A `CONCRETE` approach to torus fully homomorphic encryption

Maria Ferrara and Antonio Tortora*

Dipartimento di Matematica e Fisica, Università della Campania "Luigi Vanvitelli"

viale Lincoln, 5 - 81100 - Caserta, Italy

E-mail: maria.ferrara1@unicampania.it, antonio.tortora@unicampania.it

Abstract

The homomorphic encryption allows to operate on encrypted data, making any action less vulnerable to hacking. The implementation of a fully homomorphic cryptosystem has long been impracticable. A breakthrough was achieved only in 2009 thanks to Gentry and his innovative idea of bootstrapping. TFHE is a torus-based fully homomorphic cryptosystem using the bootstrapping technique. This paper aims to present TFHE from an algebraic point of view, starting from the `CONCRETE` library which implements TFHE.

Keywords: TFHE, fully homomorphic encryption, bootstrapping, learning with errors

1 Introduction

A fundamental tool in the field of data security and privacy is of course provided by cryptography. However, in traditional cryptographic schemes, data must be decrypted before being manipulated. A completely different approach is given by *homomorphic encryption*, which allows to operate directly on encrypted data, with advantages in terms of privacy and simplification of processes. The term "homomorphic" refers to a cryptosystem where the encryption function is a homomorphism between algebraic structures.

*The authors are members of *National Group for Algebraic and Geometric Structures, and their Applications* (GNSAGA–INdAM), and members of the non-profit association *Advances in Group Theory and Applications*.

Recall that a cryptosystem is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ of finite sets, where $\mathcal{P}, \mathcal{C}$ and $\mathcal{K}$ are respectively the sets of plaintexts, ciphertexts and keys, $\mathcal{E} = \{e_k : \mathcal{P} \to \mathcal{C} \mid k \in \mathcal{K}\}$, $\mathcal{D} = \{d_k : \mathcal{C} \to \mathcal{P} \mid k \in \mathcal{K}\}$, and such that for any encryption function $e_{k_1} \in \mathcal{E}$ there is a decryption function $d_{k_2} \in \mathcal{D}$ such that $d_{k_2}(e_{k_1}(x)) = x$ for all $x \in \mathcal{P}$. The cryptosystem is probabilistic if, for some finite set $\mathcal{S}$, each e_k is a function from $\mathcal{P} \times \mathcal{S}$ to $\mathcal{C}$ and each d_k is a function from $\mathcal{C}$ to $\mathcal{P}$. This allows to encrypt the same plaintext in different ways.

1.1 Homomorphic encryption

In literature, a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is called *partially homomorphic* if, for some binary operations $\cdot$ on $\mathcal{P}$ and $*$ on $\mathcal{C}$, the algebraic structures

- (i) $(\mathcal{P}, \cdot)$ and $(\mathcal{C}, *)$ are semigroups, and
- (ii) the encryption function e_{k_1} is a homomorphism, that is

$$e_{k_1}(x) * e_{k_1}(y) = e_{k_1}(x \cdot y)$$

for all $x, y \in \mathcal{P}$. This implies that $d_{k_2}(e_{k_1}(x) * e_{k_1}(y)) = x \cdot y$. Here, we say that the cryptosystem is *somewhat homomorphic* if $\mathcal{P}$ and $\mathcal{C}$ are each equipped with two operations, say $+, \cdot$ for $\mathcal{P}$ and $\oplus, *$ for $\mathcal{C}$, and

- (i) $(\mathcal{P}, +, \cdot)$ and $(\mathcal{C}, \oplus, *)$ are rings,
- (ii) there exist $X, Y \subseteq \mathcal{P}$, with $X, Y \neq \emptyset$, such that

$$d_{k_2}(e_{k_1}(x_1) \oplus e_{k_1}(x_2)) = x_1 + x_2, \quad (1)$$
$$d_{k_2}(e_{k_1}(y_1) * e_{k_1}(y_2)) = y_1 \cdot y_2, \quad (2)$$

for all $x_1, x_2 \in X$ and $y_1, y_2 \in Y$.

Notice that in this latter case we do not require that e_{k_1} is a (ring) homomorphism. More generally, we say that the cryptosystem is *fully homomorphic* if (1) and (2) hold for all $x_i, y_i \in \mathcal{P}$. As a consequence, for $\mathcal{P} = \mathcal{C}$ and any function

$$f : \underbrace{\mathcal{P} \times \cdots \times \mathcal{P}}_{m} \to \mathcal{P},$$

we have

$$d_{k_2}(f(e_{k_1}(x_1), , e_{k_1}(x_m))) = f(x_1, , x_m).$$

for all $x_i \in \mathcal{P}$.

The origin of fully homomorphic encryption (FHE) is the pioneering paper [10], where Rivest, Adleman and Dertouzos introduced the concept of privacy homomorphism and showed some examples of homomorphic cryptosystems, one of which was RSA. Among other things, they posed the following question: *For what algebraic systems does a useful privacy homomorphism exist*? This question remained unsoved until 2009 when Gentry put forward a first fully homomorphic cryptosystem [5]. His innovative idea consists in taking a probabilistic somewhat homomorphic cryptosystem and make it fully homomorphic. To this aim, it is needed to "clean" data before proceeding with further operations. This technique is called *bootstrapping* and can be described as follows.

