}^{(k+1)l}$, satisfying the following:

- $\|\mathbf{u}\|_\infty \leq \beta$,
- $\|\mathbf{u} \cdot \mathbf{h} - \mathbf{v}\| \leq \epsilon$, and
- if $\mathbf{v}$ is uniformly distributed in $\mathbb{T}_N[X]^{k+1}$, then $\mathsf{E}[\mathbf{u} \cdot \mathbf{h} - \mathbf{v}] = 0$.

As mentioned, in this paper we will always use a gadget on the form presented in equation (5.1). Thus, $\mathbf{h}$ will always refer to this in the rest of the text. Now we will present an algorithm which we then will show is a valid decomposition algorithm on $\mathbf{h}$, given the right parameters.

Algorithm 5.2 Gadget decomposition of a TLWE sample

Input: a TLWE sample $(a, b) = (a_1, \ldots, a_k, b = a_{k+1}) \in \mathbb{T}_N[X]^{k+1}$

Output: A vector $[e_{1,1}, \ldots, e_{k+1,l}] \in \mathcal{R}^{(k+1)l}$

1: For each a_i, let $a_{i,j} \in \mathbb{T}$ be such that $a_i = \sum_{j=0}^{N-1} a_{i,j} X^j$. Set $\bar{a}_{i,j}$ equal to the closest multiple of $\frac{1}{B_g^l}$ to $a_{i,j}$.
2: Decompose each $\bar{a}_{i,j}$ uniquely as $\sum_{p=1}^{l} \bar{a}_{i,j,p} \frac{1}{B_g^p}$ where each $\bar{a}_{i,j,p} \in [-B_g/2, B_g/2)$.
3: **for** $i = 1$ **to** $k + 1$ **do**
4: **for** $p = 1$ **to** l **do**
5: $e_{i,p} = \sum_{j=0}^{N-1} \bar{a}_{i,j,p} X^j$
6: **return** $(e_{i,p})_{i,p}$

Lemma 5.3. *Let $l, B_g \in \mathbb{N}$, $\beta = B_g/2$, and $\epsilon = 1/(2B_g^l)$. Then, Algorithm 5.2 is a valid $\mathsf{Dec}_{\mathbf{h},\beta,\epsilon}$ on the gadget $\mathbf{h}$.*

Proof. Let $\mathbf{v} = (a_1, \ldots, a_k, b = a_{k+1}) \in \mathbb{T}_N[X]^{k+1}$ be a TLWE sample given as input to algorithm 5.2, with corresponding output $\mathbf{u} = [e_{1,1}, \ldots, e_{k+1,l}]$. By construction, each coefficient of $e_{i,j}$ is at most $B_g/2$ (in absolute value), so $\|\mathbf{u}\| \leq B_g/2 = \beta$.

Let $\epsilon_{\text{dec}} = \mathbf{u} \cdot \mathbf{h} - \mathbf{v}$. Then, $\epsilon_{\text{dec}i,j} = \sum_{p=0}^{l} e_{i,p} \cdot \frac{1}{B_g^p} - a_{i,j} = \bar{a}_{i,j} - a_{i,j}$, for all $i \in [1, k+1]$ and $j \in [1, l]$. Since $\bar{a}_{i,j}$ is defined as the multiple of $1/B_g^l$ closest to $a_{i,j}$, surely $|\bar{a}_{i,j} - a_{i,j}| \leq 1/(2B_g^l) = \epsilon$. Thus, ϵ_{dec} has a concentrated distribution when $\mathbf{v}$ is uniformly distributed. To verify it is zero-centered, let f be the function from $\mathbb{T}$ to $\mathbb{T}$ which rounds an element x to its closest multiple of $\frac{1}{B_g^l}$ and let g be defined $g(x) = 2f(x) - x$ on the torus. If $a_{i,j}$ is uniformly distributed, then

$$\mathsf{E}(\epsilon_{\text{dec}i,j}) = \mathsf{E}(a_{i,j} - f(a_{i,j})) = \mathsf{E}(g(a_{i,j}) - f(g(a_{i,j}))) = \mathsf{E}(f(a_{i,j}) - a_{i,j}) = -\mathsf{E}(\epsilon_{\text{dec}i,j}).$$

Hence, the expectation of ϵ_{dec} is 0. □

We now know, thanks to lemma 5.3, that we are able to decompose a TLWE sample into a short vector, with a certain degree of precicion. The decomposition makes calculating on encrypted data more efficient, while the approximation introduces a small amount of noise. The external product we are pursuing uses both the decomposition of an TLWE sample and another type of sample, called TGSW. A TGSW sample is, loosely speaking, a matrix where each row is a TLWE sample. More precisely, it is the sum of a multiple of $\mathbf{h}$ and a matrix, each row of which is a homogeneous TLWE sample.

