FUNDAÇÃO GETULIO VARGAS
SCHOOL OF APPLIED MATHEMATICS

RENER DE SOUZA OLIVEIRA

A SURVEY ON FULLY HOMOMORPHIC ENCRYPTION WITH STATISTICAL APPLICATIONS

Rio de Janeiro
2022

RENER DE SOUZA OLIVEIRA

A SURVEY ON FULLY HOMOMORPHIC ENCRYPTION WITH STATISTICAL APPLICATIONS

Bachelor dissertation presented to the School of Applied Mathematics (FGV/EMAp) to obtain the Bachelor's degree in Applied Mathematics.

Areas of Study: Cryptography, Machine Learning

Advisor: Rodrigo dos Santos Targino

Rio de Janeiro

2022

Ficha catalográfica elaborada pela BMHS/FGV

Oliveira, Rener de Souza
A survey on fully homomorphic encryption with statistical applications/ Rener de Souza Oliveira. – 2022.
52f.

Bachelor Dissertation (Undergraduation) – School of Applied Mathematics.

Advisor: Rodrigo dos Santos Targino.
Includes bibliography.

1. Private Machine Learning. 2. Fully Homomorphic Encryption. I. Targino, Rodrigo dos Santos. II. School of Applied Mathematics. III. A survey on fully homomorphic encryption with statistical applications

I dedicate this dissertation to my grandfather Valter (in memoriam) and my great-aunt Varly (in memoriam), who were my inspiration for humility and simplicity and always believed in my studies. I know they are proud of me now!

Acknowledgements

I thank God for giving me the strength to finish this work; my family, who always supported my dream to study math; FGV's Center for the Development of Mathematics and Science (CDMC), who made my dream possible with close assistance and financial support; my advisor Rodrigo Targino, who proposed this work theme and guided me through the whole process; my algebra teacher Luciano Guimarães, who helped me learn the theoretical bases of this work and all my teachers from School of Applied Mathematics (EMAp), who always worked thinking in our learning process.

Abstract

The amount of data generated by individuals and enterprises is growing exponentially over the last decades, which empowers the use of machine learning methods since, for statistical purposes, the more data a model can have access to, the more accurately it will predict or represent reality. The problem emerges when the model must deal with sensitive data such as medical records, financial history, or genomic data, in which additional care must be taken in order to protect the privacy of data owners. Encrypting sensitive data might appear a good solution at first sight, but it can considerably limit the ability to do statistical analysis. This work is a survey on *Fully Homomorphic Encryption (FHE)*, a special kind of cryptography scheme that still permits some machine learning methods to run over encrypted data, while it has strong mathematical guarantees of privacy protection.

Keywords: Machine Learning, Cryptography, Fully Homomorphic Encryption, Logistic Regression.

Resumo

A quantidade de dados gerados por pessoas e empresas está crescendo exponencialmente nas últimas décadas, o que potencializa o uso de métodos de aprendizado de máquina, pois, para fins estatísticos, quanto mais dados um modelo tiver acesso, mais preciso ele será na previsão ou representação da realidade. O problema surge quando o modelo precisa lidar com dados sensíveis, como registros médicos, histórico financeiro ou dados genômicos, nos cuidado adicional deve ser tomado para proteger a privacidade dos proprietários dos dados. Criptografar tais dados, pode parecer uma boa solução à primeira vista, mas pode limitar consideravelmente a capacidade de fazer análises estatísticas. Este trabalho é uma pesquisa sobre *Criptografia Completamente Homomórgica* um tipo especial de esquema de criptografia que permite a execução de alguns modelos de aprendizado sobre dados criptografados, enquanto, ao mesmo tempo, mantém fortes garantias matemáticas de proteção de privacidade.

Palavras-chave: Aprendizado de Máquinas, Criptografia, Criptografia Totalmente Homomórfica, Regressão Logística.

Contents

1 INTRODUCTION 9

2 ALGEBRAIC REVIEW 11
2.1 Basic structures 11
2.2 Homomorphisms and Quotient Rings 12
2.3 Cyclotomic polynomials 15
2.4 Lattices 19
2.4.1 Lattice Problems 20
2.4.2 Ring Learning with Errors 21

3 FULLY HOMOMORPHIC ENCRYPTION 23
3.1 Privacy Homomorphisms 23
3.1.1 Requirements and Limitations 24
3.2 Bootstrappable encryption 25
3.2.1 Overview and Bootstrapping 25
3.2.2 An integer scheme 27
3.2.3 Practical considerations and further research 29
3.3 FHE over the complex numbers 29
3.3.1 Encoding and Decoding 31
3.3.2 Encryption, Decryption, and Relinearization 33
3.3.3 Approximate Bootstrapping 34

4 PRIVATE LOGISTIC REGRESSION 35
4.1 Statistical Review 35
4.2 Homomorphic Training 36
4.2.1 Ciphertext packing and data representation 38
4.2.2 Batch Inner Product 38
4.3 Data Applications 41

5 CONCLUSIONS AND FURTHER WORK 43

References 44

APPENDIX 47

APPENDIX A – AN IDEAL LATTICE SCHEME 48

A.1 Initial definitions 48
A.2 Abstract construction 48
A.3 Concrete construction using ideal lattices 50

1 Introduction

As the amount of data generated by individuals and companies exponentially increases over the years, machine learning (ML) methods become very suitable and permit various impactful applications: Cellphones lock systems by facial recognition, automatic diagnosis in medical exams, credit scoring in the banking industry, estimated time arrival in transport companies, or social media ads recommendation.

Although, for statistical purposes, the more data, the better, additional care must be taken when such information is sensitive, such as financial transactions, educational records, genomic data, and socioeconomic attributes. In the last years, regulators have raised concerns about data privacy, releasing and updating laws protecting the rights of data owners, for example, General Data Protection Regulation, GDPR (COUNCIL OF EUROPEAN UNION, 2016).

This work is an intersection of two areas: Statistics and Cryptography, although they might appear completely unrelated at first sight, such a combination can yield a possible solution for dealing with machine learning applications that have to use sensitive data. A cryptography scheme might empower data privacy since, after encryption, the original data is inaccessible by untrusted parties, But then the data becomes unavailable for executing statistical models too!

A natural question emerges: Can we still compute something on a dataset after encrypting it? Although we're discussing a contemporaneous, this question was first proposed in a 1978 article (RIVEST; ADLEMAN; DERTOUZOS, 1978) by the creators of the RSA cryptography system, one of the most famous and important encryption schemes.

In a classical public-key cryptography scheme, there is a secret key sk, kept only by those who can see the original data, and a public key pk that can be publicly shared and it's used for encryption. Besides that, there are two functions: $\mathrm{Enc}(\mathrm{pk}, \cdot)$ for encryption and $\mathrm{Dec}(\mathrm{sk}, \cdot)$ for decryption. Suppose we have two messages m_1, m_2 (it can be our sensitive data) in a given space and a bivariate function $f(\cdot)$ in this space (an ML training algorithm, for example). The problem proposed by the article is the existence of an alternative function f' in the encrypted domain, the set generated by all possible encrypted messages, such that:

$$\mathrm{Dec}(\mathrm{sk}, f'(\mathrm{Enc}(\mathrm{pk}, m_1), \mathrm{Enc}(\mathrm{pk}, m_2)) = f(m_1, m_2)$$

The above equation says that we can use f' in the encrypted messages, and after decryption, the result would be the same as applying f directly to the plain messages. A cryptography scheme that satisfies this relation is called *Fully Homomorphic Encryption (FHE).*

The existence of an FHE scheme remained an open problem until 2009, in Gentry's Ph.D. thesis (GENTRY, 2009). He proposed a scheme that satisfied the above relation for f being an addition or multiplication function. Although the question was mathematically answered, Gentry's scheme was very inefficient in practice due to the computational cost of encrypted operations and storage space used by encrypted messages.

The subsequent papers after his work focused on proposing better and more practical schemes. A parallel research group has also emerged, studying how to modify training or prediction ML algorithms to the FHE setup, so that they could be executed over a completely encrypted dataset.

This work has three main chapters: Chapter 2 is an abstract algebra review the theory needed for cryptography; Chapter 3 goes through FHE research, formally defining Rivest's original problem, explaining Gentry's solution and presenting a more recent and practical scheme, known as HEAAN (CHEON; KIM, et al., 2017). Chapter 4 is an application of HEAAN scheme to the logistic regression training algorithm, with experimental results on a real dataset.

2 Algebraic Review

In this chapter, we shall discuss the algebraic theoretical background that grounds cryptography schemes. Section 2.1 defines basic algebraic structures such as groups and rings. Section 2.2 goes through homomorphism properties and quotient rings, two important concepts. Section 2.3 explains cyclotomic polynomials, a special type of polynomial that will be used in the next section and chapter. Section 2.4 defines lattices, an algebraic structure that has interesting computationally hard problems that will guarantee the security of our encryption schemes.

2.1 Basic structures

Definition 2.1.1 (Group)**.** A non-empty set G is called a group under the operation $*$, if it satisfies the following axioms:

a) (Closure) G is closed under $*$, i.e, for all $a, b \in G$, the result $a * b$ is also in G;
b) (Associativity) $(a * b) * c = a \times (b * c)$, for all $a, b, c \in G$;
c) (Existence of identity) There exists $e \in G$ such that $a * e = e * a = a$ for all $a \in G$;
d) (Existence of inverse) There exists $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$ for all $a \in G$;

   We shall denote the group as $(G, *)$. If, in addition to the above axioms, the group also satisfies the next property, it is called an **abelian group**.

e) (Commutativity) For all $a, b \in G$, $a * b = b * a$.

Examples of abelian groups[1]: $(\mathbb{Z}, +)$ with identity element 0 and inverse of a being $-a$. $(\mathbb{Q}^\times, \cdot)$ with identity 1 and inverse $a^{-1} = 1/a$.