Let $e_{k_1} : \mathcal{P} \times \mathcal{S} \to \mathcal{C}$ and $d_{k_2} : \mathcal{C} \to \mathcal{P}$ be the encryption function and the decryption function, respectively, of a probabilistic cryptosystem, and let X, Y be non-empty sets of $\mathcal{S}$ such that

$$d_{k_2}(e_{k_1}(x_1, r_1) \oplus e_{k_1}(x_2, r_2)) = x_1 + x_2$$

and

$$d_{k_2}(e_{k_1}(y_1, s_1) * e_{k_1}(y_2, s_2)) = y_1 \cdot y_2,$$

for all $x_i, y_i \in \mathcal{P}$ and all $r_i \in X, s_i \in Y$. Suppose further that, for a given plaintext x, the element $(x, r) \in \mathcal{P} \times \mathcal{S}$ is *noisy*, in the sense that $r \notin X$. Then, for a new encryption function $g_k : \mathcal{C} \to \mathcal{C}$, depending on the *bootstrapping key* $k = e_{k_3}(k_1)$ for some k_3, the bootstrapping enables to reduce the noise: indeed, it yields an element $r' \in X$ such that $g_k(e_{k_1}(x, r)) = e_{k_1}(x, r')$. Similarly, if $y \in \mathcal{P}$ and $s \notin Y$, we get $g_k(e_{k_1}(y, s)) = e_{k_1}(y, s')$ for some $s' \in Y$. This procedure guarantees that

$$d_{k_2}(e_{k_1}(x_i, r_i) \oplus e_{k_1}(x, r')) = x_i + x$$

and

$$d_{k_2}(e_{k_1}(y_i, s_i) * e_{k_1}(y, s')) = y_i \cdot y.$$

1.2 Fully homomorphic encryption over the torus

Following Gentry's scheme, many other FHE cryptosystems were introduced but all these had a common issue: the bootstrapping took too long to run (minutes) and the bootstrapping key was too large (gigabytes). In 2015, Ducas and Micciancio [4] presented a new method to reduce the running time of the bootstrapping procedure. Their work inspired the birth of the TFHE cryptosystem [1], which runs the bootstrapping in terms of milliseconds and produces a bootstrapping key whose size is measured in megabytes rather than gigabytes.

The letter T in TFHE stands for *torus*, indeed the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. The algebraic structure $(\mathbb{T}, +)$ is an abelian group and thus it can be seen as a $\mathbb{Z}$-module in a canonical way. TFHE is implemented in the `CONCRETE` library of ZAMA [2]. Of course, working with finite precision of 32 or 64 bits, the elements of $\mathbb{T}$ are actually those of its submodule

$$\mathbb{T}_q = \left\{ \frac{i}{q} \,|\, i \in \mathbb{Z}_q \right\} = \left\{ 0, \frac{1}{q}, \ldots, \frac{q-1}{q} \right\}$$

where $q = 2^r$ and r is either 32 or 64; here $\mathbb{Z}_q$ is the group of integers modulo q. Notice also that in `CONCRETE`, for technical requirements, the elements of $\mathbb{T}_q$ are identified with the elements of $\mathbb{Z}_q$.

We point out that, over $\mathbb{Z}_q$, TFHE is fully homomorphic only with respect to the addition. Nevertheless, the bootstrapping is *programmable*, in the sense that it allows to evaluate at the same time a given function on the input ciphertext while reducing the noise. This implies that the product $x \cdot y = \frac{(x+y)^2}{4} - \frac{(x-y)^2}{4}$ of two real numbers can be computed using two bootstrapping operations with the real function $z \mapsto \frac{z^2}{4}$. Hence, TFHE is a fully homomorphic cryptosystem over the field of real numbers.

This paper aims to present TFHE starting directly from `CONCRETE`. It is mostly a survey of [2] and [3], although the approach is more algebraic.

2 A variant of TFHE

2.1 Learning with errors

TFHE is based on the learning with errors problem and its version on rings. The *learning with errors* (LWE) problem, introduced by Regev in [9], asks to recover a secret $s = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$ given a system of m linear equations with small error terms $e_1, \ldots, e_m$, such as the following

$$\begin{cases} a_{11}s_1 + a_{12}s_2 + \ldots + a_{1n}s_n + e_1 = b_1 \pmod{q} \\ a_{21}s_1 + a_{22}s_2 + \ldots + a_{2n}s_n + e_2 = b_2 \pmod{q} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{m1}s_1 + a_{m2}s_2 + \ldots + a_{mn}s_n + e_m = b_m \pmod{q}, \end{cases}$$

in which q and all a_{ij}, b_i are known. Of course, for a fixed value of e, it is possible to derive s in polynomial time using Gaussian elimination. In general each e_i is randomly selected from a discretized gaussian distribution with mean zero and low standard deviation.

More generally, for a ring R, the *ring learning with errors* (RLWE) problem is to find $s \in R$, given many equations $b_i = a_i s + e_i$ where the a_i's are uniformly random in R and the e_i's are "small" in R [7]. The LWE and RWE problems are considered hard and secure even against quantum algorithms.

2.2 The encryption and decryption process

Following TFHE's implementation [3], `CONCRETE` encrypts plaintexts into LWE ciphertexts, and deals with RLWE ciphertexts for the bootstrapping.

Let $\mathbb{B} = \{0, 1\}$ and $q = 2^r$. The parameter r is called the *bit precision* (usually $r = 32$ or 64). For $0 < j < r - 1$, let

$$\mathcal{P} = \{\mu \in \mathbb{Z}_q \,|\, \mu = \mu_j \cdot 2^j + \ldots + \mu_{r-1} \cdot 2^{r-1}, \mu_i \in \mathbb{B}\}$$

be the plaintext space and let $\mathcal{C} = \mathbb{Z}_q$ be the ciphertext space. Consider also the set

$$\mathcal{E} = \{e \in \mathbb{Z}_q \,|\, e = \epsilon_0 \cdot 2^0 + \ldots + \epsilon_{j-1} \cdot 2^{j-1}, \epsilon_i \in \mathbb{B}\}.$$

According to [2], for a secret key $s = (s_1, \ldots, s_n) \in \mathbb{B}^n$ and a *random mask* $a = (a_1, \ldots, a_n) \in \mathbb{Z}_q^n$, the encryption function is given by

$$\text{LWE}_s : (\mu, a, e) \in \mathcal{P} \times \mathbb{Z}_q^n \times \mathcal{E} \mapsto (a, b) \in \mathbb{Z}_q^n \times \mathcal{C}$$

where

$$b = \sum_{i=1}^{n} s_i a_i + \mu + e \pmod{q}.$$

The element e is called *noise*.