Definition 5.4 (TGSW sample)**.** Let l and $k \geq 1$ be two integers and $\mathbf{h}$ the gadget in equation 5.1. We say that $\mathbf{C} \in \mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X])$ is a fresh TGSW sample of $\mu \in \mathcal{R}/\mathbf{h}^{\perp}$ if $\mathbf{C} = \mathbf{Z} + \mu \cdot \mathbf{h}$, where all rows of $\mathbf{Z} \in \mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X])$ are homogeneous TLWE samples with equal noise parameter. The polynomial μ is the message of $\mathbf{C}$, denoted $\mathsf{msg}(\mathbf{C})$, and the noise parameter of $\mathbf{C}$ is said to be the same as the noise parameter of the TLWE samples in $\mathbf{Z}$.

Since a TGSW sample is a matrix where each row is a TLWE sample, it inherits some useful properties of these TLWE samples. As mentioned, for example, it has the same noise parameter as the TLWE samples it contains. Unsurprisingly, TGSW samples also permit the same linear operations as TLWE. In particular, the phase and error are both linear when defined as vectors of phases and errors of TLWE samples.

Definition 5.5. Let $A \in \mathcal{M}_{(k+1)l,k+1}(\mathbb{T}_N[X])$ be a TGSW sample. The phase of A, denoted $\varphi_{\mathsf{s}}(A)$, is defined as the vector of phases of each of the $(k+1)l$ TLWE samples in A. Correspondingly, the error $\mathsf{Err}(A)$ is a vector of the errors of the TLWE samples in A.

We are now equipped with all we need to introduce and analyze the operation we wish to use as a homomorphic product between a noisy ciphertext and an encrypted key. Recall that the decryption operation involves a product between ciphertext and key, and hence this is what we perform homomorphically in the bootstrapping.

Definition 5.6 (External product)**.** We define the product $\boxdot$ as

$$A \boxdot \mathbf{b} = \mathsf{Dec}_{\mathbf{h},\beta,\epsilon}(\mathbf{b}) \cdot A.$$

Theorem 5.7 (Worst-case external product)**.** *Let A be a valid TGSW sample of a message μ_A and let $\mathbf{b}$ be a valid TLWE sample of a message μ_b. Then $A \boxdot \mathbf{b}$ is a TLWE sample for message $\mu_A \cdot \mu_b$. and $\|\mathsf{Err}(A \boxdot \mathbf{b})\|_\infty \leq (k+1)lN\beta\|\mathsf{Err}(A)\|_\infty + \|\mu_A\|_1(1+kN)\epsilon + \|\mu_A\|_1\|\mathsf{Err}(\mathbf{b})\|_\infty$. If $\|\mathsf{Err}(A \boxdot \mathbf{b})\|_\infty \leq 1/4$, then $A \boxdot \mathbf{b}$ is a valid TLWE sample.*

Proof. Since $A = \mathsf{TGSW}(\mu_A)$, we have $A = Z_A + \mu_A \cdot \mathbf{h}$ where Z_A is a TGSW encryption of 0. Similarly, $\mathbf{b} = \mathbf{z_b} + (\mathbf{0}, \mu_\mathbf{b})$, where $\mathbf{z_b}$ is a TLWE encryption of 0.