Definition 2.1.2 (Subgroup)**.** Let $(G, *)$ be a group, $H \subseteq G$ is a subgroup of G if it satisfies the following:

- $H \neq \varnothing$;
- H is closed under G operation: $a, b \in H \Rightarrow a * b \in H$;
- H is closed under taking inverses: $a \in H \Rightarrow a^{-1} \in H$.

Associativity is inherited by G and the existence of identity element can be deduced by the above properties (TENGAN; MARTINS, 2017).

[1] $\mathbb{Q}^\times = \mathbb{Q}/\{0\}$

Definition 2.1.3 (Ring)**.** A set R is called a (commutative) *ring* if it has two operations: addition $(+)$ and multiplication $(\times)$ satisfying the following properties:

a) R is an abelian group under addition

b) (multiplicative associativity) $(a \times b) \times c = a \times (b \times c)$, for all $a,\ b,\ c \in R$;

c) (distributivity) $a \times (b + c) = a \times b + a \times c$ for all $a,\ b,\ c \in R$;

d) (multiplicative commutativity) $a \times b = b \times a$ for all $a,\ b \in R$;

e) (multiplicative identity) There exists an element 1, such that, $1 \times a = a \times 1 = a$, for all $a \in R$.

Property d) is not mandatory for general non-commutative rings, but in this text, the term "ring" indicates a commutative one. Some examples of rings: $\mathbb{Z}$, ring of integers; $\mathbb{Z}_q$, the integers mod-q and $\mathbb{Z}[X]$, set of polynomial with integer coefficients. The latter is a special ring that will be the base of our encryption schemes.

Definition 2.1.4 (Ideal)**.** Given a ring R, a subset $I \subseteq R$ is called an *ideal* of R if its an additive subgroup of R and satisfies:

$$r \times i \in I \text{ for all } r \in R,\ i \in I$$

Every ideal has a set of generators $G = \{g_1, \dots, g_n\}$, such that

$$I = (G) = \left\{ \sum_{i=1}^{n} g_i r_i | r_i \in R \right\}$$

G is called a *basis* of I if its elements are linearly independent. I is a *principal ideal* if G has just a single element.

Two ideals I and J are called *relatively prime ideals* of a ring R if $I + J = R$, where $I + J = \{i + j | i \in I, j \in J\}$.

An example of (principal) ideal of the ring of integers is $(2) = \{2r | r \in \mathbb{Z}\}$, so-called even numbers. One can easily verify that it satisfies all the above properties.

Definition 2.1.5 (Fields)**.** A Field F is a commutative ring, where for every non-zero element $a \in F$ there exists $a^{-1} \in F$ such that $aa^{-1} = 1$, being 1 the ring unity element.

$\mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$ are examples of fields, but $\mathbb{Z}$ is not.

2.2 Homomorphisms and Quotient Rings

Definition 2.2.1 (Ring homomorphism)**.** Let R and S be rings. A map $\varphi : R \to S$ is called a ring homomorphism if it preserves addition/multiplication operations and the unity element:

$$\begin{aligned}\varphi(a+b) &= \varphi(a)+\varphi(b)\\ \varphi(a\times b) &= \varphi(a)\times\varphi(b)\\ \varphi(1_R) &= 1_S\end{aligned}$$

Similarly, we can define group homomorphisms that will preserve their unique underlying operation. $\varphi(0_R) = 0_S$ is a consequence of group properties[2] (TENGAN; MARTINS, 2017).

Some ring homomorphisms examples:

a) Integer modular reduction operation:

$$\begin{aligned}\varphi &: \mathbb{Z} \to \mathbb{Z}_q\\ \varphi(a) &= a(\text{mod } q)\end{aligned}$$

b) Polynomial evaluation at $a \in R$:

$$\begin{aligned}\varphi_a &: R[X] \to R\\ \varphi_a(p(X)) &= p(a),\end{aligned}$$

where $R[X]$ is the set of polynomials with coefficients in a ring R

Definition 2.2.2 (Kernel)**.** Let $\varphi : R \to S$ be a ring homomorphism. The kernel of φ is the pre image of 0:

$$\ker\varphi = \{r \in R | \varphi(r) = 0\}$$

In the above homomorphism examples, the kernel of modular reduction by q is the set of multiples of q. The kernel of polynomial evaluation at a is the set of all polynomials in $p(X) \in R[X]$ such that a is one of its roots, i.e., $p(a) = 0$.

Proposition 2.2.1. *If $\varphi : R \to S$ is a ring homomorphism, then $\ker\varphi$ is an ideal of R. (MCIVOR, 2016)*

Proof. We have to prove that $\ker\varphi$ is an additive subgroup of R and that it satisfies:

$$r \times i \in \ker\varphi, \forall r \in R, i \in \ker\varphi,$$

which means $\varphi(r \times I) = 0$. Indeed, that's the case: $\varphi(r \times i) = \varphi(r) \times \varphi(i) = \varphi(r) \times 0 = 0$. Now let's prove the three subgroup axioms (Def. 2.1.2) for the addition operation, assuming $\varphi(0) = 0$ as a fact.

- $\ker\varphi \neq \varnothing$: Since $\varphi(0) = 0$, $\ker\varphi$ has at least 0 as element;

[2] Here 0_R denotes the identity element of R when viewed as an additive group

- $\ker\varphi$ is closed under $+$: Take $a, b \in \ker\varphi$. Using this fact and additive homomorphism property, we have $\varphi(a+b) = \varphi(a) + \varphi(b) = 0 + 0 = 0 \Rightarrow a + b \in \ker\varphi$;
- $\ker\varphi$ is closed under (additive) inverse: If $a \in \ker\varphi$, $\varphi(a) = 0$ then $0 = \varphi(0) = \varphi(a + (-a)) = \varphi(a) + \varphi(-a) = \varphi(-a)$. So $\phi(-a) = 0$ impling $-a \in \ker\varphi$

$\square$

Definition 2.2.3 (Isomorphism)**.** A map $\varphi : R \to S$ is called an isomorphism if it is a bijective homomorphism. In this case, we denote $R \cong S$.

A famous example of group isomorphism is

$$\varphi : \mathbb{C} \to \mathbb{R}^2$$
$$\varphi(a+bi) = (a, b),$$

if we define multiplication in $\mathbb{R}^2$ as $(a, b)(c, d) = (ac - bd, ad + bc)$ it become a ring isomorphism.

Definition 2.2.4 (Cosets)**.** Let H be a subgroup of an abelian group $(G, \cdot)$. A translation of H,

$$\lambda_g(H) = \{g \cdot h | \ h \in H\},$$

is called a *coset* of H.

We denote the set of all cosets of H by:

$$G/H \overset{\text{def}}{=} \{g \cdot H | g \in G\}$$

Since a ring R can be viewed as an additive abelian group and an ideal $I \subseteq R$ as a subgroup. We extend the coset notion to rings as:

$$R/I = \{I + r | r \in R\}$$

For example: Take the ring of integers $\mathbb{Z}$ and the multiples of three ideal $I = (3)$. The cosets of I are translations the translations of $I = \{\ldots, -3, 0, 3, 6, \ldots\}$:

$$0 + I = \{\ldots, -3, 0, 3, 6, \ldots\}$$
$$1 + I = \{\ldots, -2, 1, 4, 7, \ldots\}$$
$$2 + I = \{\ldots, -1, 2, 5, 8, \ldots\}$$

Summing other elements of $\mathbb{Z}$ to I will result in one of the above sets. We can give an algebraic structure to these cosets and transform them into rings!

For each coset, we choose a representative r for $\bar{r} = r + I$, and we can define addition, multiplication and prove the ring axioms for these operations, characterizing the *Quotient Ring* (MCIVOR, 2016). When working with a quotient ring in practice, instead of summing two (possibly infinite) cosets, we elect some representatives and operate on them; In the above example, $\mathbb{Z}/(3) = \{\bar{0}, \bar{1}, \bar{2}\}$, and $\bar{1} + \bar{2} = \bar{3}$, which is equivalent to $\bar{0}$.

A common notation for integers modulo q, for example, is $\mathbb{Z}/q\mathbb{Z}$ which is actually the set of cosets of the ideal of multiples of n. It's also common to choose the representatives as $\{0, 1, \ldots, q-1\}$, but there is nothing stopping us from choosing for example $\mathbb{Z} \cap (-q/2, q/2]$; in this case the representative are centered in 0 for an odd q, e.g., $\mathbb{Z}/3\mathbb{Z} = \{\overline{-1}, \bar{0}, \bar{1}\}$. That's exactly the choice of representatives we make in Section 3.3, where the goal is to minimize the size of messages and ciphertext for a better performance of the encryption scheme.

The following theorem is perhaps one of the most important theorems of group and ring theory and provides an alternative way of looking at quotient rings of kernels.

Theorem 2.2.1 (First Ring Homomorphism Theorem)**.** *Let R, S be rings, if $\varphi : R \to S$ is a ring homomorphism, and $\varphi(R)$ its image, then $R/\ker\varphi \cong \varphi(R)$. For proof, refer to (MCIVOR, 2016).*

Returning to the example of $\mathbb{Z}/q\mathbb{Z}$, take the modular reduction homomorphism $\varphi(a) = a \pmod{q}$, its kernel is the ideal $q\mathbb{Z}$ of multiples of q. What the above theorem is saying is that the image $\varphi(\mathbb{Z}) = \mathbb{Z} \pmod{q} = \{0, 1, 2\}$ is isomorphic to the quotient ring $\mathbb{Z}/q\mathbb{Z}$.