Conversely, the decryption function is $\pi \circ \varphi$, indeed the composition of functions

$$\pi : \mu_0 \cdot 2^0 + \ldots + \mu_{r-1} \cdot 2^{r-1} \in \mathbb{Z}_q \mapsto \mu_j \cdot 2^j + \ldots + \mu_{r-1} 2^{r-1} \in \mathcal{P} \tag{3}$$

and

$$\varphi : (a, b) \in \mathbb{Z}_q^n \times \mathcal{C} \mapsto b - \sum_{i=1}^{n} s_i a_i \in \mathbb{Z}_q.$$

In fact, if $b = \sum_{i=1}^{n} s_i a_i + \mu + e \pmod{q}$, then

$$\pi(\varphi(b)) = \pi(\mu + e) = \mu.$$

Actually $\pi(\mu + e) = 2^j \lfloor \frac{\mu+e}{2^j} \rceil$, provided that $e < 2^{j-1}$ or $\epsilon_{j-1} = 0$; here $\lfloor \alpha \rceil$ stands for the nearest integer to α. In fact, if

$$\mu + e = ((\mu_j \cdot 2^j + \ldots + \mu_{r-1} \cdot 2^{r-1}) + \epsilon_0 \cdot 2^0 + \ldots + \epsilon_{j-2} \cdot 2^{j-2}),$$

then

$$2^j \left\lfloor \frac{\mu}{2^j} + \frac{e}{2^j} \right\rceil = 2^j \left(\frac{\mu}{2^j} + \left\lfloor \frac{e}{2^j} \right\rceil \right) = \mu. \tag{4}$$

For the ring setting, let $\mathbb{Z}_{q,N}[x] = \mathbb{Z}_q[x]/I$ where $N > 1$ is a power of 2 and I is the ideal generated by the polynomial $x^N + 1$. Throughout, we will use the notation modulo I for the elements of $\mathbb{Z}_{q,N}[x]$. Consider then the sets

$$\mathbb{Z}_N[x] = \{s(x) \in \mathbb{Z}_{q,N}[x] \mid s(x) = s_0 + s_1 x + \ldots + s_{N-1}x^{N-1}, s_i \in \mathbb{B}\}$$

and

$$\mathcal{E}' = \{e(x) \in \mathbb{Z}_{q,N}[x] \mid e(x) = e_0 + e_1 x + \ldots + e_{N-1}x^{N-1}, e_i \in \mathcal{E}\}.$$

Now the plaintext space is

$$\mathcal{P}' = \{\mu(x) \in \mathbb{Z}_{q,N}[x] \mid \mu(x) = \mu_0 + \mu_1 x + \ldots + \mu_{N-1}x^{N-1}, \mu_i \in \mathcal{P}\}$$

and the ciphertext space is $\mathcal{C}' = \mathbb{Z}_{q,N}[x]$. As above, for a secret key $s(x) = (s_1(x), \ldots, s_k(x)) \in \mathbb{Z}_N[x]^k$ and a public random mask $a(x) = (a_1(x), \ldots, a_k(x)) \in \mathbb{Z}_{q,N}[x]^k$, the encryption function is given by

$$\begin{aligned} \mathrm{RLWE}_{s(x)} : \mathcal{P}' \times \mathbb{Z}_{q,N}[x]^k \times \mathcal{E}' &\to \mathbb{Z}_{q,N}[x]^k \times \mathcal{C}' \\ (\mu(x), a(x), e(x)) &\mapsto (a(x), b(x)) \end{aligned}$$

where

$$b(x) = \sum_{i=1}^{k} s_i(x)a_i(x) + \mu(x) + e(x)$$

and $e(x)$ is the noise.

The decryption function is $\pi' \circ \varphi'$ where π' maps

$$(\alpha_0 + \alpha_1 x + \ldots + \alpha_{N-1}x^{N-1}) \in \mathbb{Z}_{q,N}[x]$$

to

$$(\pi(\alpha_0) + \pi(\alpha_1)x + \ldots + \pi(\alpha_{N-1})x^{N-1}) \in \mathcal{P}',$$

and

$$\varphi' : (a(x), b(x)) \in \mathbb{Z}_{q,N}[x]^k \times \mathcal{C}' \mapsto b(x) - \sum_{i=1}^{k} s_i(x)a_i(x) \in \mathcal{C}'.$$

If $b(x) = \sum_{i=1}^{k} s_i(x)a_i(x) + \mu(x) + e(x)$, then

$$\pi'(\varphi'(b(x))) = \pi'(\mu(x) + e(x)).$$

On the other hand, $\mu(x) + e(x) = (\mu_0 + e_0) + (\mu_1 + e_1)x + \ldots + (\mu_{N-1} + e_{N-1})x^{N-1}$ with $\mu_i \in \mathcal{P}$ and $e_i \in \mathcal{E}$. Since $\pi(\mu_i + e_i) = \mu_i$ for any i, we have $\pi'(\mu(x) + e(x)) = \mu(x)$.

2.3 The encoding and decoding process

`CONCRETE` allows to work with real numbers, encoding any real number of a closed interval $\mathcal{I} = [a, b]$ with an element of $\mathcal{P}$. Of course, a decoding function is applied after decryption. There are two different ways for the encoding/decoding process.