Let $\mathbf{u} = \mathsf{Dec}_{\mathbf{h},\beta,\epsilon}(\mathbf{b}) \in \mathcal{R}^{(k+1)l}$. Then

$$A \boxdot \mathbf{b} = \mathbf{u} \cdot A = \mathbf{u} \cdot Z_A + \mu_A \cdot (\mathbf{u} \cdot \mathbf{h}).$$

It follows from definition 5.1 that $\mathbf{u} \cdot \mathbf{h} = \mathbf{b} + \epsilon_{\mathsf{dec}}$ for some $\|\epsilon_{\mathsf{dec}}\|_\infty \leq \epsilon$. Thus,

$$A \boxdot \mathbf{b} = \mathbf{u} \cdot Z_A + \mu_A \cdot (\mathbf{b} + \epsilon_{\mathsf{dec}}) = \mathbf{u} \cdot Z_A + \mu_A \cdot \epsilon_{\mathsf{dec}} + \mu_A \cdot \mathbf{z_b} + (\mathbf{0}, \mu_A \cdot \mu_b).$$

The phase is then given by

$$\varphi_\mathbf{s}(A \boxdot \mathbf{b}) = \mathbf{u} \cdot \mathsf{Err}(A) + \mu_A \cdot \varphi_\mathbf{s}(\epsilon_{\mathsf{dec}}) + \mu_A \cdot \mathsf{Err}(\mathbf{b}) + \mu_A\mu_\mathbf{b}.$$

Taking the expectation, we get $\mathsf{msg}(A \boxdot \mathbf{b}) = 0 + 0 + 0 + \mu_A\mu_\mathbf{b}$, so $\mathsf{Err}(A \boxdot \mathbf{b}) = \varphi_\mathbf{s}(A \boxdot \mathbf{b}) - \mu_A\mu_\mathbf{b}$.

Then, the lipschitzianity of the phase implies

$$\begin{aligned}
\|\mathsf{Err}(A \boxdot \mathbf{b})\|_\infty &= \|\varphi_\mathbf{s}(A \boxdot \mathbf{b}) - \mu_A\mu_\mathbf{b}\|_\infty \\
&\leq \|\mathbf{u} \cdot \mathsf{Err}(A)\|_\infty + \|\mu_A \cdot \varphi_\mathbf{s}(\epsilon_{\mathsf{dec}})\|_\infty + \|\mu_A \cdot \mathsf{Err}(\mathbf{b})\|_\infty \\
&\leq (k+1)lN\beta\eta_A + \|\mu_A\|_1(1+kN)\|\epsilon_{\mathsf{dec}}\|_\infty + \|\mu_A\|_1\eta_\mathbf{b}.
\end{aligned}$$

□

The theorem gives us an upper bound on the amplitude of the error after performing the external product, i.e a worst case estimate of the error. The next corollary tells us more about the realistic case by giving a bound on the variance of the error.

Corollary 5.8 (Average-case external product)**.** *Under assumptions in Theorem 5.7 and assuming all error coefficients are independent, it holds that*

$$\mathsf{Var}(\mathsf{Err}(A \boxdot \mathbf{b})) \leq (k+1)lN\beta^2\mathsf{Var}(\mathsf{Err}(A)) + (1+kN)\|\mu_A\|_2^2\epsilon^2 + \|\mu_A\|_2^2\mathsf{Var}(\mathsf{Err}(\mathbf{b})).$$

Proof. Let $\vartheta_A = \mathsf{Var}(\mathsf{Err}(A \boxdot \mathbf{b})) = \mathsf{Var}(\varphi_{\mathbf{s}}(Z_A))$ and $\vartheta_b = \mathsf{Var}(\mathsf{Err}(\mathbf{b})) = \mathsf{Var}(\varphi_{\mathbf{s}}(\mathbf{z_b}))$. Similarliy as in the proof of theorem 5.7, we get

$$\mathsf{Err}(A \boxdot \mathbf{b}) = \mathbf{u} \cdot \mathsf{Err}(A) + \mu_A \cdot \varphi_{\mathbf{s}}(\epsilon_{\mathsf{dec}}) + \mu_A \cdot \mathsf{Err}(\mathbf{b}).$$

Lipschitzianity and independence of errors gives

$$\begin{aligned}\mathsf{Var}(\mathsf{Err}(A \boxdot \mathbf{b})) &\leq \mathsf{Var}(\mathbf{u} \cdot \mathsf{Err}(A)) + \mathsf{Var}(\mu_A \cdot \varphi(\epsilon_{\mathsf{dec}})) + \mathsf{Var}(\mu_A \cdot \mathsf{Err}(\mathbf{b})) \\ &\leq (k+1)lN\beta^2\vartheta_A + (1+kN)\|\mu_A\|_2^2\epsilon^2 + \|\mu_A\|_2^2\vartheta_{\mathbf{b}}.\end{aligned}$$

□

5.2 New bootstrapping procedure

In this section, we wish to use the external product to speed up the bootstrapping presented earlier. We also present key switching, a procedure to optimize further, and decrease the size of the bootstrapping key.

