Fully Homomorphic Encryption and Bootstrapping

Rémi Leluc, Elie Chedemail, Adéchola Kouande, Quyen Nguyen, Njaka Andriamandratomanana

Fully Homomorphic Encryption and Bootstrapping

Rémi Leluc REMI.LELUC@TELECOM-PARIS.FR
LTCI, Télécom Paris, France

Adéchola Kouande KOUANDE@CEREMADE.DAUPHINE.FR
CEREMADE, Paris-Dauphine, France

Elie Chedemail ELIE.CHEDEMAIL2@ENSAI.FR
ENSAI, Rennes, France

Thi Thu Quyen Nguyen THI-THU-QUYEN.NGUYEN@INRIA.FR
IRISA, Rennes, France

Njaka Andriamandratomanana NJAKA-HARILALA.ANDRIAMANDRATOMANANA@UNICAEN.FR
LMNO , Caen, France

Abstract

This report is the result of a work done during the SEME (*Semaine d'Étude Mathématiques Entreprise*) in Rennes (May 2nd - May 6th 2022). The project presented here concerns the company Ravel Technologies and deals with the field of homomorphic encryption for secure data processing. Homomorphic encryption is an encryption that allows users to do computations on encrypted data without first decrypting them. An encryption algorithm must switch with elementary operations, which are at least addition and multiplication. A straightforward application of a homomorphic encryption for the delegation of calculations concerns cloud computing service where there is a need to perform calculations while preserving the confidentiality of the data, *e.g.* for the medical and banking sectors. Among possible algorithms, those that have become popular over the last ten years are based on the so-called Learning With Errors (LWE) problem, for performance and security reasons. Since this technique introduces noise into the encryption, which can grow during homomorphic computations to the point that later decryption fails, it is necessary to consider a noise reduction method such as bootstrapping.

Keywords: homomorphic encryption, bootstrap, (ring) learning with errors, blind rotation

1. Introduction

Context. Homomorphic encryption is a form of encryption that permits users to perform computations on its encrypted data without first decrypting them. These resulting computations are left in an encrypted form which, when decrypted, results in an identical output to that produced had the operations been performed on the unencrypted data. This allows data to be encrypted and out-sourced to commercial privacy-preserving cloud environments for processing, all while encrypted. An encryption scheme that supports arbitrary computations on ciphertexts is known as fully homomorphic encryption (FHE). Such a scheme enables the construction of programs for any desirable functionality, which can be run on encrypted inputs to produce an encryption of the result. Since such a program do not need to decrypt its data, it can be run by an untrusted party without revealing its inputs and internal states. Fully homomorphic encryptions have great practical implications in the outsourcing of private computations, for instance, in the context of cloud computing and many applications for the banking and medical sectors.

Historical background. With the breakthrough discoveries of key exchange in 1976 (Diffie and Hellman, 1976), public-key encryption (also known as asymmetric encryption), and digital signatures in 1978 (Rivest et al., 1978), the scope of cryptology broadened considerably. In public-key cryptography, each party has a pair of keys: a public one and a private (or secret) one. The public one can be published, e.g., on the Internet, and allows anyone to encrypt a message, that can only be decrypted with the corresponding private key. In order to explain this concept, a famous analogy is often used: the public key corresponds to an open lock, whereas the private key corresponds to the lock's key. Publishing the public key is equivalent to making the open lock available; then anyone can write a message, put it in a box, and close the box with the provided lock. The sealed box is then sent to the recipient, who can open it with the appropriate key. Interestingly, since it uses lattice cryptography, homomorphic encryption is resistant to quantum computing algorithm hacks and appears as a viable solution for secure data processing on the cloud. However, the main obstacle that researchers faced for more than 30 years was the growth of the noise: after many computations, the noise level will become too large and it will be impossible to proceed without losing correctness.

Bootstrap revolution. The turning point came in 2009 with the breakthrough by Craig Gentry (Gentry, 2009) who put forward the first plausible construction for an FHE (Fully Homomorphic Encryption) scheme based on the hardness of some lattice problems. This breakthrough had the effect of reigniting FHE as a topic of research, and since then many important results followed. Although the techniques improved greatly, and the schemes became simpler and more efficient, the original blueprint presented in Gentry's thesis continues to underlie all known FHE constructions.

The key idea that was proposed in Gentry's work is that of bootstrapping. By this term, we denote the process of refreshing a ciphertext in order to produce a new ciphertext that encrypts the same message, but with a lower level of noise so that more homomorphic operations can be evaluated on it. This operation is at the very core of any FHE schemes known to date, and consists of homomorphically evaluating the decryption circuit of the scheme. Roughly speaking, it is like decrypting the ciphertext with the secret key, and then re-encrypting the message, with the difference that the secret key is not known and it is replaced by an encryption of the secret key, called the bootstrapping key. This requires an additional hardness assumption, called circular security assumption, meaning that we must assume it is safe to publish an encryption of the secret key under itself. Although this assumption is still not well studied and understood, and it is sometimes regarded with suspicion, no attacks that exploit this extra piece of information have been proposed.