Let h be an integer such that $0 < j \le h \le r-1$, and set

$$\mathcal{M} = \{\mu^* \in \mathbb{Z}_q \mid \mu^* = \mu_0 \cdot 2^0 + \ldots + \mu_h \cdot 2^h, \mu_i \in \mathbb{B}\}.$$

Define the function $\psi : \mathcal{I} \to \mathcal{M}$ by the rule

$$\psi(y) = \left\lfloor 2^j(2^{h-j+1}-1)\frac{y-a}{b-a} \right\rceil.$$

The integer $h-j+1$ represents the number of bits reserved to store y, while j is the number of bits reserved to the noise. Notice that $2^{h-j+1}-1 = 1 \cdot 2^{h-j} + 1 \cdot 2^{h-j-1} + \ldots + 1 \cdot 2^0$, hence

$$2^j(2^{h-j+1}-1) = 1 \cdot 2^j + \ldots + 1 \cdot 2^{h-1} + 1 \cdot 2^h.$$

Obviously, $2^j(2^{h-j+1}-1)\frac{y-a}{b-a} \le 2^j(2^{h-j+1}-1)$. Thus

$$\left\lfloor 2^j(2^{h-j+1}-1)\frac{y-a}{b-a} \right\rceil = \mu_0 \cdot 2^0 + \ldots + \mu_h \cdot 2^h \in \mathcal{M}.$$

The first encoding function is then $E_1 = \pi \circ \psi$, where π is defined as in (3) and therefore $\pi(\mu_0 \cdot 2^0 + \ldots + \mu_h \cdot 2^h) = \mu_j \cdot 2^j + \ldots + \mu_h \cdot 2^h \in \mathcal{P}$.

The decoding function is

$$D_1 : \mu^* \in \mathcal{M} \mapsto a + \frac{\mu^*(b-a)}{2^j(2^{h-j+1}-1)} \in \mathcal{I}.$$

Actually $a + \frac{\mu^*(b-a)}{2^j(2^{h-j+1}-1)} \in \mathcal{I}$. In fact, if $a + \frac{\mu^*(b-a)}{2^j(2^{h-j+1}-1)} > b$ then $\mu^* > 2^j(2^{h-j+1}-1)$, which is impossible. Moreover, $D_1(E_1(y)) \approx y$ for any $y \in \mathcal{I}$.

The second encoding function is

$$E_2 : y \in \mathcal{I} \mapsto 2^j \left\lfloor (2^{h-j+1}-1)\frac{y-a}{b-a} \right\rceil \in \mathcal{P}.$$

Since $(2^{h-j+1}-1)\frac{y-a}{b-a} \le (2^{h-j+1}-1)$, we have

$$\left\lfloor (2^{h-j+1}-1)\frac{y-a}{b-a} \right\rceil = \mu_0 \cdot 2^0 + \ldots + \mu_{h-j} \cdot 2^{h-j} \in \mathcal{M}$$

and so $2^j \left\lfloor (2^{h-j+1}-1)\frac{y-a}{b-a} \right\rceil \in \mathcal{P}$.

In this case the decoding function is

$$D_2 : \mu \in \mathcal{P} \mapsto a + \frac{\mu(b-a)}{2^j(2^{h-j+1}-1)} \in \mathcal{I}.$$

Of course $a + \frac{\mu(b-a)}{2^j(2^{h-j+1}-1)} < b$, otherwise $\mu > 2^j(2^{h-j+1}-1)$, and as before $D_2(E_2(m)) \approx y$ for any $y \in \mathcal{I}$.

3 Operations on ciphertexts

Two LWE ciphertexts can easily be added, but their sum could be a noisy LWE ciphertext. In order to bootstrap such a ciphertext and hence reduce its noise, in [1], following the GSW construction [6], the authors introduced an "external product" of an RLWE ciphertext by an RGSW ciphertext. The result is an approximation of an RLWE ciphertext. This operation depends on some properties of the gadget matrix [8].

3.1 Addition of LWE ciphertexts

In $\mathcal{P} \times \mathbb{Z}_q^n \times \mathcal{E}$ we define

$$(\mu_1, a^{(1)}, e_1) + (\mu_2, a^{(2)}, e_2) = \begin{cases} (\mu_1+\mu_2, a^{(1)}+a^{(2)}, e_1+e_2) & \text{if } e_1+e_2 \in \mathcal{E} \\ 0 & \text{if } e_1+e_2 \in \mathbb{Z}_q \backslash \mathcal{E}. \end{cases}$$

Let

$$\mathrm{LWE}_s(\mu_1, a^{(1)}, e_1) = \left(a^{(1)}, \sum_{i=1}^{n} s_i a_i^{(1)} + \mu_1 + e_1\right) = (a^{(1)}, b_1)$$

and

$$\mathrm{LWE}_s(\mu_2, a^{(2)}, e_2) = \left(a_2, \sum_{i=1}^{n} s_i a_i^{(2)} + \mu_2 + e_2\right) = (a^{(2)}, b_2).$$

Then, provided that $e_1 + e_2 \in \mathcal{E}$, it follows that

$$\mathrm{LWE}_s((\mu_1, a^{(1)}, e_1) + (\mu_2, a^{(2)}, e_2)) = \mathrm{LWE}_s(\mu_1+\mu_2, a^{(1)}+a^{(2)}, e_1+e_2) =$$

$$\left(a^{(1)}+a^{(2)}, \sum_{i=1}^{n} s_i(a_i^{(1)}+a_i^{(2)}) + (\mu_1+\mu_2) + (e_1+e_2)\right) = (a^{(1)}+a^{(2)}, b_1+b_2).$$

This shows that, without bootstrapping, TFHE is somewhat homomorphic with respect to the addition.

Notice also that, for any $(\mu, a, e) \in \mathcal{P} \times \mathbb{Z}_q^n \times \mathcal{E}$ and any $h \in \mathbb{Z}$ such that $h \geq 0$ and $he \in \mathcal{E}$, we have

$$h \cdot \mathrm{LWE}_s(\mu, a, e) = \underbrace{\mathrm{LWE}_s(\mu, a, e) + \ldots + \mathrm{LWE}_s(\mu, a, e)}_{h}$$

$$= \left(ha, \sum_{i=1}^{n} s_i h a_i + h\mu + he \right) = \mathrm{LWE}_s(ha, h\mu, he).$$

Obviously, if $h < 0$, then $h \cdot \mathrm{LWE}_s(\mu, a, e) = (-h) \cdot (-\mathrm{LWE}_s(\mu, a, e)) = (-h) \cdot \mathrm{LWE}_s(-\mu, -a, -e)$.