Let $\mathbb{Z}[X]$ be the polynomials with integer coefficients and $f(x) = x^n + 1$, a monic polynomial with degree n. Define the homomorphism $\varphi : \mathbb{Z}[X] \to \mathbb{Z}[X]$, that take a polynomial $p(x) \in \mathbb{Z}[X]$ and returns $r(x)$, the remainder of the polynomial division $p(x)/f(x)$, using grade-school division algorithm. $\ker\varphi$ is the set of polynomials with remainder equal 0, i.e., the set of multiples of $x^n + 1$ in $\mathbb{Z}[X]$, which is the ideal generated by f (Prop. 2.2.1). The Homomorphism Theorem affirms that $\mathbb{Z}[X]/(f(x)) \cong \varphi(\mathbb{Z}[X])$, so we can represent the set of cosets of $(x^n + 1)$ simply as the image of φ: the set of remainders in the division by $f(x)$, which can be viewed as the integer polynomials with degree less than n. This will be a central ring in the encryption schemes we shall further define.

2.3 Cyclotomic polynomials

In this section, we define and review some properties of cyclotomic polynomials, which will play a central role in the homomorphic cryptography setup.

Definition 2.3.1 (Roots of unity)**.** The n^{th} roots of unity are the solution set of the equation $x^n - 1 = 0$ in the field of complex number $\mathbb{C}$:

$$\sqrt[n]{1} = \{\zeta_n^k; k = 0, 1, \dots, n-1\},$$

where[3] $\zeta_n = \exp(2\pi i/n)$

In the complex plane, these roots are distributed over the unitary circumference and equally separated by an angle of $2\pi/n$. Figure 1 show the example of the 8^{th} roots of unity.

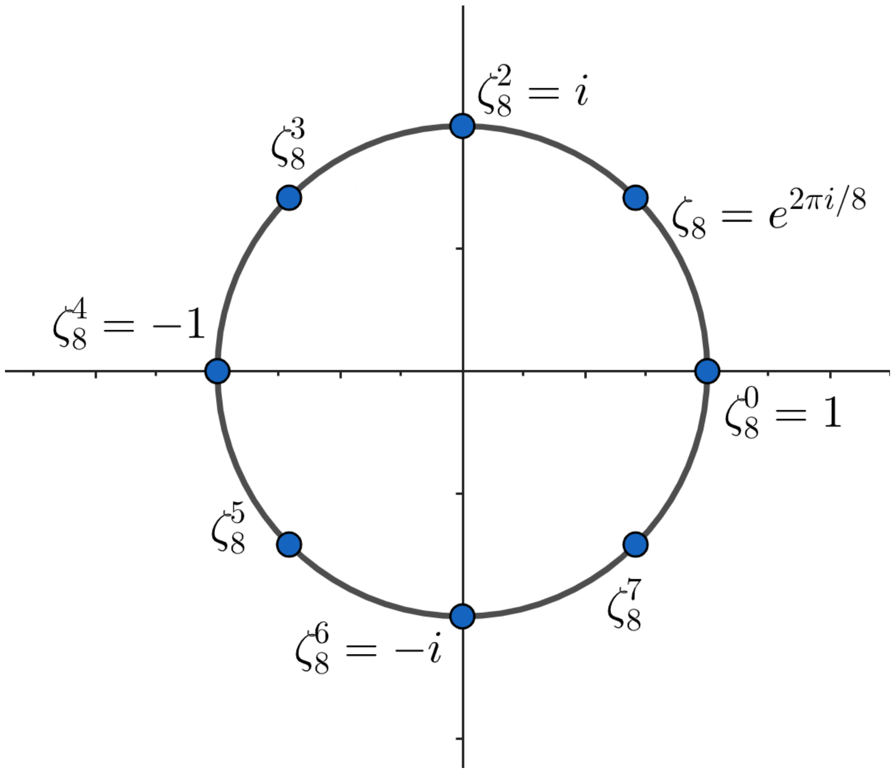


Figure 1 – 8^{th} roots of unity

Definition 2.3.2 (Primitive roots of unity)**.** (CORN; MANZOOR; DASH, 2016) The n^{th} primitive roots of unity are:

$$\{\zeta \in \mathbb{C}; \zeta^n = 1 \text{ and } \zeta^k \neq 1, \forall\ k < n\},$$

for positive integers k. They are the subset of n^{th} roots of unity which are not k^{th} roots of unity, for all $k < n$. They can be alternatively defined as:

$$\{\zeta_n^k; 1 \leq k \leq n, \gcd(k, n) = 1\}$$

From the Figure 1 example, the 8^{th} primitive roots of unity are $\zeta_8, \zeta_8^3, \zeta_8^5, \zeta_8^7$.

[3] In Euler's notation $\exp(i\theta) = \cos\theta + i \cdot \sin\theta$

Definition 2.3.3 (Cyclotomic polynomial)**.** The *n-th cyclotomical polynomial* is defined as:

$$\Phi_n(x) = \prod_{\substack{1\le k\le n\\ \gcd(k,n)=1}}^{n} (x-\zeta_n^k)$$

Notice that its roots are the n^{th} primitive roots of unity.

Using the definition, we can derive some cyclotomical polynomials, for example $\Phi_1(x) = x-1$ trivially, and $\Phi_2(x) = x+1$, since from the 2^{nd} roots of unity -1 and $+1$, only -1 are primitive ones.

The 3^{rd} primitive roots of are ζ_3^1 and $\zeta_3^2 = \overline{\zeta_3^1}$, then:

$$\begin{aligned}\Phi_3(x) &= (x-\zeta_3^1)(x-\zeta_3^2)\\ &= x^2 - x(\zeta_3^1+\zeta_3^2) + \zeta_3^1\zeta_3^2\\ &= x^2 - x(\zeta_3^1+\overline{\zeta_3^1}) + e^{2\pi i(1+2)/3}\\ &= x^2 - x(2\cdot\cos(2\pi/3)) + e^{2\pi i} \qquad (2.1)\\ &= x^2 + x\left(2\cdot\frac{1}{2}\right)+1\\ &= x^2+x+1\end{aligned}$$

The equality (1) holds due to the sum of complex conjugates being equal to two times the real part since we cancel out the imaginary terms.

Theorem 2.3.1. *For all positive integers n we have:*

$$x^n - 1 = \prod_{d|n} \Phi_d(x)$$

The above theorem provides a analytical formula to recursively generate the cyclotomic polynomials:

$$\Phi_n(x) = \frac{x^n-1}{\prod\limits_{\substack{d|n\\ d\neq n}} \Phi_d(x)}$$

We can use it to derive simpler, and non-recursive expressions for particular interesting cases:

Corollary 2.3.1 (Prime Cyclotomical Polynomial)**.** *If p is a prime number, then:*

$$\Phi_p(x) = x^{p-1} + x^{p-2} + \ldots + x + 1$$

Proof. By the previous theorem, and the fact that the only $d < p$ satisfying $d|n$ is $d = 1$:

$$\Phi_p(x) = \frac{x^p - 1}{\prod\limits_{\substack{d|p \\ d \neq p}} \Phi_d(x)} = \frac{x^p - 1}{\Phi_1(x)} = \frac{x^p - 1}{x - 1}$$

The polynomial division algorithm concludes that such a division actually results in what is claimed in the corollary.

An easy way to assert it, is by multiplying the divisor $x - 1$ by $\sum\limits_{i=0}^{p-1} x^i$ and checking if it is equal to $x^p - 1$:

$$\begin{aligned}(x-1)\left(\sum_{i=0}^{p-1} x^i\right) &= \sum_{i=0}^{p-1} x^{i+1} - \sum_{i=0}^{p-1} x^i \\ &= \sum_{i=0}^{p-1} x^{i+1} - x^i \\ &= x^{(p-1)+1} - x^0 \\ &= x^p - 1\end{aligned}$$

□

The following formula will be crucial to instantiate plaintext and ciphertext spaces in the encryption schemes.

Corollary 2.3.2 (Power of Two Cyclotomic Polynomial)**.** *If $M = 2^n$, for a given positive integer n, then:*

$$\Phi_M(x) = x^{M/2} + 1$$

Proof. We'll proceed by strong induction in n (HERSTEIN, 1996), proving an equivalent statement[4]:

$$\Phi_{M_n}(x) = \frac{x^{M_n} - 1}{x^{M_n/2} - 1} \tag{2.2}$$

We're denoting 2^n by M_n to have a cleaner notation of nested exponents; hence $M_n/2 = M_{n-1}$.

(Base case) For $n = 1$, $\Phi_{M_1}(x) = \Phi(2) = x + 1$, as we derived before. This is the LHS of 2.2. The RHS is $\frac{x^2-1}{x-1} = \frac{(x+1)(x-1)}{x-1} = x + 1$, so the formula is valid in this case.

(Inductive Hypothesis) Assume that 2.2 hold for M_k for all $k < n$ positive integers.

(Inductive Step) Let's deduce that it also holds for M_n: By Theorem 2.3.1, we have:

$$\Phi_{M_n}(x) = \frac{x^{M_n} - 1}{\prod\limits_{\substack{d|M_n \\ d \neq M_n}} \Phi_d(x)}$$

[4] One can easily verify that $\dfrac{x^M - 1}{x^{M/2} - 1} = x^{M/2} + 1$ using the polynomial division algorithm for example, or multiplying the divisor with the quotient

Notice that $\{d \in \mathbb{Z}^+; d|M_n\} = \{M_0, M_1, \ldots, M_{n-1}\}$, i.e., the divisors of M_n are the powers of two with exponent less than n. Then our expression becomes:

$$\Phi_n(x) = \frac{x^{M_n} - 1}{\Phi_{M_0}\Phi_{M_1}\ldots\Phi_{M_{n-1}}}$$

Using $\Phi_{M_0}(x) = x - 1$ and the inductive hypothesis, we can rewrite the denominator as:

$$\begin{aligned}\Phi_{M_0}\Phi_{M_1}\ldots\Phi_{M_{n-1}} &= (x-1)\cdot\frac{x^{M_1}-1}{x^{M_0}-1}\cdot\frac{x^{M_2}-1}{x^{M_1}-1}\cdot\ldots\cdot\frac{x^{M_{n-1}}-1}{x^{M_{n-2}}-1}\\ &= x^{M_{n-1}} - 1,\end{aligned}$$

since that's the only term left after canceling out left numerators with right denominators. We then conclude:

$$\Phi_{M_n}(x) = \frac{x^{M_n} - 1}{x^{M_{n-1}} - 1} = x^{M_n/2} + 1$$

□

Another interesting property of cyclotomic polynomials, which is quite impressive, is the following theorem:

Theorem 2.3.2. *"For any positive integer n we have $\Phi_n(x) \in \mathbb{Z}[x]$, That is, $\Phi_n(x)$ is a polynomial with integer coefficients."(SUN, 2013)*

We defined these polynomials as products of a bunch of complex numbers, but the coefficients end up in the integers! Such a beautiful property, together with the fact the cyclotomic polynomials are monic (SUN, 2013), will allow us to determine later quotient rings like $\mathbb{Z}[x]/(\Phi_n(x))$ to our encryption context.