The first techique we present is to extract an LWE sample from a TLWE sample. We do this by first making a vector of the coefficients of the TLWE key, known as $\mathsf{KeyExtract}$. We then extract the coefficients of the polynomials in the TLWE sample.

Definition 5.9 (TLWE extraction)**.** Let $(\mathbf{a}'', b'')$ be a $\mathrm{TLWE}_{\mathbf{s}''}(\mu)$ sample with key $\mathbf{s}'' = (s_1'', \ldots, s_k'') \in \mathcal{R}^k$. We define $\mathsf{KeyExtract}(\mathbf{s}'')$ as the integer vector

$$\mathbf{s}' = \big(\mathsf{coefs}(s_1''(X)), \ldots, \mathsf{coefs}(s_k''(X))\big) \in \mathbb{Z}^{kN}$$

where $\mathsf{coefs}(\cdot)$ extracts a vector of polynomial coefficients. Further, we define $\mathsf{SampleExtract}(\mathbf{a}'', b'')$ as the $\mathrm{LWE}_{\mathbf{s}'}$ sample $(\mathbf{a}', b') \in \mathbb{T}^{kN+1}$ where

$$\mathbf{a}' = \big(\mathsf{coefs}(a_1''(1/X)), \ldots, \mathsf{coefs}(a_k''(1/X))\big)$$

and $b' = b_0''$ is the constant term of b''.

Next follows a technical lemma about the properties of key- and sample extraction.

Lemma 5.10. *Let* $\mathbf{s}' = \mathsf{KeyExtract}(\mathbf{s}'')$ *and* $(\mathbf{a}', b') = \mathsf{SampleExtract}(\mathbf{a}'', b'')$*. Then*

- $\varphi_{\mathbf{s}'}(\mathbf{a}', b')$ *equals the constant term of* $\varphi_{\mathbf{s}''}(\mathbf{a}'', b'')$*,*

- $\mathsf{msg}(\mathbf{a}', b')$ *equals the constant term of* $\mathsf{msg}(\mathbf{a}'', b'')$,
- $\|\mathsf{Err}(\mathbf{a}', b')\|_\infty \leq \|\mathsf{Err}(\mathbf{a}'', b'')\|_\infty$, *and*
- $\mathsf{Var}(\mathsf{Err}(\mathbf{a}', b')) \leq \mathsf{Var}(\mathsf{Err}(\mathbf{a}'', b''))$.

Definition 5.11. Let $\mathbf{s}' \in \{0,1\}^{n'}$, $\mathbf{s} \in \{0,1\}^n$, $\gamma \in \mathbb{R}$, $t \in \mathbb{N}$. A *key switching secret* $\mathsf{KS}_{\mathbf{s}' \to \mathbf{s}, \gamma, t}$ is a sequence of fresh LWE samples $\mathsf{KS}_{i,j} \in \mathsf{LWE}_{\mathbf{s}, \gamma}(s'_i \cdot 2^{-j})$ for $i \in [1, n']$, $j \in [1, t]$.

Algorithm 5.12 Key-switching procedure

Input: A LWE sample $(\mathbf{a}' = (a'_1, \ldots, a'_{n'}), b') \in \mathsf{LWE}_{\mathbf{s}'}(\mu)$, a switching key $\mathsf{KS}_{\mathbf{s}' \to \mathbf{s}}$ where $\mathbf{s}' \in \{0,1\}^{n'}$, $\mathbf{s} \in \{0,1\}^n$ and a precision parameter $t \in \mathbb{N}$.

Output: a LWE sample $\mathsf{LWE}_{\mathbf{s}}(\mu)$.