3.2 The gadget matrix

Given the positive integers k, l, β, with $l\beta \leq r$, and the element $g = (2^{r-\beta}, 2^{r-2\beta}, \ldots, 2^{r-l\beta}) \in \mathbb{Z}_{q,N}[x]^l$, let consider the following block diagonal matrix, also known as the *gadget matrix*,

$$G^T = diag(\underbrace{g^T, \ldots, g^T}_{k+1}) = \begin{pmatrix} g^T & 0 & \ldots & 0 \\ 0 & g^T & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & g^T \end{pmatrix} \in \mathbb{Z}_{q,N}[x]^{(k+1)l \times (k+1)}.$$

Let $d \in \mathbb{Z}_q$. We can see d as an integer in $(-\frac{q}{2}, \frac{q}{2}]$: in fact, if $d > \frac{q}{2}$, then $d = \frac{q}{2} + \alpha$ for some $0 < \alpha < \frac{q}{2}$; but d is congruent to $d - q = \alpha - \frac{q}{2}$ modulo q, where $-\frac{q}{2} < \alpha - \frac{q}{2} < 0$. Write next the representation of $\lfloor 2^{l\beta} \frac{d}{q} \rceil$ to the base 2^β, taking into account that $-2^{l\beta-1} \leq 2^{l\beta} \frac{d}{q} \leq 2^{l\beta-1}$:

$$\left\lfloor 2^{l\beta} \frac{d}{q} \right\rceil = \delta_0 + \delta_1 2^\beta + \ldots + \delta_{l-1} 2^{(l-1)\beta}.$$

Naturally each $\delta_i \in \mathbb{Z}$ is such that $-2^\beta < \delta_i < 2^\beta$. Then we have the following approximation of d:

$$d \approx \frac{q}{2^{l\beta}} \left\lfloor 2^{l\beta} \frac{d}{q} \right\rceil = 2^{r-l\beta} \left\lfloor 2^{l\beta} \frac{d}{q} \right\rceil = \delta_0 2^{r-l\beta} + \delta_1 2^{r-(l-1)\beta} + \ldots + \delta_{l-1} 2^{r-\beta}$$

or, equivalently, $d \approx \sum_{i=1}^{l} d_i 2^{r-i\beta} = g^{-1}(d) g^T$ with $g^{-1}(d) = (d_1, \ldots, d_l) = (\delta_{l-1}, \ldots, \delta_0)$.

By extension, for a polynomial $a(x) = a_0 + a_1 x + \ldots + a_{N-1} x^{N-1} \in \mathbb{Z}_{q,N}[x]$, we set $g^{-1}(a_j) = (d_{j1}, \ldots, d_{jl})$ and

$$g^{-1}(a(x)) = \sum_{j=0}^{N-1} g^{-1}(a_j) x^j = \left(\sum_{j=0}^{N-1} d_{j1} x^j, \ldots, \sum_{j=0}^{N-1} d_{jl} x^j \right).$$

Since $a_j \approx \sum_{i=1}^{l} d_{ji} 2^{r-i\beta}$, it follows that

$$\begin{aligned} a(x) \approx & \sum_{i=1}^{l} d_{0i} 2^{r-i\beta} + \sum_{i=1}^{l} d_{1i} 2^{r-i\beta} x + \ldots + \sum_{i=1}^{l} d_{(N-1)i} 2^{r-i\beta} x^{N-1} \\ = & \left(\sum_{j=0}^{N-1} d_{j1} x^j \right) 2^{r-\beta} + \left(\sum_{j=0}^{N-1} d_{j2} x^j \right) 2^{r-2\beta} + \ldots + \left(\sum_{j=0}^{N-1} d_{jl} x^j \right) 2^{r-l\beta} \\ = & g^{-1}(a(x)) g^T. \end{aligned}$$

More generally, define the function $G^{-1} : \mathbb{Z}_{q,N}[x]^{k+1} \to \mathbb{Z}_{q,N}[x]^{(k+1)l}$ such that

$$G^{-1}(a_1(x), \ldots, a_{k+1}(x)) = (g^{-1}(a_1(x)), \ldots, g^{-1}(a_{k+1}(x))),$$

for any $(a_1(x), \ldots, a_{k+1}(x)) \in \mathbb{Z}_{q,N}[x]^{k+1}$. Thus

$$\begin{aligned} G^{-1}(a_1(x), \ldots, a_{k+1}(x)) G^T &= (g^{-1}(a_1(x)) g^T, \ldots, g^{-1}(a_{k+1}(x)) g^T) \\ &\approx (a_1(x), \ldots, a_{k+1}(x)). \end{aligned} \tag{5}$$

3.3 RGSW ciphertexts

An RGSW ciphertext is a matrix encrypting a polynomial in $\mathbb{Z}_N[x]$. Unlike TFHE, we deal here with the particular case where the polynomial is 0 or 1.