2.4 Lattices

This section introduces lattices and hard lattice problems that will base the security of fully homomorphic schemes.

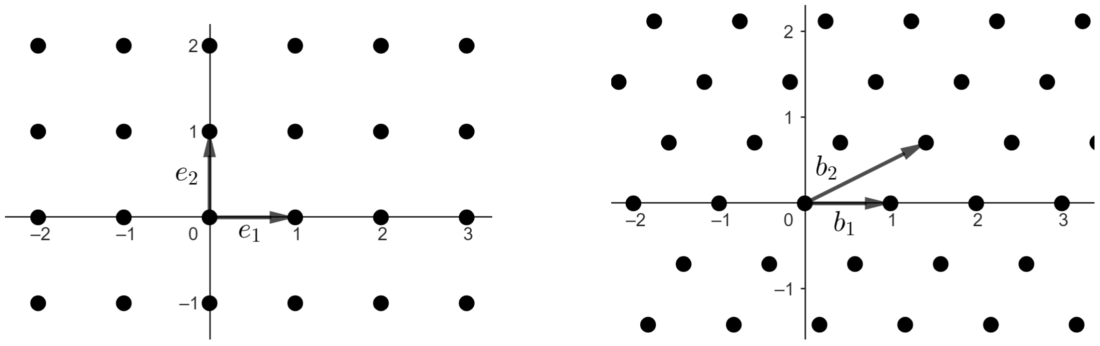
Definition 2.4.1 (Lattice)**.** Let $b_1, \ldots, b_k \in \mathbb{R}^n$ be linearly independent vectors, then

$$\mathcal{L} = \left\{\sum_{i=1}^{k} y_i b_i \mid y_i \in \mathbb{Z}\right\}$$

is a (n-dimensional) **lattice** and $(b_1, \ldots, b_k)$ its **basis**. If we create a $n \times k$ matrix $\mathbf{B}$, where the i-th column is b_i, then we can write

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}y|\ y \in \mathbb{Z}^k\} \subset \text{span}(B)$$

Figure 2 shows two geometrical examples in $\mathbb{R}^2$, the first one using canonical basis (e_1, e_2), and the second suing $B = (b_1, b_2)$, with $b_1 = e_1$ and $b_2 = \frac{1}{\sqrt{2}}(2, 1)$



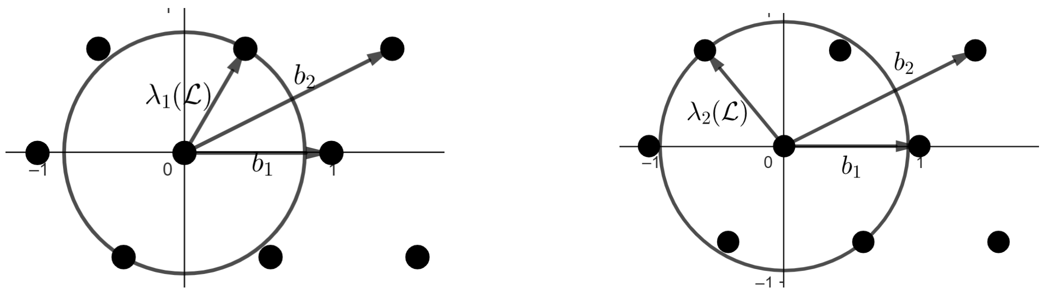
(a) Canonical basis (b) Basis $B = (b_1, b_2)$

Figure 2 – Lattices examples

Definition 2.4.2 (Ideal lattice)**.** An **ideal lattice** $\mathcal{L}$ is a lattice that is closed under addition and multiplication: For $v_1, v_2 \in \mathcal{L}$, we have $v_1 + v_2 \in \mathcal{L}$ and $v_1 \cdot v_2 \in \mathcal{L}$, with the operations inherited by $\mathbb{R}^n$

Definition 2.4.3 (Successive minima)**.** The i^{th} successive minima of an n-dimensional lattice $\mathcal{L}$, denoted by $\lambda_i(\mathcal{L})$ is the smallest r such that origin-centered ball with radius r contains i linearly independent (LI) lattice vectors.

In Figure 2a example $\lambda_1(\mathcal{L}) = \lambda_2(\mathcal{L}) = 1$, since the unity ball contains the two LI canonical vectors. But in Figure 2b, it's not so obvious, but $\lambda_1(\mathcal{L}) = ||b_3||_2 \approx 0.82$, where $b_3 = b_2 - b_1$ and $\lambda_2(\mathcal{L}) = ||b_4||_2 \approx 0.92$, with $b_4 = b_3 - b_1$. In Figure 3, we can visually verify that b_3 and b_4 are indeed LI.



(a) 1st successive minima (b) 2nd successive minima

Figure 3 – Successive Minimas

2.4.1 Lattice Problems

Now we can define two famous lattice problems, all of them considered NP-hard (LIU, 2018). "The approximation factor γ is usually taken as a function of lattice dimension $\gamma(n)$." (MICCIANCIO, 2007)

Definition 2.4.4 (Shortest Vector Problem - SVP)**.** Given a lattice $\mathcal{L}$ and a norm $||.||$, find a non-zero $v \in \mathcal{L}$ such that $||v|| \leq \gamma\lambda_1(\mathcal{L})$, i.e., the non-zero lattice vector closest to the origin.

In the above example of $\mathcal{L}(b_1, b_2)$ the shortest vector is b_3, for $\gamma = 1$.

Definition 2.4.5 (Shortest Independent Vector Problem - SIVP)**.** Given a lattice $\mathcal{L}$ of dimension n and a norm $||.||$, find n linearly independent vectors $v_1, v_2, \dots, v_n$ such that $\max\limits_i ||v_i|| \leq \gamma\lambda_n(\mathcal{L})$.

For $\mathcal{L}(b_1, b_2)$, the solution for the SIVP$_\gamma$ with $\gamma = 1$ is $\{b_3, b_4\}$.

These problems establish the so-called *lattice-based cryptography*, which are cryptosystems that have their security based on solving the above problems of variants. All The fully homomorphic encryption schemes we shall dive deeper into in the next chapter are lattice-based.

2.4.2 Ring Learning with Errors

Now that we have computationally complex challenges, the next step is, based on these results, constructing a concrete secure cryptography scheme. The following two problems are a step in this direction.

Definition 2.4.6 (Learning with Errors - LWE)**.** Given an integer n, a prime integer q such that[5] $2 \leq q \leq \text{poly}(n)$ and a distribution χ over $\mathbb{Z}_q$. Fix $\boldsymbol{s} \in \mathbb{Z}_q^n$, the Search LWE$_{n,q,\chi}$ problem is to recover $\boldsymbol{s}$, given observations $(\boldsymbol{a}_i, b_i)$, where[6]:

$$\begin{aligned}
&\boldsymbol{a}_i \xleftarrow{\mathcal{U}} \mathbb{Z}_q^n \\
&b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \ (\text{mod} q) \\
&e_i \xleftarrow{\chi} \mathbb{Z}_q
\end{aligned}$$

The Decision LWE$_{n,q,\chi}$ is the problem of distinguishing between the distribution of $(\boldsymbol{a}_i, b_i)$ gotten from the above procedure and the uniform distribution over $\mathbb{Z}_q^{n+1}$.

Under reasonable assumptions on χ, if an algorithm efficiently solves the above (decision) problem, GapSVP and SIVP can also be solved efficiently (REGEV, 2009).

LWE provides a nice setup for private key encryption since one can expose $(\boldsymbol{a}_i, b_i)$ while the hardness of lattice-based problems guarantees the security of $\boldsymbol{s}$. Further, we shall name $(\boldsymbol{a}_i, b_i)$ as the public key and $\boldsymbol{s}$ as the private key. One can still prove that Seach-LWE can be reduced to Decision-LWE (LYUBASHEVSKY; PEIKERT; REGEV, 2010), which completes the cycle.

In our main FHE construction, the scheme is based on a variant of LWE:

5 poly(n) denotes a polynomial function of n.

6 Notation $x \xleftarrow{\chi} S$, means x were drawn from distribution χ over the set S, and $\mathcal{U}$ is the uniform distribution

Definition 2.4.7. [Ring Learning with Errors - RLWE] Set integers n, q, with n being a power of two, and $2 \leq q \leq \text{poly}(n)$. Let $f(x) = x^n + 1$ (the $2n$-th cyclotomic polynomial), define $\mathcal{R} = \mathbb{Z}[X]/(f(X))$, a distribution χ over $\mathcal{R}$ and $\mathcal{R}_q = \mathcal{R}/(q) = \mathbb{Z}_q[X]/(f(X))$ which can be represented by polynomials with integer coefficients wrapped by q and with degree up to $n-1$.