Outline. Section 2 introduces the mathematical background then homomorphic encryption is presented in Section 3. The well-known schemes LWE and RLWE are explained in Section 4 and the whole boostrap procedure is detailed in Section 5. Finally, Section 6 concludes the report with discussions for further research.

2. Preliminaries

Notations. Let n be a power of two. We denote the $2n$-th cyclotomic ring by $\mathcal{R} := \mathbb{Z}[X]/(X^n+1)$ and its quotient ring by $\mathcal{R}_Q := \mathcal{R}/Q\mathcal{R}$. Ring elements in $\mathcal{R}$ are indicated in bold, e.g. $\boldsymbol{a} = \boldsymbol{a}(X)$. For two vectors $\vec{a}$ and $\vec{b}$, we denote their inner product by $\langle \vec{a}, \vec{b} \rangle$. We write the floor, ceiling and round functions as $\lfloor \cdot \rfloor, \lceil \cdot \rceil$ and $\lfloor \cdot \rceil$, respectively. For $q \in \mathbb{Z}$ and

$q > 1$, we identify the ring $\mathbb{Z}_q$ with $[-q/2, q/2)$ as the representative interval, and for $x \in \mathbb{Z}$ we denote the centered remainder of x modulo q by $[x]_q \in \mathbb{Z}_q$. We extend these notations to elements of $\mathcal{R}$ by applying them coefficient-wise. We use $a \leftarrow S$ to denote uniform sampling from the set S. We denote sampling according to a distribution χ by $a \leftarrow \chi$.

Probabilities. Given a probability distribution $\mathcal{D}$, we use $x \leftarrow D$ to denote that x is sampled from $\mathcal{D}$. For a set $S, x \leftarrow S$ denotes that x is sampled uniformly from S. A distribution χ over the integers is called B-bounded if it is supported on $[-B, B]$.

Definition 1 (Discrete Gaussian) *The discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ over the integers is the probability distribution that assigns a probability proportional to $\exp\left(-\pi|x|^2/\sigma^2\right)$ to each $x \in \mathbb{Z}$.*

The discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ is then used to define a distribution χ on R. The distribution χ is in general not as simple as just sampling the coefficients according to $D_{\mathbb{Z},\sigma}$. However, for the polynomial $f(x) = x^d + 1$ with d a power of 2 , we can indeed define χ as $D^d_{\mathbb{Z},\sigma}$. For more general cyclotomic polynomials, sampling from χ is only slightly more involved. Recall that for the normal distribution $\mathcal{N}\left(0, \sigma^2\right)$, we have that $\text{Prob}_{x \leftarrow \mathcal{N}(0,\sigma^2)}[|x| > k \cdot \sigma] = \text{erf}(k/\sqrt{2})$. As such define the function $\beta(\epsilon) := \min\{\beta \mid \text{erf}(\beta/\sqrt{2}) < \epsilon\}$, then with probability $1 - \epsilon$ the samples are bounded by $\beta \cdot \sigma$.

3. Homomorphic Encryption (HE)

Homomorphic encryption is a form of encryption with an additional evaluation capability for computing over encrypted data without access to the secret key. The result of such a computation remains encrypted. Homomorphic refers to homomorphism in algebra: the encryption and decryption functions can be thought of as homomorphisms between the plaintext space $\mathscr{P} = (\{plaintext\}, +, \times)$ and the ciphertext space $\mathscr{C} = (\{ciphertext\}, \oplus, \otimes)$. For any encryption $\varphi : \mathscr{P} \to \mathscr{C}$, the homomorphic property is described as follows.

Definition 2 (Homomorphic encryption) *Let $\varphi : \mathscr{P} \to \mathscr{C}$ be any encryption scheme. φ is called homomorphic encryption (HE) if it preserves addition or multiplication of two messages, i.e., $\varphi(m_1 + m_2) = \varphi(m_1) \oplus \varphi(m_2)$ or $\varphi(m_1 \times m_2) = \varphi(m_1) \otimes \varphi(m_2)$. Moreove, φ is called fully homomorphic encryption (FHE) if it preserves both addition and multiplication.*

Injecting errors. Among all the possible algorithms for homomorphic encryption, those that have become popular over the last ten years are all based on the so-called Learning With Errors (LWE) problem, for performance and security reasons. In these schemes, an error term is injected during the encryption procedure for security purposes. The reason is that these encryption schemes rely on the hardness of solving "noisy" problems, i.e., problems where the relations are not exact, but are perturbed by a moderate quantity of error. Combining multiple ciphertexts through homomorphic operations has the side effect of combining the noises as well, thus increasing the magnitude of the error in the resulting encryption. When the error grows beyond a certain threshold, correctness is lost, meaning that the decryption procedure will not return the expected result. We say that an encryption scheme is somewhat homomorphic if it can evaluate a certain number of homomorphic operations, before the error grows too much to maintain the correctness of the evaluation.