With k and l as in 3.2, put $\bar{k} = (k+1)l$. For a secret key $s(x) = (s_1(x), \ldots, s_k(x)) \in \mathbb{Z}_N[x]^k$, let

$$\begin{aligned} \mathrm{RLWE}_{s(x)}(0, a_{11}(x), \ldots, a_{1k}(x), e_1(x)) &= (0, a_{11}(x), \ldots, a_{1k}(x), b_1(x)) \\ &\vdots \\ \mathrm{RLWE}_{s(x)}(0, a_{\bar{k}1}(x), \ldots, a_{\bar{k}k}(x), e_{\bar{k}}(x)) &= (0, a_{\bar{k}1}(x), \ldots, a_{\bar{k}k}(x), b_{\bar{k}}(x)) \end{aligned}$$

where each $b_i = \sum_{j=1}^{k} s_j(x) a_{ij}(x) + e_i(x)$ for some $e_i(x) \in \mathcal{E}'$. Taking

$$Z = \begin{pmatrix} a_{11}(x) & \ldots & a_{1k}(x) & b_1(x) \\ \vdots & \vdots & \vdots & \vdots \\ a_{\bar{k}1}(x) & \ldots & a_{\bar{k}k}(x) & b_{\bar{k}}(x) \end{pmatrix} \in \mathbb{Z}_{q,N}[x]^{\bar{k} \times (k+1)},$$

we define a new encryption function, as follows

$$\mathrm{RGSW}_{s(x)} : m \in \mathbb{B} \mapsto Z + mG^T \in \mathbb{Z}_{q,N}^{(k+1)l \times (k+1)}.$$

For a matrix $C = (c_1(x), \ldots, c_{\bar{k}}(x)) \in \mathbb{Z}_{q,N}[x]^{1\times\bar{k}}$, we remark that

$$CZ = \left(\sum_{i=1}^{\bar{k}} c_i(x)a_{i1}(x), \ldots, \sum_{i=1}^{\bar{k}} c_i(x)a_{ik}(x), \sum_{i=1}^{\bar{k}} c_i(x)b_i(x) \right)$$

where

$$\begin{aligned} \sum_{i=1}^{\bar{k}} c_i(x)b_i(x) &= \sum_{i=1}^{\bar{k}} c_i(x) \left(\sum_{j=1}^{k} s_j(x)a_{ij}(x) + e_i(x) \right) \\ &= \sum_{j=1}^{k} s_j(x) \left(\sum_{i=1}^{\bar{k}} c_i(x)a_{ij}(x) \right) + \sum_{i=1}^{\bar{k}} c_i(x)e_i(x). \end{aligned}$$

This implies that, up to the noise, CZ can be seen as an $\mathrm{RLWE}_{s(x)}$ ciphertext. In fact,

$$CZ = \mathrm{RLWE}_{s(x)} \left(0, \sum_{i=1}^{\bar{k}} c_i(x)a_{i1}(x), \ldots, \sum_{i=1}^{\bar{k}} c_i(x)a_{ik}(x), \sum_{i=1}^{\bar{k}} c_i(x)e_i(x) \right) \quad (6)$$

provided that $\sum_{i=1}^{\bar{k}} c_i(x)e_i(x) \in \mathcal{E}'$.

3.4 External product of ciphertexts

Given $m \in \mathbb{B}$, $\mu(x) \in \mathbb{Z}_{q,N}[x]$ and a secret key $s(x) \in \mathbb{Z}_N[x]^k$, put $\mathcal{C}_m = \mathrm{RGSW}_{s(x)}(m)$ and let $c_{\mu(x)}$ be the RLWE ciphertext of $\mu(x)$ encrypted using $s(x)$. Denote by $\mathcal{C}_m \boxdot c_{\mu(x)}$ the matrix multiplication of $G^{-1}(c_{\mu(x)}) \in \mathbb{Z}_{q,N}^{1\times(k+1)l}$ by $\mathcal{C}_m \in \mathbb{Z}_{q,N}^{(k+1)l\times(k+1)}$. We claim that, if the noise keeps "small", then $\mathcal{C}_m \boxdot c_{\mu(x)}$ is an approximation of $\mathrm{RLWE}_{s(x)}(m\mu(x), a'(x), e'(x))$, for some $a'(x) \in \mathbb{Z}_{q,N}[x]^k$ and $e'(x) \in \mathcal{E}'$.

First, notice that

$$G^{-1}(c_{\mu(x)})\mathcal{C}_m = G^{-1}(c_{\mu(x)})(Z + mG^T) = G^{-1}(c_{\mu(x)})Z + m(G^{-1}(c_{\mu(x)})G^T).$$

According to (6), we may assume that $G^{-1}(c_{\mu(x)})Z = \mathrm{RLWE}_{s(x)}(0, a(x), e(x))$ for some $a(x), e(x)$. On the other hand, we saw in (5) that $G^{-1}(c_{\mu(x)})G^T \approx c_{\mu(x)}$. Therefore

$$\mathcal{C}_m \boxdot c_{\mu(x)} \approx \mathrm{RLWE}_{s(x)}(0, a(x), e(x)) + mc_{\mu(x)}.$$

In particular, if $\mathrm{RLWE}_{s(x)}(0, a(x), e(x)) = (a(x), \sum_{i=1}^{k} s_i(x)a_i(x) + e_i(x))$,

$$c_{\mu(x)} = \left(\bar{a}(x), \sum_{i=1}^{k} s_i(x)\bar{a}_i(x) + \mu(x) + \bar{e}_i(x) \right)$$

and $e(x) + m\bar{e}_i(x) \in \mathcal{E}'$, we conclude that

$$\mathcal{C}_m \boxdot c_{\mu(x)} \approx \mathrm{RLWE}_{s(x)}(m\mu(x), a(x) + m\bar{a}(x), e(x) + m\bar{e}_i(x)), \qquad (7)$$

as claimed.

4 Bootstrapping algorithms

In TFHE the bootstrapping procedure involves two main algorithms, the blind rotation and the sample extraction. The blind rotation requires first a switch modulus algorithm to scale by $\frac{2N}{q}$ each component of the LWE ciphertext. Whereas the sample extraction is usually followed by a key switching algorithm which converts a ciphertext under a key into a ciphertext under another key.

4.1 Blind rotation

Let $b = \sum_{i=1}^{n} s_i a_i + \mu^*$, with $\mu^* = \mu + e$, be a noisy LWE ciphertext corresponding to the mask $(a_1, \ldots, a_n)$, and let consider the so-called *test polynomial*

$$t(x) = t_0 + t_1 x + \ldots + t_{N-1} x^{N-1} \in \mathbb{Z}_{q,N}[x],$$

where each $t_j = \pi(\frac{q}{2N} j)$ and π is defined as in (3). Suppose further that b has at least one bit of padding left, that is $b < 2^{r-1}$. Then $\mu^* < 2^{r-1}$ and so, arguing as in (4), we obtain $\bar{\mu}^* = \lfloor \frac{2N}{q} \mu^* \rceil \leq N - 1$. Hence, $t_{\bar{\mu}^*} = \mu$.