Let $s = s(x) \in \mathcal{R}_q$ be an element chosen uniformly from the ring. The Search RLWE$_{n,q,\chi}$ problem is to recover s given a set of pairs $(a_i, b_i) \in \mathcal{R}_q^2$, with

$$\begin{aligned} a_i &\xleftarrow{\mathcal{U}} \mathcal{R}_q \\ b_i &= a_i \cdot s + e_i \pmod{q} \\ e_i &\xleftarrow{\chi} \mathcal{R} \end{aligned}$$

Analogously with LWE, we define Decision RLWE$_{n,q,\chi}$ as the problem of distinguishing the distribution of the above pairs from a uniform distribution over $\mathcal{R}_q^2$.

The Search RLWE can also be reduced to Decision RLWE. Moreover, under some assumptions over the parameters n, q, χ, which we won't state here, an efficient algorithm for Decision RLWE$_{n,q,\chi}$ implies an efficient algorithm for solving SVP (LYUBASHEVSKY; PEIKERT; REGEV, 2010).

The choice of $f(x)$ being a power of two cyclotomic polynomial is purely technical; in general, $f(x)$ could be any cyclotomic polynomial, and all the theory would still holds. We shall further want to multiply two instances of the cyclotomic ring, and there is a method based on Fast Fourier Transform to perform such task (JIA, 2022). But choosing specifically $f(x) = x^n + 1$ admits optimized and faster implementations (LYUBASHEVSKY; MICCIANCIO, et al., 2008).

3 Fully Homomorphic Encryption

This chapter gives an overview of the FHE research. Section 3.1 returns in 1978 and revises the original proposal of the existence of FHE schemes (RIVEST; ADLEMAN; DERTOUZOS, 1978), which was called *privacy homomorphisms*. Section 3.2 explores Gentry's solution through his Ph.D. thesis (GENTRY, 2009) proving such existence. Section 3.3 presents the so-called CKKS scheme (CHEON; KIM, et al., 2017), a practical and modern scheme that allows approximate encryption of complex (so, also real) numbers, well suitable for machine learning applications.

3.1 Privacy Homomorphisms

Assume we can represent the unencrypted data by an algebraic structure $\mathcal{P} = (S; f_1, \ldots, f_k)$, i.e., a set S ported with the operations $f_1, \ldots, f_k$. We will further call this structure the plaintext space.

An alternative algebraic structure, the ciphertext space $\mathcal{C} = (S', f'_1, \ldots, f'_k)$, is constructed to represent the encrypted data. To build a *privacy homomorphism* (RIVEST; ADLEMAN; DERTOUZOS, 1978), one needs a decryption function $\phi : S' \to S$ and its inverse $\phi^{-1} : S \to S'$ satisfying the homomorphic property from $\mathcal{C}$ do $\mathcal{P}$:

$$\begin{gathered} f'_i(a, b, \ldots) = c \Rightarrow \\ f_i(\phi(a), \phi(b), \ldots) = \phi(c), \text{ for } i = 1, \ldots, k \\ \text{with } a, b, \ldots \in S' \end{gathered} \tag{3.1}$$

This means that for all available operations f'_i, its evaluation on encrypted elements must result in a value that, after decryption, corresponds to the same computation on the unencrypted domain.

An example of privacy homomorphism is the RSA cryptosystem (RIVEST; SHAMIR; ADLEMAN, 1978), which uses $\mathcal{P} = (\mathbb{Z}_p; \times_p)$, the integers modulo p with p prime, and the multiplication modulo p. Setting $N = pq$, where q is a large prime and choosing e coprime with $(p-1)(q-1)$, the ciphertext space as $(\mathbb{Z}_N; \times_N)$ and is connected with $\mathcal{P}$ through the encryption function:

$$\begin{gathered} \phi^{-1} : \mathbb{Z}_p \to \mathbb{Z}_N \\ \phi^{-1}(x) = x^e (\operatorname{mod} N) \end{gathered}$$

Taking $x, y \in \mathbb{Z}_p$ and its encrypted versions $x' = \phi^{-1}(x),\ y' = \phi^{-1}(y)$, we have:

$$\begin{aligned} x' \times_N y' &= (x^e)(y^e)(\operatorname{mod} N) \\ &= (xy)^e (\operatorname{mod} N), \end{aligned}$$

which is an encryption of xy. Then the multiplication satisfies property 3.1, showing that such a system is indeed a privacy homomorphism.

3.1.1 Requirements and Limitations

The following properties for $\mathcal{C}, \phi$ and ϕ^{-1} are required by the authors:

a) for a given element $s \in S$, its encrypted version $\phi^{-1}(s)$ should not require much more storage space;

b) ϕ and ϕ^{-1} should be easy to compute;

c) the operations f_i' should be efficiently computable in $\mathcal{C}$;

d) ϕ should not be vulnerable to the chosen plaintext attack;

e) The operations of $\mathcal{C}$ should not be sufficient to yield an efficient computation of ϕ.

The last requirement forces a critical restriction on such morphisms: a comparison operator "$\leq$" can't be available in the ciphertext space, otherwise, no secure privacy homomorphism exists.

Take for example $\mathcal{P} = (\mathbb{N}; +, \leq)$ and $\mathcal{C} = (W; +', \leq')$ for some W. A malicious party who has $\phi^{-1}(n)$ and wants to discover what $n \in \mathbb{N}$ generated such ciphertext can apply the following binary search strategy:

a) compute $1' = \phi^{-1}(1)$;

b) compute $2' = 1' +' 1'$, then $4' = 2' +' 2'$;

c) continue until finding k, such that $\phi^{-1}(n) \leq' (2^k)' = \phi^{-1}(2^k)$

d) knowing that $n \in [2^{k-1}, 2^k]$, compute an encryption of the interval midpoint $\phi^{-1}(m) = \phi^{-1}(2^{k-1} - 2^{k-2})$;

e) homomorphically compare $\phi^{-1}(n) \leq' \phi^{-1}(2^{k-1}) +' \phi^{-1}(m)$;

f) repeat the last two steps properly redefining the interval until getting n exactly.

This is an efficient $O(\log n)$ algorithm to compute the decryption function ϕ, using the operations in $\mathcal{C}$ and the ability to generate encryptions of arbitrary constants (such as 1 and m in the above example).

The article finishes with the authors pondering if such an approach with all required security restrictions could be worthwhile in practice and what algebraic structures $\mathcal{P}$ would provide useful privacy homomorphisms.

3.2 Bootstrappable encryption

The existence of (useful) privacy homomorphisms remained an open problem for more than 30 years until (GENTRY, 2009) came up with a beautiful solution in his Ph.D. thesis. He introduced a privacy homomorphism based on ideal lattices and proposed a method named bootstrapping, his most significant insight, to control ciphertext noise growth. A second paper was proposed, with a slightly simpler scheme using integers instead of lattices (DIJK et al., 2010).

This section summarizes Gentry's achievement, 3.2.1 with a complete summarized overview of the solution, 3.2.1 describing the scheme over the integer. We let the explanation of the original cryptosystem over ideal lattices to Appendix A since not reading doesn't interfere in the text flow.

3.2.1 Overview and Bootstrapping

Let's denote $\phi^{-1}(.)$ by $\text{Enc}(\text{pk}, .)$, now requiring a public key pk, and $\phi(.)$ by $\text{Dec}(\text{sk}, .)$, where sk is a secret key. The construction also requires an evaluate algorithm $\text{Eval}(\text{pk}; .)$ that computes a function f in the ciphertext space. In the previous section's construction, this algorithm $\text{Eval}(\text{pk}, f_i, \ldots) = f_i'(\ldots)$, i.e., the computation of f in the encrypted domain won't be a direct evaluation, but a modified one, adapting to restrictions and patterns of the new space.

Take a function $f \in \mathcal{P}$ in the plaintext space, and some elements $s_1, s_2, \ldots \in S$, and its encrypted versions $c_i = \text{Enc}(\text{pk}, s_i)$. Rewriting 3.1 in this new notation, we get the correctness property for f:

$$\text{Dec}(\text{sk}, \text{Eval}(\text{pk}, f, c_1, c_2, \ldots)) = f(s_1, s_2, \ldots) \tag{3.2}$$

The encryption function **must add random noise to the plaintext** messages to resist the chosen ciphertext attack as required in the property (d) of 3.1.1. Calling $\text{Enc}(\text{pk}, s)$ multiple times would yield different ciphertexts each time, but decrypting all of them, returns to s.

Definition 3.2.1 (Fully Homomorphic Encryption - FHE)**.** The first definition of FHE scheme defines it as a scheme that allows anyone to evaluate "any desired function f" over the encrypted data $c_1, c_2, \ldots$ without decrypting it, just as privacy homomorphisms. (GENTRY, 2009)

In practice, what is proven is the existence of a scheme that can evaluate addition and multiplication functions, which are enough to compose any boolean circuit.

$$\mathrm{Dec}(\mathrm{sk}, \mathrm{Eval}(\mathrm{pk}, +, c_1, c_2)) = s_1 + s_2$$
$$\mathrm{Dec}(\mathrm{sk}, \mathrm{Eval}(\mathrm{pk}, \times, c_1, c_2)) = s_1 \times s_2$$

Because of the noise added in the encryption step, if we perform multiple additions and multiplications, the noise can grow at levels that might affect the above correctness property (this will soon become clearer). Then, we introduce the following concept:

Definition 3.2.2 (Somewhat Homomorphic Encryption - SHE)**.** Let F be a circuit of composition of additions and multiplications with a given depth. A SHE scheme satisfies the following:

$$\mathrm{Dec}(\mathrm{sk}, \mathrm{Eval}(\mathrm{pk}, F, c_1, c_2, \ldots)) = f(s_1, s_2, \ldots)$$

only if the depth of F is less than the scheme's maximum allowed depth.

The bottleneck to achieving an FHE scheme is the introduction of noise in the encryption function. A circuit F with a big enough number of multiplications, for example, might not satisfy the correctness property because of the noise's size compared to the original message.