1: **for** $i = 1$ **to** n' **do**
2: Let $\bar{a}'_i$ be the closest multiple of $\frac{1}{2^t}$ to a'_i.
3: Binary decompose each $\bar{a}'_i = \sum_{j=1}^{l} a_{i,j} \cdot 2^{-j}$ such that $a_{i,j} \in \{0,1\}$
4: **return** $(\mathbf{0}, b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \cdot \mathsf{KS}_{i,j}$

Lemma 5.13 (Key switching)**.** *Let* $(\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}'}(\mu)$*, where* $\mathbf{s}' \in \{0,1\}^{n'}$ *and let* $\eta' = \|\mathsf{Err}(\mathbf{a}', b')\|_\infty$*. Then, algorithm 5.12 with* $(\mathbf{a}', b')$ *and a keyswitching key* $\mathsf{KS}_{\mathbf{s}' \to \mathbf{s}, \gamma, t}$ *as input, outputs an* LWE *sample* $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}(\mu)$ *with* $\|\mathsf{Err}(\mathbf{a}, b)\|_\infty \leq \eta' + n' t \gamma + n' 2^{-(t+1)}$.

Proof. From the assumptions in the lemma, we get

$$\begin{aligned}
\varphi_{\mathbf{s}}(\mathbf{a}, b) &= \varphi_{\mathbf{s}}(\mathbf{0}, b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \varphi_{\mathbf{s}'}(\mathsf{KS}_{i,j}) = b' - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} (2^{-j} s'_i + \mathsf{Err}(\mathsf{KS}_{i,j})) \\
&= b' - \sum_{i=1}^{n'} \bar{a}'_i s'_i - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) = b' - \sum_{i=1}^{n'} a'_i s'_i - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{n'} (a'_i - \bar{a}'_i) s'_i \\
&= \varphi_{\mathbf{s}'}(\mathbf{a}', b') - \sum_{i=1}^{n'} \sum_{j=1}^{t} a_{i,j} \mathsf{Err}(\mathsf{KS}_{i,j}) + \sum_{i=1}^{n'} (a'_i - \bar{a}'_i) s'_i.
\end{aligned}$$

Recall that $\mathsf{msg}(\mathbf{a}, b) = \mathsf{E}(\varphi(\mathbf{a}, b))$. By algorithm 5.12, line 2 we get that $|a'_i - \bar{a}'_i| < 2^{-(t+1)}$ and that the expected difference is 0. Hence, the expected value of the last sum on the right-hand side is 0. Consequently, $\mathsf{msg}(\mathbf{a}, b) = \mathsf{msg}(\mathbf{a}', b')$. Further

$$\|\mathsf{Err}(\mathbf{a}, b)\|_\infty = \|\varphi(\mathbf{a}, b) - \mathsf{msg}(\mathbf{a}, b)\|_\infty \leq \eta' + n' t \gamma + n' 2^{-(t+1)}.$$

□

Definition 5.14 (Bootstrapping key)**.** Let $\mathbf{s} \in \mathbb{B}^n$, $\mathbf{s}'' \in \mathbb{B}_N[X]^k$ and α be a noise parameter. A *bootstrapping key* $\mathsf{BK}_{\mathbf{s} \to \mathbf{s}'', \alpha}$ is a sequence of n TGSW samples $\{\mathsf{BK}_i\}_{i=1}^n$ where $\mathsf{BK}_i \in \mathsf{TGSW}_{\mathbf{s}'', \alpha}(s_i)$.

Algorithm 5.15 Bootstrapping procedure

Input: A LWE sample $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s},\eta}(\mu)$, a bootstrapping key $\mathsf{BK}_{\mathbf{s}\to\mathbf{s}'',\alpha}$, a keyswitch key $\mathsf{KS}_{\mathbf{s}'\to\mathbf{s},\gamma}$ where $\mathbf{s}' = \mathsf{KeyExtract}(\mathbf{s}'')$, two fixed messages $\mu_0, \mu_1 \in \mathbb{T}$