The blind rotation consists in finding an RLWE ciphertext of the polynomial $x^{-\bar{\mu}^*} t(x)$, whose constant term is actually $t_{\bar{\mu}^*}$. In what follows we will rely on

$$\widetilde{\mu}^* = \bar{b} - \sum_{i=1}^{n} s_i \bar{a}_i,$$

where $\bar{b} = \lfloor \frac{2N}{q} b \rceil$ and $\bar{a}_i = \lfloor \frac{2N}{q} a_i \rceil$. This approximation may produce a small additional error, which is called *drift*.

Let $s = (s_1, \ldots, s_n)$ and $s(x) = (s_1(x), \ldots, s_k(x))$ be the secret keys used to encrypt the input LWE ciphertexts and RLWE ciphertexts, respectively.

Then the *bootstrapping key* is defined to be a list of n RGSW ciphertexts, each one encrypting s_i, namely by

$$(\mathcal{C}_{s_1}, \ldots, \mathcal{C}_{s_n}) = (\mathrm{RGSW}_{s(x)}(s_1), \ldots, \mathrm{RGSW}_{s(x)}(s_n)).$$

Define recursively

$$c_i = \mathcal{C}_{s_i} \boxdot (x^{\bar{a}_i} c_{i-1} - c_{i-1}) + c_{i-1},$$

where

$$c_0 = (\underbrace{0, \ldots, 0}_{k}, x^{-\bar{b}} t(x)) = \mathrm{RLWE}_{s(x)}(x^{-\bar{b}} t(x), \underbrace{0, \ldots, 0}_{k+1}).$$

Then, for any $j \in \{1, \ldots, n\}$, we get

$$c_j \approx \mathrm{RLWE}_{s(x)}(x^{-\bar{b} + \sum_{i=1}^{j} s_i \bar{a}_i} t(x), a(x), e(x))$$

for some $a(x)$ and $e(x)$. In fact, applying (7) with $m = s_j$ and $s_0 = 0$, we have

$$\begin{aligned} c_j \approx\ & \mathcal{C}_{s_j} \boxdot \mathrm{RLWE}_{s(x)}((x^{a_j} - 1) x^{-\bar{b} + \sum_{i=1}^{j-1} s_i \bar{a}_i} t(x), a(x), e(x)) + \\ & \mathrm{RLWE}_{s(x)}(x^{-\bar{b} + \sum_{i=1}^{j-1} s_i \bar{a}_i} t(x), a(x), e(x)) \\ \approx\ & \mathrm{RLWE}_{s(x)}(s_j (x^{a_j} - 1) x^{-\bar{b} + \sum_{i=1}^{j-1} s_i \bar{a}_i} t(x), a'(x), e'(x)) + \\ & \mathrm{RLWE}_{s(x)}(x^{-\bar{b} + \sum_{i=1}^{j-1} s_i \bar{a}_i} t(x), a(x), e(x)) \\ =\ & \begin{cases} \mathrm{RLWE}_{s(x)}(x^{-\bar{b} + \sum_{i=1}^{j} s_i \bar{a}_i} t(x), a'(x), e'(x)) & \text{if } s_j = 0 \\ \mathrm{RLWE}_{s(x)}(x^{-\bar{b} + \sum_{i=1}^{j} s_i \bar{a}_i} t(x), a''(x), e''(x)) & \text{if } s_j = 1. \end{cases} \end{aligned}$$

In particular,

$$c_n \approx \mathrm{RLWE}_{s(x)}(u(x), a(x), e(x)) \tag{8}$$

where $u(x) = x^{-\widetilde{\mu}^*} t(x) = u_0 + u_1 x + \ldots + u_{N-1} x^{N-1}$ and $u_0 = t_{\widetilde{\mu}^*}$ is an approximation of μ. Therefore c_n is the desired RLWE ciphertext.

4.2 Sample extraction

The next step is to extract u_0, as an LWE ciphertext of μ with less noise than b. With the same notation of 4.1, let

$$\mathrm{RLWE}_{s(x)}(u(x), a(x), e(x)) = (a_1(x), \ldots, a_k(x), b(x))$$

and, for any $1 \leq i \leq k$, put

$$\begin{aligned} s_i(x) = &\ s_{i,0} + s_{i,1}x + \ldots + s_{i,N-1}x^{N-1}, \\ a_i(x) = &\ a_{i,0} + a_{i,1}x + \ldots + a_{i,N-1}x^{N-1}, \\ e(x) = &\ e_0 + e_1 x + \ldots + e_{N-1}x^{N-1}. \end{aligned}$$

Then

$$\begin{aligned} b(x) = &\ \sum_{i=1}^{k} s_i(x)a_i(x) + u(x) + e(x) \\ = &\ b_0 + b_1 x + \ldots + b_{N-1}x^{N-1} \end{aligned}$$

where, since $x^N = -1$ in $\mathbb{Z}_{q,N}[x]$, we have

$$b_0 = \left(\sum_{i=1}^{k} s_{i,0}a_{i,0} - (s_{i,1}a_{i,N-1} + \ldots + s_{i,N-1}a_{i,1}) \right) + u_0 + e_0.$$

Taking

$$s' = (s_{1,0}, s_{1,1}, \ldots, s_{1,N-1}, \ldots, s_{k,0}, s_{k,1}, \ldots, s_{k,N-1}),$$

it follows that

$$\begin{aligned} &\mathrm{LWE}_{s'}(a_{1,0}, -a_{1,N-1}, \ldots, -a_{1,1}, \ldots a_{k,0}, -a_{k,N-1}, \ldots, -a_{k,1}, u_0, e_0) = \\ &\quad (a_{1,0}, -a_{1,N-1}, \ldots, -a_{1,1}, \ldots a_{k,0}, -a_{k,N-1}, \ldots, -a_{k,1}, b_0). \end{aligned}$$