Then a process called *bootstrap* (or recrypt) is introduced (GENTRY, 2009), which is a way of reducing the ciphertext noise by homomorphically evaluating the own scheme decryption function, producing a new ciphertext that encrypts the same message with less noise.

To achieve a fully homomorphic scheme out of a somewhat homomorphic one, it's enough to prove that the scheme is "bootstrappable", i.e., that it can correctly evaluate circuits that are deeper than its own decryption circuit.

Suppose c encrypts s under pk_1, and we want to reduce its noise; one way to do that is to decrypt c using the secret key, getting back to the plaintext space (without any noise) and then encrypt s again generating a fresh and new encryption. To maintain security and not reveal the actual message, the process is done homomorphically.

Suppose we have $\mathrm{sk}^* = \mathrm{Encrypt}(\mathrm{sk}_1, \mathrm{pk}_2)$, an encryption of the secret key under a second public key pk_2. Define the Recrypt algorithm as:

$$\begin{aligned}
&\mathrm{Recrypt}(\mathrm{pk}_2, \mathrm{sk}^*, \mathrm{Dec}, c)\\
&\qquad\qquad \text{Set } c^* = \mathrm{Enc}(c, \mathrm{pk}_2)\\
&\qquad\qquad \text{Output } c_2 = \mathrm{Eval}(\mathrm{pk}_2, \mathrm{Dec}, \mathrm{sk}^*, c^*)
\end{aligned}$$

Figure 4 illustrates this procedure through the blue arrows.

$$\begin{tikzcd}
c^* \arrow[r, "\mathrm{Eval}(\mathrm{pk}_2, \mathrm{Dec}, \mathrm{sk}^*, \cdot)"] & c_2 \arrow[d, "\mathrm{Dec}(\mathrm{sk}, \cdot)"] \\
c \arrow[u, "\mathrm{Enc}(\mathrm{pk}_2, \cdot)"] \arrow[r, "\mathrm{Dec}(\mathrm{sk}, \cdot)"'] & m
\end{tikzcd}$$

Figure 4 – Bootstrapping procedure

The Enc function might introduce some noise to c_2, but as long as this noise is less than the one associated with c, we succeed in refreshing ciphertext noise.

The SHE schemes proposed by Gentry were not originally bootstrappable because the decryption was too deep. The work then dedicates a lot of effort to minimize the depth of the decryption circuit, but the ability to bootstrap is only achieved by a technique called *squashing*, where the encrypter starts the decryption process, giving partial hints to the decryptor. Although he could arrive at an FHE scheme, squashing introduced additional security requirements.

3.2.2 An integer scheme

A second work from Gentry (DIJK et al., 2010) introduces a conceptually simpler SHE scheme using the ring of integers as the underlying algebraic structure instead of ideal lattices.

We start with a private key scheme and then transform it into a public key one:

- KeyGen - sets the private key as an odd integer p
- $\mathrm{Enc}(p, m)$ - given a bit message $m \in \{0, 1\}$, outputs $c = pq + 2r + m$, where q and r are choosen randomly in some prescribed interval.
- $\mathrm{Dec}(p, c)$ - outputs $(c \bmod p) \bmod 2$

Correctness property can be easily checked, assuming $2r < p/2$:

$$\begin{aligned}
&\text{Dec}(p, \text{Enc}(p, m)) \\
&= \text{Dec}(p, pq + 2r + m) \\
&= ((pq + 2r + m) \bmod p) \bmod 2 \\
&= (2r + m) \bmod 2 \\
&= m
\end{aligned}$$

Let's emphasize the need for bootstrapping with a toy example.

Let's say we want to add $m = 1$ continuously. Set $p = 29$, and let's suppose the encryption algorithm sampled $q = 8, r = 3$. Now, take the ciphertext $pq + 2r + m$ and add-t to itself $c + c = p(2q) + 4r + m + m$.

In the decryption function, the modulo-p reduction will eliminate $p(2q)$, and mod 2 will output us $m + m = 0$ as desired.

In general, if $\alpha \geq 1$ is an integer., $\alpha c = p(\alpha q) + 2\alpha r + \alpha m$,

If we want to discover how many times we can add c to itself and the decryption still works, the only condition is $2\alpha r < p$. Solving the inequality for $p = 29, r = 3$ we get $\alpha < 29/6 \approx 4.83$. So for $\alpha = 5$ we have no guarantee that the scheme will work.

Since $\alpha = 5$ is odd $\alpha m = 1$, so we would like $\text{Dec}(29, 5c) = 1$, but:

$$\begin{aligned}
&\text{Dec}(29, 5c) \\
&= \text{Dec}(29, 29 \cdot (5q) + 2 \cdot 5r + 5m) \\
&= ((29 \cdot (5q) + 30 + 5m) \bmod 29) \bmod 2 \\
&= (1 + 5m) \bmod 2 \\
&= 6 \bmod 2 \\
&= 0
\end{aligned}$$

Then decryption failed because the error was sum up to $5\alpha r$ and eventually got bigger than p.

To obtain a public key encryption scheme out of the above private key one, we set:

$\text{pk} = (x_1, \ldots, x_\tau)$, where $x_i = pq_i + 2r_i$, p is a given private key, and q_i, r_i are random sampled for each generation of x_i.

Then the encryption algorithm is transformed:

- $\text{Enc}(\text{pk}, m)$ - random sample S from $\{1, \ldots, \tau\}$ and an integer r, then output $c = m + 2r + 2\sum_{i \in S} x_i$

The security of the scheme is based on the hardness of the *approximate gcd problem*, i.e, find p given its “approximate multiples” $x_1, \dots, x_\tau$.

3.2.3 Practical considerations and further research

An implementation of Gentry’s scheme (using ideal lattices) was proposed in (GENTRY; HALEVI, 2011). They used a powerful cloud machine to benchmark the system: 64-bit quad-core Intel Xeon E5450 processor, 3GHz, with 24GB of RAM.

Using $n = 2^{15}$ as the lattice dimension (see Appendix A), a parameter that provides a high level of security, the scheme took 2.2 hours for KeyGen and 31 minutes for Recrypt. The real bottleneck is the Recrypt algorithm since Keygen is executed only one time, but we need to bootstrap many times for evaluating deep functions, such as iterative algorithms for machine learning optimization. Although Gentry’s work was a big mathematical breakthrough, making a practical FHE scheme remained an open problem.

The research community has focused on achieving computationally practical schemes ever since. In the decade after 2009, a lot of solutions came up with real improvements over the original construction. A standardization was proposed (ALBRECHT et al., 2018) to discuss security, implementation API, and applications, listing some important schemes such as BFV (FAN; VERCAUTEREN, 2012), BGV (BRAKERSKI; GENTRY; VAIKUNTANATHAN, 2011), and GSW (GENTRY; SAHAI; WATERS, 2013). We will present the CKKS/HEAAN[1] scheme (CHEON; KIM, et al., 2017), which is considered the most appropriate for machine learning applications since it has a native solution for supporting complex numbers (so, also real), while previous schemes used boolean or integer based encoding. Although it’s not listed on 2018 standards, the scheme is a candidate for future versions of the document.

3.3 FHE over the complex numbers

The message m of the HEAAN scheme will live in the plaintext of the cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ with N being a power of two; the ciphertext will be tuples in $\mathcal{R}_q \times \mathcal{R}_q$. This can be quite contradicting to the proposal of the scheme for supporting complex numbers, but this can be reached using a clever encoding/decoding procedure that can map a complex vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$ to a polynomial $m \in \mathcal{R}_q$, and vice-versa. This allows us to perform operations in a SIMD[2], saving computational costs. Figure 5 outlines the process of how CKKS computes a function over a complex vector.

[1] HEAAN is the initials of the paper title "Homomorphic Encryption for Arithmetic of Approximate Numbers".
CKKS stands for the author’s surname initials: Jung H. Cheon, Andrey Kim, Miran Kim, and Yongsoo Song

[2] Single instruction, multiple data

$$\begin{tikzcd}
\begin{matrix} \boldsymbol{z} \\ \in \mathbb{C}^{N/2} \end{matrix} \arrow[r, "\text{Encode}"] & \begin{matrix} m(X) \\ \in \mathcal{R}_q \end{matrix} \arrow[r, "\text{Encrypt}"] & \begin{matrix} c = (c_1(X), c_2(X)) \\ \in \mathcal{R}_q^2 \end{matrix} \arrow[d, "\text{Compute } f"] \\
\begin{matrix} \boldsymbol{z}' = f(\boldsymbol{z}) \\ \in \mathbb{C}^{N/2} \end{matrix} & \begin{matrix} m'(X) = f(m) \\ \in \mathcal{R}_q \end{matrix} \arrow[l, "\text{Decode}"'] & \begin{matrix} c' = \text{Eval}(f, c) \\ \in \mathcal{R}_q^2 \end{matrix} \arrow[l, "\text{Decrypt}"']
\end{tikzcd}$$

Figure 5 – CKKS scheme overview. Adapted (HUYNH, 2021)

HEAAN is an approximate scheme that doesn't satisfy the identity correctness property, $\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, m)) = m$. Instead, one will get $\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, m)) = m + e$, and we can guarantee that e is small enough compared to m so that we recover an approximate decryption of the original message.

The scheme requires a process called *relinearization* after each multiplication, introduced by BFV scheme (FAN; VERCAUTEREN, 2012) and also used in CKKS. Directly multiplying two ciphertexts $\boldsymbol{c} = (c_1, c_2)$ and $\boldsymbol{c}' = (c_1', c_2')$ yields a vector in $(d_0, d_1, d_2) \in \mathcal{R}_q^3$ where:

$$\begin{aligned} d_0 &= c_1 \cdot c_1' \\ d_1 &= c_1 \cdot c_2' + c_2 \cdot c_1' \\ d_2 &= c_2 \cdot c_2' \end{aligned}$$

The goal of relinearization is to transform the above triple into an equivalent tuple, i.e., a tuple that approximately preserves the decryption correctness. This process is also used in BFV and BGV schemes; although is quite expansive and introduces additional noise, it avoids ciphertext size growth, keeping it as a two-dimensional vector of polynomials.