Output: A LWE sample $\mathsf{LWE}_{\mathbf{s},\nu}\left(\mu_0 \text{ if } \varphi_{\mathbf{s}}(\mathbf{a}, b) \in \left(-\frac{1}{4}, \frac{1}{4}\right); \mu_1 \text{ otherwise}\right)$

1: Let $\bar{\mu} = \frac{\mu_1+\mu_0}{2}$ and $\bar{\mu}' = \mu_0 - \bar{\mu}$

2: Let $\bar{b} = \lfloor 2Nb \rceil$ and $\bar{a}_i = \lfloor 2Na_i \rceil$ for each $i \in \{1, \dots, n\}$

3: Let $\mathsf{testv} \coloneqq (1 + X + \cdots + X^{N-1}) \cdot X^{-\frac{2N}{4}} \cdot \bar{\mu}' \in \mathbb{T}_N[X]$

4: $\mathsf{ACC} \leftarrow \left(X^{\bar{b}} \cdot (\mathbf{0}, \mathsf{testv})\right) \in \mathbb{T}_N[X]^{k+1}$

5: **for** i=1 **to** n **do**

6: $\quad \mathsf{ACC} \leftarrow [\mathbf{h} + (X^{-\bar{a}_i} - 1) \cdot \mathsf{BK}_i] \boxdot \mathsf{ACC}$

7: Let $\mathbf{u} \coloneqq (\mathbf{0}, \bar{\mu}) + \mathsf{SampleExtract}(\mathsf{ACC})$

8: **return** $\mathsf{KeySwitch}_{\mathsf{KS}}(\mathbf{u})$

Theorem 5.16 (Bootstrapping theorem)**.** *Let* $\mathbf{h}$ *and* $\mathsf{Dec}_{\mathbf{h},\varepsilon,\beta}$ *be the gadget and decomposition function as in definition 5.1. Let* $\mathbf{s} \in \mathbb{B}_N[X]^k$ *and let* α, γ *be noise amplitudes. Finally, let* $\mathsf{BK} = \mathsf{BK}_{\mathbf{s}\to\mathbf{s}'',\alpha}$ *be a bootstrapping key and let* $\mathbf{s}' = \mathsf{KeyExtract}(\mathbf{s}'') \in \mathbb{B}^{kN}$ *and* $\mathsf{KS} = \mathsf{KS}_{\mathbf{s}'\to\mathbf{s},\gamma,t}$ *be a keyswitching secret.*

Given $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}(\mu)$ *for* $\mu \in \mathbb{T}$*, two fixed messages* μ_0, μ_1*, algorithm 5.15 outputs a sample in* $\mathsf{LWE}_{\mathbf{s}}(\mu')$ *such that*

$$\mu' = \begin{cases} \mu_0 & \text{if } |\varphi_{\mathbf{s}}(\mathbf{a}, b)| < \frac{1}{4} - \delta \\ \mu_1 & \text{if } |\varphi_{\mathbf{s}}(\mathbf{a}, b)| > \frac{1}{4} + \delta \end{cases}$$

where δ *is the cumulated rounding error from line 2 in the algorithm. (Note that* $\delta = 0$ *if each coefficient of* $(\mathbf{a}, b)$ *is a multiple of* $1/(2N)$ *and* δ *is at most* $(n+1)/(4N)$*.)*

Further, let $\mathbf{v}$ *be the output of the Bootstrapping Procedure. Then*

$$\|\mathsf{Err}(\mathbf{v})\|_\infty \leq 2n(k+1)l\beta N\alpha + kNt\gamma + n(1+kN)\varepsilon + kN2^{-(t+1)}.$$

Proof. Clearly, after line 1 of the algorithm, $\bar{\mu} + \bar{\mu}' = \mu_0$ and $\bar{\mu} - \bar{\mu}' = \mu_1$. By defining $\bar{\varphi} = \bar{b} - \sum_{i=1}^n \bar{a}_i s_i \bmod 2N$, we get

$$\left|\varphi - \frac{\bar{\varphi}}{2N}\right| = b - \frac{\lfloor 2Nb \rceil}{2N} + \sum_{i=1}^n \left(a_i - \frac{\lfloor 2Na_i \rceil}{2N}\right) s_i \leq \frac{1}{4N} + \sum_{i=1}^n \frac{1}{4N} \leq \frac{n+1}{4N}.$$

If each coefficient of $(\mathbf{a}, b)$ is a multiple of $1/(2N)$, then $\varphi = \bar{\varphi}/(2N)$. In any case, $|\varphi - \bar{\varphi}/(2N)| < \delta$.