Parameters trade-off. When practically instantiating an encryption scheme, one of the most delicate steps is that of choosing parameters. This usually requires finding a sensible trade-off between security and efficiency, and remains one of the potentially weak points for theoretically secure schemes. This is also the case for homomorphic encryption schemes, but in this case the problem of finding parameters is even more important. In fact, parameters like the size of the modulus or the standard deviation of noise terms define the threshold below which the noise must remain in order to guarantee a correct decryption.

4. Learning With Errors (LWE) and Ring Learning With Errors (RLWE)

The learning with errors (LWE) problem was introduced in Regev (2009), and has become one of the most known problems in lattice-based cryptography. It has been used to construct several cryptosystems, and it is believed to be hard even for quantum computers. Learning with errors (LWE) is a conjecturally hard problem in lattice-based cryptography. Its hardness allows the construction of public key encryption/decryption schemes.

4.1 LWE Encryption

LWE encryption is the encryption based on LWE problem. This encryption scheme is parameterized by two positive integers, n and q, and an error distribution χ over $\mathbb{Z}$, usually a discrete Gaussian of width $\alpha q, 0 < \alpha < 1$.

Definition 3 (LWE distribution) *Given a secret vector* $\mathbf{s} \in \mathbb{Z}_q^n$*, the LWE distribution* $LWE_{\mathbf{s},\chi}$ *over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *is sampled by picking* $\vec{a} \leftarrow \mathbb{Z}_q^n$*, an error* $e \leftarrow \chi$*, and returning* $(\vec{a}, b = \langle \vec{s}, \vec{a} \rangle + e)$.

In other words, for $n, q, t \in \mathbb{N}^*$ with $t|q$ and a message $m \in \mathbb{Z}_t \subset \mathbb{Z}_q$, the LWE encryption with the key $\vec{s} \leftarrow \chi_{key}(\mathbb{Z}_q^n)$ of m is defined as:

$$\mathrm{LWE}_{q,\vec{s}}(m) := (\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + \tilde{m} + e) \in \mathbb{Z}_q^{n+1}$$

where $\vec{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi_{error}(\mathbb{Z}_q)$ and $\tilde{m} = \frac{q}{t}m$. The decryption of a ciphertext $(\vec{a}, b)$ of m is

$$\mathrm{LWE}_{\vec{s}}^{-1}(\vec{a}, b) := \lceil \frac{t}{q}(b - \langle \vec{a}, \vec{s} \rangle) \rfloor \in \mathbb{Z}_q.$$

Let $Err_{\mathrm{LWE}}((\vec{a}, b), m) = \frac{t}{q}(b - \langle a, s \rangle) - m = \frac{t}{q}e$. If $|\frac{t}{q}e| \in [0, 1/2]$ then $\mathrm{LWE}^{-1}(\vec{a}, b) = m$, thus the decryption is successful. Here we observe the interest of keeping the function Err_{LWE_s} small in order to decrypt successfully. For this case, we set $t << q$ and χ_{error} as small noise distribution. Note that the addition of LWE ciphertexts is straightforward and is given by

$$\begin{aligned} \mathrm{LWE}_{q,\vec{s}}(m_1) + \mathrm{LWE}_{q,\vec{s}}(m_2) &= (\vec{a}_1, b_1) + (\vec{a}_2, b_2) = (\vec{a}_1 + \vec{a}_2, b_1 + b_2) \\ &= (\vec{a}_1 + \vec{a}_2, \langle \vec{a}_1 + \vec{a}_2, s \rangle + (\tilde{m}_1 + \tilde{m}_2) + (e_1 + e_2)) \\ &= LWE_{q,\vec{s}}(m_1 + m_2). \end{aligned}$$

LWE encryption is then additively homomorphic with an accumulated noise. Such noise needs to be reduced in order to ensure a succesful decryption. This is the aim of the bootstrap procedure that we describe in Section 5.

4.2 Ring-LWE Encryption

RLWE encryption was introduced in Stehlé et al. (2009); Lyubashevsky et al. (2010) and is the encryption based on LWE problem on a well chosen polynomial quotient ring. Similarly to LWE, the encryption distribution of RLWE is given by the following definition.