The original HEAAN scheme was not fully homomorphic but a somewhat homomorphic one. Following Gentry's blueprint (GENTRY, 2009), to transform it into an FHE scheme, we should be able to evaluate the decryption circuit homomorphically, which is still an open problem for CKKS. Instead, the authors later proposed an *approximate bootstrapping* procedure (CHEON; HAN, et al., 2018), which follows Gentry's idea, but with an approximate decryption function. The consequence is that we lose precision after each bootstrap evaluation, but for statistical purposes, that's not a problem as long as we can control this approximation error.

3.3.1 Encoding and Decoding

The plaintext space, as we described before, is the cyclotomic ring $\mathbb{Z}[X]/(\Phi_M(X))$, with $M = 2N$ and N power of two. Our goal is to map complex $N/2$-dimensional[3] vectors $\boldsymbol{z} \in \mathbb{C}^{N/2}$ into elements of the ring (encoding) and vice-versa (decoding).

We begin with the *canonical embedding map*:

$$\begin{gathered}\sigma : \mathbb{C}[X]/(X^N + 1) \longrightarrow \mathbb{C}^N \\ m(X) \longrightarrow (m(\zeta_M), m(\zeta_M^3), \ldots, m(\zeta_M^{2N-1})), \\ \text{with } \zeta_M = \exp(2\pi i/M)\end{gathered}$$

The above map takes a polynomial with complex coefficients and evaluates it in the M^{th} primitive roots of unity. Since M is a power of two, from Definition 2.3.2, the primitive roots will be the ζ_M^k for $1 \leq k \leq M$ with $\gcd(k, M) = 1$, hence, k ranges through the first N odd numbers. The output is indeed a N- dimensional complex vector since both the polynomial coefficients and the primitive roots are complex.

An interesting fact is that σ is an isomorphism, and we can use its inverse σ^{-1} to encode a vector in $\boldsymbol{z} \in \mathbb{C}^N$ to a polynomial in $m(X) \in \mathbb{C}[X]/(\Phi_M(X))$. Recall that $m(X)$ can be written as $m(X) = \sum_{j=0}^{N-1} \alpha_j X^j$; So to construct σ^{-1} we must be able to take $\boldsymbol{z} = (z_1, \ldots, z_N))^T$ and return the coefficients $(\alpha_j)_{j=0,1,\ldots,N-1}$ such that $(m(\omega_1), m(\omega_3), \ldots, m(\omega_{2N-1}))^T = \boldsymbol{z}$, where $\omega_k = \zeta_M^k$ to simplify notation. This can be viewed as a linear system:

$$\begin{gathered} m(\omega_{2k-1}) = \sum_{j=0}^{N-1} \alpha_j \omega_{2k-1}^j = z_k \\ \text{for } k = 1, 2, \ldots, N; \end{gathered}$$

or as its matrix form:

$$A\boldsymbol{\alpha} = \boldsymbol{z},$$

$$\text{with } A = \begin{bmatrix} 1 & \omega_1 & \omega_1^2 & \ldots & \omega_1^{N-1} \\ 1 & \omega_3 & \omega_3^2 & \ldots & \omega_3^{N-1} \\ 1 & \omega_5 & \omega_5^2 & \ldots & \omega_5^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_{2N-1} & \omega_{2N-1}^2 & \ldots & \omega_{2N-1}^{N-1} \end{bmatrix} \text{ and } \boldsymbol{\alpha} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N-1} \end{bmatrix}$$

The matrix A is the Vandermonde matrix for the primitive roots $\omega_k = \zeta_M^k$, and has a known inverse (TURNER, 1966). Then, we can find the coefficients through $\boldsymbol{\alpha} = A^{-1}\boldsymbol{z}$,

[3] The reason for this dimension will soon become clearer.

arriving to the inverse canonical embedding map:

$$\sigma^{-1}:\ \mathbb{C}^N \longrightarrow \mathbb{C}[X]/(X^N+1)$$
$$\sigma^{-1}(\boldsymbol{z}) = m(X),$$

with $m(X) = \sum_{j=0}^{N-1} \alpha_j X^j$ and $\alpha_j = (A^{-1})_j \boldsymbol{z}$

The complex primitive roots are symmetrical conjugates. In Figure 1 for example $\omega_1 = \overline{\omega_1} = \omega_7$, in general for the M^{th} primitive roots, $\omega_j = \overline{\omega_{-j}}$, where $-j$ index is taken modulo M. Remember that, for decoding, the goal was to map $\mathcal{R} = \mathbb{Z}[X]/(X^N+1)$ to $\mathbb{C}^{N/2}$. Until now, we have a map $\mathcal{R} \to \mathbb{C}^N$. The image $\sigma(\mathcal{R})$ can be reduced to half its dimension, noticing that $m(\omega_j) = \overline{m(\omega_{-j})}$:

$$\begin{aligned}\overline{m(\omega_{-j})} &= \overline{\sum_{k=0}^{N-1} \alpha_k \omega_{-j}^k} = \sum_{k=0}^{N-1} \overline{\alpha_k \omega_{-j}^k} \\ &= \sum_{k=0}^{N-1} \alpha_k \overline{\omega_{-j}^k} \\ &= \sum_{k=0}^{N-1} \alpha_k \omega_j^k = m(\omega_j)\end{aligned} \tag{3.3}$$

Equality 3.3 holds because $\alpha_k \in \mathbb{Z} \subset \mathbb{R}$ so it does not affect the conjugate operator. Hence we have proved that the image $\sigma(\mathcal{R})$ are vectors in $\mathbb{C}^N$ whose coordinate j are complex conjugates of coordinate $-j$. We can define now the subring $\mathbb{H} \subseteq \mathbb{C}^N$ as:

$$\mathbb{H} = \{(z_j)_{j\in\mathbb{Z}_M^*}; z_j = \overline{z_{-j}},\ \forall j \in \mathbb{Z}_M^*\},$$

where $\mathbb{Z}_M^* = \{k \in \mathbb{Z}; \gcd(k, M) = 1\} = \{1, 3, 5, \dots, 2N-1\}$.

The image $\sigma(\mathcal{R})$ is actually in $\mathbb{H}$. So, we can define the projection $\pi : \mathbb{H} \to \mathbb{C}^{N/2}$ that takes a N-dimensional vector in $\mathbb{H}$ and outputs a $N/2$-dimensional one, removing conjugates elements. Naturally, we also define $\pi^{-1} : \mathbb{C}^{N/2} \to \mathbb{H}$ that do the opposite, expanding a vector including the conjugates of its elements. Then, the decoding procedure will be $\pi \circ \sigma : \mathcal{R} \to \mathbb{C}^{N/2}$.

Encoding is not that trivial. Notice that the codomain of σ^{-1} is $\mathbb{C}[X]/(X^N+1)$, so we cannot guarantee that the polynomials will have integer coefficients. The first step is expanding $\boldsymbol{z} \in \mathbb{C}^{N/2}$ to $\mathbb{H}$ through $\pi^{-1}(\boldsymbol{z})$, but to guarantee that applying σ^{-1} will return an integer polynomial, we need to project $\pi^{-1}(\boldsymbol{z})$ to the image $\sigma(\mathcal{R})$ using some rounding process $\lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$. This will approximate π^{-1} to a close element in $\sigma(\mathcal{R})$ so that applying the inverse map results in a ring element. For more details about rounding techniques, refer to (CHEON; KIM, et al., 2017) and (LYUBASHEVSKY; PEIKERT; REGEV, 2013).

For precision preservation purposes, the encoding procedure multiplies $\pi^{-1}(\boldsymbol{z})$ by a predefined scaling factor $\Delta > 1$. Then we define Ecd (encode) and Dcd (decode) functions as:

$$\begin{aligned}\mathrm{Ecd} : \mathbb{C}^{N/2} &\longrightarrow \mathcal{R} \\ \mathrm{Ecd}(\boldsymbol{z}; \Delta) &= \sigma^{-1}(\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}) \\ \mathrm{Dcd} : \mathcal{R} &\longrightarrow \mathbb{C}^{N/2} \\ \mathrm{Dcd}(m; \Delta) &= \pi \circ \sigma(\Delta^{-1} \cdot m)\end{aligned}$$

3.3.2 Encryption, Decryption, and Relinearization

Now let's describe the concrete cryptographic primitives of the CKKS scheme. Fix an integer base p and define $q_\ell = p^\ell q_0$ for some given $q_0 \in \mathbb{Z}$. This is a *leveled homomorphic encryption* scheme, with levels $0 \leq \ell \leq L$; after each multiplication, we rescale the message to a lower level to control the size of noise.

The notation $\mathbb{Z}_q$ will choose the representatives from $\mathbb{Z} \cap (q/2, q/2]$, and $[\cdot]_q$ applied to a polynomial means wrapping its coefficients modulo q. λ is a security parameter, which known attacks taking $\Omega(2^\lambda)$.

The Discrete Gaussian distribution $\mathcal{DG}(\sigma^2)$ over the polynomial ring $\mathcal{R}$ samples the coefficients from a discrete gaussian with variance σ^2 over $\mathbb{Z}^N$ (CHEON; KIM, et al., 2017); We'll use this distribution for sampling RLWE errors. Another useful distribution is $\mathcal{ZO}(\rho)$ over $\mathcal{R}_3 = \mathbb{Z}_3[X]/(X^N + 1)$, i.e., the polynomials with degree up to $N-1$ with coefficients in $\{0, \pm 1\}$. $\mathcal{ZO}(\rho)$ draws N samples with probability $1 - \rho$ of being zero and $\rho/2$ of being $+1$ or -1, this becomes the coefficients of the polynomial in $\mathcal{R}_3$.