Further, the test vector testv is defined such that for all $p \in [0, 2N]$, the constant term of $X^p \cdot \mathsf{testv}$ is $\bar{\mu}'$ if $p \in (-N/2, N/2)$ and $-\bar{\mu}'$ otherwise.

Now, consider the loop on lines 5 and 6. At the start of the loop, ACC contains a trivial ciphertext, so $(\mathsf{ACC}) = \left(X^{\bar{b}} \cdot \mathsf{testv}\right)$ and $\|\mathsf{Err}(\mathsf{ACC})\|_\infty = 0$. During iteration i, let $A_i = \mathbf{h} + \left(x^{-\bar{a}_i} - 1\right) \cdot \mathsf{BK}_i$. This is a TGSW-sample of a message $X^{-\bar{a}_i s_i}$ with noise $\|\mathsf{Err}(A_i)\|_\infty \leq 2\|\mathsf{Err}(\mathsf{BK}_i)\|_\infty$. Thus

$$\begin{aligned}\mathsf{msg}(\mathsf{ACC}_i) &= \mathsf{msg}(A_i \boxdot \mathsf{ACC}_{i-1}) = \mathsf{msg}(A_i) \cdot \mathsf{msg}(\mathsf{ACC}_{i-1}) \\ &= X^{-\bar{a}_i s_i} \cdot \left(X^{b - \sum_{j=1}^{i-1} \bar{a}_j s_j} \cdot \mathsf{testv}\right),\end{aligned}$$

and

$$\begin{aligned}\|\mathsf{Err}(\mathsf{ACC}_i)\|_\infty &\leq (k+1)lN\beta\|\mathsf{Err}(A_i)\|_\infty + \|\mathsf{msg}(A_i)\|_1(1+kN)\varepsilon + \|\mathsf{msg}(A_i)\|_1\|\mathsf{Err}(\mathsf{ACC}_{i-1})\|_\infty \\ &\leq (k+1)lN\beta 2\|\mathsf{Err}(\mathsf{BK}_i)\|_\infty + (1+kN)\varepsilon + \|\mathsf{Err}(\mathsf{ACC}_{i-1})\|_\infty.\end{aligned}$$

By induction on i, this proves that after step n, the following holds:

$$\begin{aligned}\mathsf{msg}(\mathsf{ACC}) &= X^{b - \sum_{j=1}^{n} \bar{a}_j s_j} \cdot \mathsf{testv} \\ \|\mathsf{Err}(\mathsf{ACC}\|_\infty &\leq \sum_{j=1}^{n} \left(2(k+1)lN\beta\|\mathsf{Err}(\mathsf{BK}_j)\|_\infty + (1+kN)\varepsilon\right).\end{aligned}$$

After line 7, the message of $\mathbf{u}$ is equal to the constant term of of the message of ACC, that is $X^{\bar{\varphi}} \cdot \mathsf{testv}$. Recall that this is $\bar{\mu}'$ if $\bar{\varphi} \in [-N/2, N/2]$ and $-\bar{\mu}'$ otherwise. Hence, if $\varphi_{\mathbf{s}}(\mathbf{a}, b) \in [-1/4 + \delta, 1/4 - \delta)$, then $\mathsf{msg}(u) = \bar{\mu}'$ and if not, then $\mathsf{msg}(u) = -\bar{\mu}'$.

Finally, since $\mathsf{KeySwitch}$ doesn't change the message and $\mathsf{SampleExtract}$ doesn't add to the noise,

$$\begin{aligned}\mathsf{msg}(v) &= \mathsf{msg}(u) \\ \|\mathsf{Err}(\mathbf{v})\|_\infty &\leq \|\mathsf{Err}(\mathbf{u})\|_\infty + kNt\gamma + kN2^{-(t+1)}.\end{aligned}$$

□

The bottom line of the theorem is that we can take an LWE ciphertext with a message in $\mathbb{T}$ and a bootstrapping key, and using the procedure of algorithm 5.15 we obtain a new ciphertext with fixed error bound. In the LWE ciphertext we interpret a message greater than $\frac{1}{4}$ (in absolute value) as 1 and a message less than $\frac{1}{4}$ as 0, allowing us to evaluate boolean operations.