Definition 4 (Ring-LWE distribution) *Let $\mathcal{R}$ be a polynomial ring such that $\mathcal{R} = \mathbb{Z}[X]/f(X)$, with f some cyclotomic polynomial of degree n. Also, let $q \geq 2$ be an integer modulus, and let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ be the quotient ring. Finally, let χ be an error distribution over $\mathcal{R}$. For a fixed secret $\boldsymbol{s} \in \mathcal{R}_q$, the ring-LWE distribution ring-lwe $_{s,\chi}$ is sampled by taking $\boldsymbol{a} \leftarrow \mathcal{R}_q, \boldsymbol{e} \leftarrow \chi$, and outputting $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{s} \cdot \boldsymbol{a} + \boldsymbol{e})$. All the computations are, as usual, done modulo q.*

In other words, for $n, Q, t \in \mathbb{N}^*$ with $t|q$ and a message $\boldsymbol{m}(X) \in \mathcal{R}_Q$, the RLWE encryption with the key $\boldsymbol{z}(X) \leftarrow \chi_{key}(\mathcal{R}_Q)$ of $\boldsymbol{m} \in \mathcal{R}_t$ is defined as:

$$\text{RLWE}_{Q,\boldsymbol{z}}(\boldsymbol{m}) := (\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{z} + \tilde{\boldsymbol{m}} + \boldsymbol{e}) \in \mathcal{R}_Q^2$$

where $\boldsymbol{a} \leftarrow \mathcal{R}_Q$, error $\boldsymbol{e} \leftarrow \chi_{error}(\mathcal{R}_Q)$ and $\tilde{\boldsymbol{m}} = \frac{Q}{t}\boldsymbol{m}$. Again, note that from the definition, RLWE encryption is linearly homomorphic. The advantage of using ring-LWE instead of plain LWE is compactness and efficiency. In fact, in the case of ring-LWE, each sample gives a n-dimensional pseudorandom ring element $b \in \mathcal{R}$, instead of just a pseudorandom scalar $b \in \mathbb{Z}_q$. We can thus say that a single ring-LWE sample with $a \in \mathcal{R}$ takes the place of n LWE samples with vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$. Moreover, thanks to techniques like the Fast Fourier Transform (FFT) (Brigham, 1988), the multiplication between ring elements can be performed in quasi-linear time.

External Product for RLWE. Since the RLWE encryption scheme is homomorphic for the addition, we are now interested in the multiplication operation. For that matter, we rely on another encryption scheme with allows to define an external product on RLWE. Consider the following ecryption scheme

$$\text{RLWE}'_{\boldsymbol{z}}(\boldsymbol{m}) = \text{RLWE}_{\boldsymbol{z}}(\boldsymbol{m}, \text{RLWE}_{\boldsymbol{z}}(B\boldsymbol{m}), \cdots, \text{RLWE}_{\boldsymbol{z}}(B^{k-1}\boldsymbol{m}))$$

using the same keys $\boldsymbol{z} \in R_q$, and where ciphertexts consist of $k = \log_B q$ basic encryptions produced by the original RLWE scheme. The base B can be set differently to achieve various time/space trade-offs. For simplicity, we assume q is a power of B, but the scheme can be easily adapted to other values. Mixed base variants are also possible, where B^i is replaced by a product $B_1 \cdots B_i$. Incidentally, we note that these ciphertexts allow to recover the message $\boldsymbol{m}$ exactly, even without encoding/scaling, by first decrypting RLWE $\left(B^{k-1}\boldsymbol{m}\right)$ to recover $(\boldsymbol{m} \bmod B)$. Then subtracting $B^{k-2} \cdot (\boldsymbol{m} \bmod B)$ from RLWE$_s$ $\left(B^{k-2}\boldsymbol{m}\right)$ to recover $\left(\boldsymbol{m} \bmod B^2\right)$, and so on. We then define

$$\text{RGSW}_{\boldsymbol{z}}(\boldsymbol{m}) = (\text{RLWE}'_{\boldsymbol{z}}(-\boldsymbol{z} \cdot \boldsymbol{m}), \text{RLWE}'_{\boldsymbol{z}}(\boldsymbol{m}))$$

Interstingly, note how RGSW$_{\boldsymbol{z}}(m)$ ciphertexts can be equivalently written as

$$\text{RGSW}_{\boldsymbol{z}}(m) = (\text{RLWE}_{\boldsymbol{z}}(0), \ldots, \text{RLWE}_{\boldsymbol{z}}(0)) + \boldsymbol{m}\mathbf{G}$$

where $\mathbf{G} = \mathbf{I}_2 \otimes \left(1, B, B^2, \ldots, B^{k-1}\right)^\top \in \mathcal{R}_Q^{2k \times 2}$ is the powers-of-B "gadget matrix". Such formula may be derived from Ducas and Micciancio (2015). For the notations, note that each

ciphertext of 0 may be written as $\text{RLWE}_{\boldsymbol{z}}(0) = (\boldsymbol{a}_i, \boldsymbol{b}_i)$ and we have k such encryptions in the formula above which are stacked to obtain an element of $\mathcal{R}_Q^{2k\times 2}$. We finally obtain the multiplication operation in RLWE thanks to the following external product

$$\diamond : \text{RLWE} \times \text{RGSW} \longrightarrow \text{RLWE}$$

Such multiplication allows to define the product on ciphertexts as follows.