Hence, with the above mask, b_0 can be seen as an LWE ciphertext of u_0 under the key s'. In [1], in order to convert b_0 into an LWE ciphertext of u_0 under the key $s = (s_1, \ldots, s_n)$, a key switching algorithm is applied. This technique is similar to the bootstrapping (see [3, Appendix A]) but it slightly makes the noise increase. Here we point out that, since the parameters (n, N) are usually chosen such that $n \leq N$, we may assume $s_1(x) = \ldots = s_k(x) = s_1 + s_2 x + \ldots + s_n x^{n-1}$. So, for example, if $k = 2$ then $s' = (s_1, \ldots, s_n, \underbrace{0, \ldots, 0}_{N-n}, s_1, \ldots, s_n, \underbrace{0, \ldots, 0}_{N-n})$ and

$$\begin{aligned} b_0 = &\ s_1 a_{1,0} - s_1 a_{1,N-1} - \ldots - s_1 a_{1,N-(n-1)} + \\ &\ s_2 a_{2,0} - s_2 a_{2,N-1} - \ldots - s_2 a_{2,N-(n-1)} + u_0 + e_0, \end{aligned}$$

that is

$$\mathrm{LWE}_s(a_{1,0}, -a_{1,N-1}, \ldots, -a_{1,N-(n-1)}, a_{2,0}, -a_{2,N-1}, \ldots, -a_{2,N-(n-1)}, u_0, e_0) = (a_{1,0}, -a_{1,N-1}, \ldots, -a_{1,N-(n-1)}, a_{2,0}, -a_{2,N-1}, \ldots, -a_{2,N-(n-1)}, b_0).$$

4.3 Programmable bootstrapping

Given a function $f_q : \mathbb{Z}_q \to \mathbb{Z}_q$, suppose now that the initial test polynomial is

$$t(x) = t_0 + t_1 x + \ldots + t_{N-1}x^{N-1} \in \mathbb{Z}_{q,N}[x],$$

where $t_j = f_q(\pi(\frac{q}{2N}j))$. Thus, as in 4.1, we obtain that the constant term of the polynomial $u(x) = x^{-\widetilde{\mu}^*}t(x)$ in (8) is $t_{\widetilde{\mu}^*}$, namely $f_q(\mu)$ up to the drift. Hence, in this case, the bootstrapping transforms a ciphertext of μ into a ciphertext of $f_q(\mu)$ with a lesser noise. Furthermore, by using the encoding/decoding functions defined in 2.3, it is possible to evaluate any real-valued function of a real variable, say $f : \mathcal{I} \to \mathcal{J}$. In fact, it is enough to take $f_q = E'_i \circ f \circ D_i$ where $i = 1$ or 2, $E'_i : \mathcal{J} \to \mathbb{Z}_q$ and $D_i : \mathbb{Z}_q \to \mathcal{I}$. So, if $D'_i : \mathcal{J} \to \mathbb{Z}_q$ and $\mu = E_i(x)$ with $E_i : \mathcal{I} \to \mathbb{Z}_q$, then $f_q(\mu) \approx E'_i(f(x))$ from which it follows that $D'_i(f_q(\mu)) \approx f(x)$.

Acknowledgments. This research is supported by a grant of the University of Campania "Luigi Vanvitelli", in the framework of the projects GoAL (V:ALERE 2019) and HELM (V:ALERE 2020).

References

[1] I. Chillotti, N. Gama, M. Georgieva and M. Izabachene, *TFHE: fast fully homomorphic encryption over the torus*, J. Cryptology **33** (2020), no. 1, 34–91.

[2] I. Chillotti, M. Joye, D. Ligier, J.-B. Orfila and S. Tap, *CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE*, Demo paper at WAHC 2020 (8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography), pp. 6, https://whitepaper.zama.ai/concrete/WAHC2020Demo.pdf.

[3] I. Chillotti, M. Joye and P. Paillier, *Programmable bootstrapping enables efficient homomorphic inference of deep neural networks*, Cyber Security Cryptography and Machine Learning (CSCML 2021), 1-19, Lecture Notes in Comput. Sci., 12716, Springer, 2021.

[4] L. Ducas and D. Micciancio, *FHEW: Bootstrapping homomorphic encryption in less than a second*, Advances in Cryptology – EUROCRYPT 2015, Part I, 617–640, Lecture Notes in Comput. Sci., 9056, Springer, Heidelberg, 2015.

[5] C. Gentry, *Computing arbitrary functions on encrypted data*, Communications of the ACM **53** (2010), no. 3, 97–105.

[6] C. Gentry, A. Sahai and B. Waters, *Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based*, Advances in Cryptology – CRYPTO 2013, 75–92, Lecture Notes in Comput. Sci., 8042. Springer, Berlin, Heidelberg.

[7] V. Lyubashevsky, C. Peikert and O. Regev. 2013. *On ideal lattices and learning with errors over rings*, J. ACM **60** (2013), no. 6, Art. 43, 35 pp.

[8] D. Micciancio and C. Peikert, *Trapdoors for lattices: simpler, tighter, faster, smaller*, Advances in Cryptology – EUROCRYPT 2012, 700–718, Lecture Notes in Comput. Sci., 7237, Springer, Heidelberg, 2012.

[9] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, J. ACM **56** (2009), no. 6, Art. 34, 40 pp.

[10] R. L. Rivest, L. Adleman and M. L. Dertouzos, *On data banks and privacy homomorphism*, Foundations of secure computation (Workshop, Georgia Inst. Tech., Atlanta, Ga., 1977), 169–179, Academic, New York, 1978.