5.3 Circuits and gates

Now we will define simple operations to evaluate several of the most common gates homomorphically. For ciphertexts $\mathbf{c}_1, \mathbf{c}_2$, each with message either 0 or 1/4 we define:

- $\mathsf{HomNOT}(\mathbf{c}_1) = (\mathbf{0}, \frac{1}{4}) - \mathbf{c}_1$

- $\mathsf{HomAND}(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{Bootstrap}\left((\mathbf{0}, -\frac{1}{8}) + \mathbf{c}_1 + \mathbf{c}_2\right)$
- $\mathsf{HomNAND}(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{Bootstrap}\left((\mathbf{0}, \frac{5}{8}) - \mathbf{c}_1 - \mathbf{c}_2\right)$
- $\mathsf{HomOR}(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{Bootstrap}\left((\mathbf{0}, \frac{1}{8}) + \mathbf{c}_1 + \mathbf{c}_2\right)$
- $\mathsf{HomXOR}(\mathbf{c}_1, \mathbf{c}_2) = \mathsf{Bootstrap}\left(2 \cdot (\mathbf{c}_1 - \mathbf{c}_2)\right)$.

Here, $\mathsf{Bootstrap}(\mathbf{c})$ denotes the output of the bootstrapping procedure of algorithm 5.15 with input $\mathbf{c}$ and with $\mu_0 = 0$, $\mu_1 = \frac{1}{4}$. We illustrate the correctness by considering the $\mathsf{HomNAND}$ procedure. If both $\mathbf{c}_1$ and $\mathbf{c}_2$ encode 0, then $\tilde{\mathbf{c}} = \mathsf{Bootstrap}\left((\mathbf{0}, \frac{1}{8}) - \mathbf{c}_1 - \mathbf{c}_2\right)$ will have message $\frac{5}{8}$, and if one of the inputs have message 0 and the other $\frac{1}{4}$, then $\tilde{\mathbf{c}}$ has message $\frac{3}{8}$. If both of the inputs have messages $\frac{1}{4}$, then the result will have message $\frac{1}{8}$. Thus, since the error is smaller than $\frac{1}{8}$, $|\varphi(\tilde{\mathbf{c}})| > \frac{1}{4}$ if and only if $\mathsf{NAND}(\mathsf{msg}(\mathbf{c}_1), \mathsf{msg}(\mathbf{c}_2)) = 1$. Analog reasoning demonstrates the correctness of the rest of the homomorphic operations.

With this structure, any boolean circuit could be turned into one using only the gates mentioned above, and then evaluated homomorphically. The bootstrapping procedure is the only potential bottleneck for the efficiency of evaluating such a circuit.

An implementation of the scheme can be found at `https://github.com/tfhe/tfhe`[1]. The implementation makes use of a fast fourier transform (FFT)-alogrithm, such at FFTW[13]. Using this implementation, a bootstrapping takes less than 0.1 seconds on a standard personal computer.

[1]Made by the authors of [5]. Requires a C++11 compiler as well as a fast fourier transform (FFT) processor.

Bibliography

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

[2] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[3] Craig Gentry, Amit Sahai, and Brent Waters. *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based*, pages 75–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] Léo Ducas and Daniele Micciancio. *FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second*, pages 617–640. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[5] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 3–33, 2016.

[6] Daniele Micciancio. Efficient reductions among lattice problems. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 84–93. Society for Industrial and Applied Mathematics, 2008.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/2011/277`.

[8] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584. ACM, 2013.

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

[10] Martin R Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, 74(2):325–354, 2015.

[11] John Black, Phillip Rogaway, and Thomas Shrimpton. *Encryption-Scheme Security in the Presence of Key-Dependent Messages*, pages 62–75. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[12] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *International Cryptology Conference*, pages 297–314. Springer, 2014.

[13] Matteo Frigo and Steven G Johnson. The fastest fourier transform in the west. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE, 1997.