Lemma 5 *For two messages $\boldsymbol{m_0}, \boldsymbol{m_1} \in \mathcal{R}_Q$ with $\boldsymbol{m_1}$ small we have:*

$$RLWE_{\boldsymbol{z}}(\boldsymbol{m_0}) \diamond RGSW_{\boldsymbol{z}}(\boldsymbol{m_1}) = RLWE_{\boldsymbol{z}}(\boldsymbol{m_0} \cdot \boldsymbol{m_1})$$

5. Bootstrap procedure

Bootstrap is a noise reduction technique for LWE encryption. The general idea is to transform the LWE encryption into an RLWE encryption in which the noise of the former is removed and the noise of latter is reduced thanks to RLWE's rich settings. Once a small-noised RLWE encryption with is obtained, we can extract from it a small-noised LWE encryption. We implemented the bootstrap procedure as well as all necessary tool functions to work in ring polynomials. The code is written in Python and is available upon request. From a practical point of view, the bootstrap procedure may be expressed as follows: it takes as input a ciphertext $LWE_s(m) = (\vec{a}, b)$ with error e then apply the three steps:

- **Step 1: Blind rotation**

$$(\vec{a}, b) = LWE_s(m) \to (a_z(X), b_z(X)) = RLWE_{z(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 2: Key switching**

$$(a_z(X), b_z(X)) \to (a_s(X), b_s(X)) = RLWE_{s(X)}(X^{-b+\langle \vec{a}, \vec{s} \rangle} \cdot w(X))$$

- **Step 3: Extraction**

$$(a_s(X), b_s(X)) \to (\vec{a}, b) = LWE_s(m)$$

and the final output is a ciphertext $LWE_s(m) = (\vec{a}, b)$ with small error $e' < e$. This full procedure is summarized in Figure 1 below where the right part concerns the "bootstrap world" where operations are performed on ring polynomials.

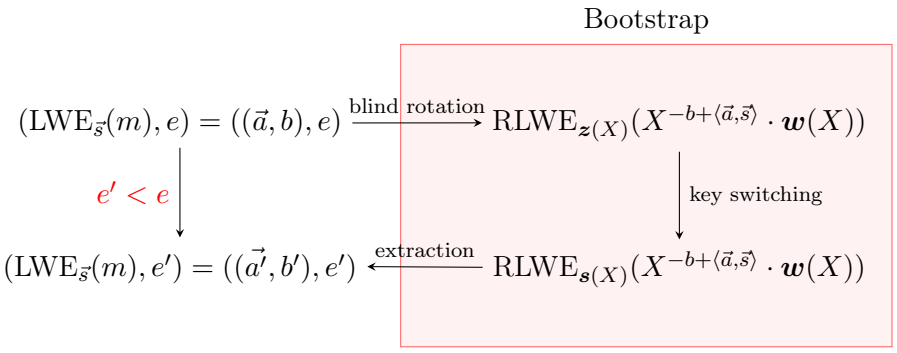

Figure 1: Representation of the bootstrap Procedure

Bootstrap scheme. For a good choice of $\boldsymbol{w}(X) \in \mathcal{R}_Q$ we compute the $\mathrm{RLWE}_{\boldsymbol{z}(X)}$ encryption of the blind rotation $X^{-b+\langle \vec{a},\vec{s}\rangle} \cdot \boldsymbol{w}(X)$. Here $\boldsymbol{w}(X)$ can be considered as a window filter and $X^{-b+\langle \vec{a},\vec{s}\rangle}$ makes it slice to the RLWE plaintext that corresponds to the LWE ciphertext of m, thus killing the LWE error e. It rests now RLWE noise due to multiplication $\diamond$. We then switch the key $\boldsymbol{z}(X)$ to $\boldsymbol{s}(X)$ in the RLWE encryption which produces again noise due to scalar multiplication in RLWE. These two noises are however controllable, which means they can be made small by choosing $\boldsymbol{z}$ and computation techniques (AP, GINX) . Finally the extraction of the last RLWE ciphertext encrypted by $\boldsymbol{s}(X)$ returns a LWE ciphertext of m with noise $e' < e$.

5.1 Blind Rotation

The main trick behind the bootstrap procedure is to work with ring polynomials. Indeed, for a polynomial $\boldsymbol{w}(X) = \sum_{i=0}^{N-1} w_i X^i$ and any $u \in \mathbb{Z}_q$, we have $\mathrm{coeff}_0(X^{-u}\boldsymbol{w}(X)) = \mathrm{coeff}_u(\boldsymbol{w}(X)) = w_u$. Recall that for any LWE ciphertext, we have $b - \langle \vec{a}, \vec{s}\rangle = \tilde{m} + e$ so that with $u = b - \langle \vec{a}, \vec{s}\rangle$, $\mathrm{coeff}_0(X^{-(b-\langle \vec{a},\vec{s}\rangle)}\boldsymbol{w}(X)) = w_{\tilde{m}+e} = w_{\tilde{m}}$. Thus, as soon as $\boldsymbol{w}(X)$ is well-chosen with coefficients that are equal by parts on subsets, one may recover a ciphertext of $\tilde{m}$ with smaller noise.

Product and Decomposition. Blind rotation is an operation that multiplies a given ring element $f \in R_Q$ by a monomial X^v, where the exponent $v = -b + \langle \vec{a}, \vec{s}\rangle$ is given by an LWE ciphertext $(\vec{a}, b) \in \mathbb{Z}_q^{n+1}$ of a message $m \in \mathbb{Z}_q$ encrypted under a secret key $\vec{s} \in \mathbb{Z}_q^n$. The output of the blind rotation is an RLWE encryption of $f \cdot X^v$ under a secret key $\mathbf{z} \in R_Q$. The operation is called *blind rotation* because it rotates the coefficients of f negacyclically (i.e with the sign negative). Consider the decomposition

$$X^{-b+\langle \mathbf{a},\mathbf{s}\rangle} \cdot f = \big(\prod_{i=1}^{n} X^{a_i \cdot s_i}\big) \cdot X^{-b} \cdot f.$$

Let ACC_0 be trivial encryption $\mathrm{RLWE}_{\mathbf{z}}(X^{-b} \cdot f) = (0, Y^{-b} \cdot f)$. Using the property of $\mathrm{RLWE} \diamond \mathrm{RGSW}$, we may recursively apply the external product to obtain

$$\mathrm{RLWE}_{\mathbf{z}}\big(X^{-b+\langle \mathbf{a},\mathbf{s}\rangle} \cdot f\big) = \mathrm{RGSW}_{\mathbf{z}}\big(\prod_{i=1}^{n} X^{a_i \cdot s_i}\big) \cdot ACC_0 = \big(\prod_{i=1}^{n} \mathrm{RGSW}_{\mathbf{z}}(X^{a_i \cdot s_i})\big) \cdot ACC_0.$$

Two different blind rotation algorithms were proposed in Ducas and Micciancio (2015); Chillotti et al. (2016). Many authors refer to the two algorithms as *AP blind rotation* and *GINX blind rotation* respectively. Both methods rely on the properties of RGSW ciphertext described above. This is where different techniques are employed to encode the elements in the product of the above formula: the AP algorithm relies on the decomposition of the elements a_i is basis B whereas the GINX algorithm takes advantage of the structure of binary secret keys.

AP blind rotation. For each $i = 1, \cdots, n$, decompose a_i in a basis $B \geq 2$ as $a_i = \sum_{j=0}^{K} a_{i,j} B^i$ where $K := \log_B q - 1$ and $0 \leq a_{i,j} \leq B - 1$. The decomposition may be rewritten as

$$\mathrm{RLWE}_{\mathbf{z}}\big(X^{-b+\langle \mathbf{a},\mathbf{s}\rangle} \cdot f\big) = \big(\prod_{i=1}^{n} \prod_{j=0}^{K} \mathrm{RGSW}_{\mathbf{z}}(X^{a_{i,j} \cdot B^i \cdot s_i})\big) \cdot ACC_0.$$

The blind rotation key is the public Bootstrap Key (BK) given by $BK^{\text{AP}}_{i,j,v} := \text{RGSW}_{\mathbf{z}}(X^{v \cdot B^j \cdot s_i})$. In practice, one needs to store these bootstrap keys. The procedure of AP blind rotation is described in Algorithm 1.

Algorithm 1 AP Blind Rotation

Require: $f, (\mathbf{a}, b), BK^{\text{AP}}_{i,j,v} = \text{RGSW}_{\mathbf{z}}(X^{v \cdot B^j \cdot s_i})$.

1. Set $\text{ACC} = (0, Y^{-b} \cdot f)$
2. **for** $1 \leq i \leq n$:
3. **for** $0 \leq j \leq K$:
4. **for** $0 \leq v < B$:
5. $\text{ACC} \leftarrow ACC \diamond BK^{\text{AP}}_{i,j,v}$
6. Return ACC

GINX Blind Rotation. This blind rotation is known to be more efficient than AP when the secret key $\mathbf{s} \in \mathbb{Z}_q$ is set to a binary or ternary vector, however its performance degrades when using larger secret keys. For each secret key element $s_i \in \mathbb{Z}_q$ for $i \in \{1, \cdots, n\}$, we decompose the key as $s_i = \sum_{j=1}^{|U|} u_j \cdot s_{i,j}$ where $s_{i,j} \in \{0, 1\}$ and $U \subset \mathbb{Z}_q$ is an appropriately chosen subset of $\mathbb{Z}_q$. For simplicity, we consider the case that $s_i \in \{0, 1\}$. Then for all $i \in \{1, \cdots, n\}$, we have that $X^{a_i \cdot s_i} = 1 + (X^{a_i} - 1)s_i$ which implies that

$$\text{RLWE}_{\mathbf{z}}\big(X^{-b+\langle \mathbf{a}, \mathbf{s} \rangle} \cdot f\big) = \text{RGSW}_{\mathbf{z}}\big(\prod_{i=1}^{n}(1 + (X^{a_i} - 1) \cdot s_i)\big) \cdot \text{ACC}_0$$

Using this technique, the blind rotation key is $BK^{\text{GINX}}_i = \text{RGSW}_{\mathbf{z}}(s_i)$. The procedure of GINX blind rotation is described in Algorithm 2.

Algorithm 2 GINX Blind Rotation

Require: $f, (\mathbf{a}, b), BK^{\text{GINX}}_i = \text{RGSW}_{\mathbf{z}}(s_i)$.

1. Set $\text{ACC} = (0, Y^{-b} \cdot f)$
2. **for** $1 \leq i \leq n$:
3. $\text{ACC} = ACC + (X^{a_i} - 1) \cdot (\text{ACC} \diamond BK^{\text{GINX}}_i)$
4. Return ACC

Observe that in line (3), if $s_i = 0$, the second addendum is ignored since it gives an encryption of 0 and the value stored by the accumulator stays the same. If $s_i = 1$, then $\text{ACC} \diamond \text{RGSW}_{\mathbf{z}}(1)$ is equal to ACC and the accumulator is updated to $X^{a_i} \cdot \text{ACC}$.

5.2 Key Switching in RLWE

Key switching operation converts a ciphertext $\text{RLWE}_{\mathbf{z}_1}(\mathbf{m})$ of a message $\mathbf{m} \in R_Q$ encrypted by a secret key $\mathbf{z}_1 \in R_Q$ to a ciphertext $\text{RLWE}_{\mathbf{z}_2}(\mathbf{m})$ encrypted by a new secret key $\mathbf{z}_2 \in R_Q$. There are different variants of the key switching technique (e.g see for details). We describe the following method. Let $(\mathbf{a}, \mathbf{b})$ be an RLWE encryption of $\mathbf{m}$ by the secret key $\mathbf{z}_1$. To offer

a space/time trade-off, we write $\mathbf{a}$ in a basis $B \geq 2$ as follows:

$$\mathbf{a} = \sum_{j=0}^{K} \alpha_j B^j \quad \text{with } \alpha_j(X) = \sum_{i=0}^{N-1} a_{i,j} X^i, 0 \leq a_{i,j} \leq B-1 \quad \text{and } K := \log_B Q - 1$$

The KeySwitching (KS) keys are given for all $j = 0, \cdots, K$ by $\text{RLWE}_{\mathbf{z}_2}(B^j \cdot \mathbf{z}_1) := (KS_j^{\mathbf{a}}, KS_j^{\mathbf{b}})$. Now rewrite the ciphertext $\mathbf{m} = \mathbf{b} - \mathbf{a} \cdot \mathbf{z}_1 - \mathbf{e}$ as

$$\mathbf{m} = \big(\mathbf{b} - \sum_{j=0}^{K} \alpha_j \cdot KS_j^{\mathbf{b}}\big) + \big(\sum_{j=0}^{K} \alpha_j \cdot KS_j^{\mathbf{a}} \cdot\big)\mathbf{z}_2 - \big(\sum_{j=0}^{K} \alpha_j \cdot \mathbf{e}_j + \mathbf{e}\big)$$

where $\mathbf{e}$ is the error in the RLWE encryption of $\mathbf{m}$ under the key $\mathbf{z}_1$ and $\mathbf{e}_j$ those in the $\text{RLWE}_{\mathbf{z}_2}(B^j \cdot \mathbf{z}_1)$. The procedure of key switching is described in Algorithm 3. Note that $(KS_j^{\mathbf{a}}, KS_j^{\mathbf{b}})$, for $j = 0, \cdots, K$, is a public switching key. We see explicitly that the key switching error is equal to the sum of the error of $\text{RLWE}_{\mathbf{z}_1}(\mathbf{m})$ and all $\text{RLWE}_{\mathbf{z}_2}(B^j \cdot \mathbf{z}_1)$ which calls for caution when choosing the various parameters for encryption.

Algorithm 3 Key Switching in RLWE

Require: $(\mathbf{a}, \mathbf{b}) = \text{RLWE}_{\mathbf{z}_1}(\mathbf{m}), \mathbf{z}_2, (KS_j^{\mathbf{a}}, KS_j^{\mathbf{b}}) := \text{RLWE}_{\mathbf{z}_2}(B^j \cdot \mathbf{z}_1)$

1. Write $\mathbf{a} = \sum_{j=0}^{K} \alpha_j B^j$ where $\alpha_j = \sum_{i=0}^{N-1} a_{i,j} X^i$ such that $|a_{i,j}| \leq B-1$
2. Return $(\mathbf{a}', \mathbf{b}') = \big(-\sum_{j=1}^{K} \alpha_j \cdot KS_j^{\mathbf{a}}, \mathbf{b} - \sum_{j=1}^{K} \alpha_j \cdot KS_j^{\mathbf{b}}\big)$

5.3 Extraction

The final step of the boostrap procedure deals with the extraction of a ciphertext of the coefficient of order 0 of a ciphertext in RLWE. To define such operation, consider a ciphertext $RLWE_{\boldsymbol{s}(X)}(\boldsymbol{m}(X)) = (\boldsymbol{a}(X), \boldsymbol{b}(X)) \in \mathcal{R}_Q$. We have: $\boldsymbol{b}(X) = \boldsymbol{a}(X) \cdot \boldsymbol{s}(X) + \tilde{\boldsymbol{m}}(X) + \boldsymbol{e}(X)$ so the coefficients of the polynomials may be identified as

$$b_i = \sum_{k=0}^{N-1} a_{i-k} s_k + \tilde{m}_i + e_i = \langle \iota_i(\boldsymbol{a}), s \rangle + \tilde{m}_i + e_i$$

where $\iota_i(\boldsymbol{a})$ is the ith anti-cyclic permutation of coefficients of $\boldsymbol{a}$. Since $\boldsymbol{e}(X)$ is small, so is e_i and $(\iota_i(\boldsymbol{a}), b_i)$ is a LWE_s encryption of m. Therefore, the extraction operation of the bootstrap procedure only requires to compute the anti-cyclic permutation of coefficients of $\boldsymbol{a}$.

6. Conclusion

If introduced on a large scale and in real-world scenarios, fully homomorphic encryption (FHE) would have enormous consequences for people's privacy, allowing them to use remote services hosted on untrusted servers, without disclosing personal informations. Over the years, homomorphic encryption has been a very active field for research, and this has led to a long list of works and improvements, e.g. Brakerski and Vaikuntanathan (2014b,a); Alperin-Sheriff and Peikert (2014). This comprised both theoretical works that put forth new ideas and constructions, and implementation works (Wang et al., 2012; Jung et al., 2021) that optimized existing constructions with the goal of achieving the best possible efficiency.

Acknowledgements This work was done during the SEME (*Semaine d'Étude Mathématiques Entreprise*) in Rennes (May 2nd - May 6th 2022) organized in collaboration with AMIES (*Agence pour les Mathématiques en Interaction avec les Entreprises et la Société*). The authors would like to thank the organizers of Université of Rennes 1 as well as the different supervisors from Ravel Technologies: Florian Méhats, Mohammed Lemou and Philippe Chartier.

References

Alperin-Sheriff, J. and Peikert, C. (2014). Faster bootstrapping with polynomial error. In Garay, J. A. and Gennaro, R., editors, *Advances in Cryptology – CRYPTO 2014*, pages 297–314, Berlin, Heidelberg. Springer Berlin Heidelberg.

Brakerski, Z. and Vaikuntanathan, V. (2014a). Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on computing*, 43(2):831–871.

Brakerski, Z. and Vaikuntanathan, V. (2014b). Lattice-based fhe as secure as pke. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 1–12.

Brigham, E. O. (1988). *The fast Fourier transform and its applications*. Prentice-Hall, Inc.

Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. (2016). Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Cheon, J. H. and Takagi, T., editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg. Springer Berlin Heidelberg.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654.

Ducas, L. and Micciancio, D. (2015). Fhew: Bootstrapping homomorphic encryption in less than a second. In Oswald, E. and Fischlin, M., editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA. Association for Computing Machinery.

Jung, W., Kim, S., Ahn, J. H., Cheon, J. H., and Lee, Y. (2021). Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 114–148.

Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 1–23. Springer.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Stehlé, D., Steinfeld, R., Tanaka, K., and Xagawa, K. (2009). Efficient public key encryption based on ideal lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 617–635. Springer.

Wang, W., Hu, Y., Chen, L., Huang, X., and Sunar, B. (2012). Accelerating fully homomorphic encryption using gpu. In *2012 IEEE conference on high performance extreme computing*, pages 1–5. IEEE.