- KeyGen(λ)- Given a security parameter λ, chooses cyclotomical dimension $M = M(\lambda)$, error standard deviation $\sigma = \sigma(\lambda)$ and an integer $P = P(\lambda)$.

  Samples $s \leftarrow \mathcal{R}_3$, $a \xleftarrow{\mathcal{U}} \mathcal{R}_{q_L}$ and $e \xleftarrow{\mathcal{DG}(\sigma^2)} \mathcal{R}$.

  Set sk $= (1, s)$ and pk $= (b, a)$, with $b = [-a \cdot s + e]_{q_L}$. Security of public key exposure is guaranteed by RLWE (Def. 2.4.7).

  Samples $a' \xleftarrow{\mathcal{U}} \mathcal{R}_{P \cdot q_L}$, $e' \xleftarrow{\mathcal{DG}(\sigma^2)} \mathcal{R}$ and set the evaluation key evk $= (b', a')$, with $b' = [-a' \cdot s + e' + P \cdot s^2]_{P \cdot q_L}$. Its use will become clear soon, and security is also guaranteed by RLWE.

- $\mathrm{Enc}(\mathrm{pk}, m) = [v \cdot \mathrm{pk} + (m + e_0, e_1)]_{q_L}$, where $v \xleftarrow{\mathcal{ZO}(0.5)} \mathcal{R}_2$, and $e_0, e_1 \xleftarrow{\mathcal{DG}(\sigma^2)} \mathcal{R}$. Notice that the ciphertext is a two-dimensional vector of polynomials in $\mathcal{R}_{q_L}$.

- $\mathrm{Dec}(\mathrm{sk}, \boldsymbol{c}) = [\langle \boldsymbol{c}, \mathrm{sk} \rangle]_{q_\ell} = [c_1 + c_2 \cdot s]_{q_\ell}$, for $\boldsymbol{c} = (c_1, c_2)$ at level ℓ.

- $\mathrm{Eval}(+, \boldsymbol{c}, \boldsymbol{c}') = [\boldsymbol{c} + \boldsymbol{c}']_{q_\ell}$, for $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{R}^2_{q_\ell}$

- $\mathrm{Eval}(\mathrm{evk}, \times, \boldsymbol{c}, \boldsymbol{c}') = (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot \mathrm{evk} \rceil (\mathrm{mod}\ q_\ell)$, where for $\boldsymbol{c} = (c_1, c_2)$ and $\boldsymbol{c}' = (c_1', c_2')$, we have, as described before $(d_0, d_1, d_2) = (c_1 \cdot c_1', c_1 \cdot c_2' + c_2 \cdot c_1', c_2 \cdot c_2')$

The multiplication of ciphertexts is using a process called *relinearizaton*, to reduce ciphertext dimension from $\mathcal{R}^3_{q_\ell}$ to $\mathcal{R}^2_{q_\ell}$. The goal is to find (d_0^*, d_1^*) such that the decryption is approximately equivalent:

$$[d_0^* + d_1^* \cdot s]_{q_\ell} \approx [d_0 + d_1 \cdot s + d_2 \cdot s^2]_{q_\ell}$$

In this case, $(d_0^*, d_1^*) = (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot \mathrm{evk} \rceil (\mathrm{mod}\ q_\ell)$ provides a ciphertext is an approximate valid encryption of the original multiplication of ciphertext (CHEON; KIM, et al., 2017).

The levels L define the maximum multiplicative depth the scheme allows because, after each multiplication, we rescale the ciphertext from $\ell \to \ell - 1$ using:

$$\mathrm{Rescale}_{\ell \to \ell'}(\boldsymbol{c}) = \left\lfloor \frac{q'_\ell}{q_\ell} \boldsymbol{c} \right\rceil$$

,

The intuition behind it is that each ciphertext has an embedded noise added by the encryption process, and multiplication increases it (exponentially!). the rescaling process controls the noise, reducing the ciphertext modulus and its built-in error. For more details, refer to (HUYNH, 2020).

Another useful operation permitted by the scheme is *rotation* of plaintext slots, i.e., for a given $\boldsymbol{z} = (z_1, z_2, \ldots, z_{N/2})$ we can homomorphically apply a left rotation by i for example, and output a ciphertext that encrypts $(z_i, z_{i+1}, \ldots, z_{N/2}, z_1, z_2 \ldots, z_{i-1})$. Analogously a right rotation method can be constructed.

3.3.3 Approximate Bootstrapping

To make an FHE scheme out of the previously described leveled homomorphic one, we need bootstrapping, i.e., be able to evaluate the decryption function $\mathrm{Dec}(\mathrm{sk}, \boldsymbol{c}) = [\langle \boldsymbol{c}, \mathrm{sk} \rangle]_{q_\ell}$ using the homomorphic scheme. While a direct approach to homomorphically compute it is still an open problem, an *approximate bootstrapping* was proposed by (CHEON; HAN, et al., 2018). The idea comes from the fact that modular reduction operation can be represented by a trigonometric function:

$$[\langle \boldsymbol{c}, \mathrm{sk} \rangle]_q = \frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q} \cdot \langle \boldsymbol{c}, \mathrm{sk} \rangle\right) + O(\varepsilon^3 \cdot q),$$

when $|[\langle \boldsymbol{c}, \mathrm{sk} \rangle]_q| \leq \varepsilon \cdot q$. CKKS scheme doesn't evaluate sine functions, it only permits additions and multiplications, so the authors used the Taylor expansion polynomial approximation, which can be represented as a composition of additions and multiplications.

4 Private Logistic Regression

In this chapter, we describe and test an algorithm for training a logistic regression model over encrypted data proposed in (HAN et al., 2018a). Section 4.1 reviews basics statistic about classical logistic regression in plain data. Section 4.2 describes how to adapt the training algorithm to the FHE setup. Finally, Section 4.3 show our experimental results.

4.1 Statistical Review

Logistic Regression is a probability model for binary class target variable. We have a dataset of n observations of d variables represented as a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and a target n-dimensional vector $\boldsymbol{y} \in \{0,1\}^n$.

For each observation $\boldsymbol{x}_i$ we have a corresponging binary responsing $y_i \in \{0,1\}$ that could mean a positive diagnosis for a disease in medical data, a credit default flag for in banking data or whether an email is spam or not. Logistic regression models the probability of a true flag $p(\boldsymbol{x}) = \mathbb{P}(Y = 1 | X = \boldsymbol{x})$ as if the log-odds were a linear function of $\boldsymbol{x}$:

$$\ln\left(\frac{p(\boldsymbol{x})}{1 - p(\boldsymbol{x})}\right) = \boldsymbol{w}^T \boldsymbol{x}',$$

where $\boldsymbol{w} = (w_0, w_1, \ldots, w_d)^T \in \mathbb{R}^{d+1}$ is the vector of weights and $\boldsymbol{x}' = (1; \boldsymbol{x})$ is an extended version of $\boldsymbol{x}$ with 1 as the first element, to multiply by the bias term w_0. Isolating $p(\boldsymbol{x})$, we get the model sigmoid model, for the probabilities:

$$p(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x}'),$$

where $\sigma(x) = 1/(1 + e^{-x})$.

The goal of the training algorithm is to find $\boldsymbol{w}$ that maximizes the bernoulli likelihood:

$$\mathcal{L}(\boldsymbol{w}) = \prod_{i=1}^{n} \sigma(\boldsymbol{w}^T \boldsymbol{x}_i')^{y_i} (1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i'))^{1-y_i}$$

Observe that, the main part is $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i')$ if $y_i = 1$ and $1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i')$ if $y_i = 0$. Transforming $y_i \in \{0,1\}$ to $y_i' = 2y_i - 1 \in \{-1, +1\}$ and using the sigmoid property $\sigma(-x) = 1 - \sigma(x)$, we can rewrite the likelihood as:

$$\mathcal{L}(\boldsymbol{w}) = \prod_{i=1}^{n} \sigma(\boldsymbol{w}^T \boldsymbol{z}_i),$$

where $\boldsymbol{z}_i = \boldsymbol{x}'_i \cdot y'_i$. This trick will simplify the gradient calculation in the maximization process. In practice we minimize the negative log-likelihood:

$$\begin{aligned}\ell(\boldsymbol{w}) &= -\frac{1}{n}\ln(\mathcal{L}(\boldsymbol{w})) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\ln(\sigma(\boldsymbol{w}^T\boldsymbol{z}_i))\end{aligned}$$

Since ln is a convex function in we can minimize it using the gradient descent iterative algorithm:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \alpha\nabla\ell(\boldsymbol{w}_i),$$

for a given learning rate $\alpha > 0$

Using $\sigma'(x) = \sigma(x)\sigma(-x)$, we can calculate the gradient:

$$\begin{aligned}\nabla\ell(\boldsymbol{w}) &= -\frac{1}{n}\sum_{i=1}^{n}\nabla\ln(\sigma(\boldsymbol{w}^T\boldsymbol{z}_i) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\nabla\sigma(\boldsymbol{w}^T\boldsymbol{z}_i)}{\sigma(\boldsymbol{w}^T\boldsymbol{z}_i)} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\sigma'(\boldsymbol{w}^T\boldsymbol{z}_i)\boldsymbol{z}_i}{\sigma(\boldsymbol{w}^T\boldsymbol{z}_i)} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sigma(-\boldsymbol{w}^T\boldsymbol{z}_i)\boldsymbol{z}_i\end{aligned}$$

Summarizing, a classical logistic regression training algorithm initializes $\boldsymbol{w}_0$ and then it update the weights by:

$$\begin{aligned}\boldsymbol{w}_{i+1} &= \boldsymbol{w}_i - \alpha\nabla\ell(\boldsymbol{w}_i), \\ \text{where } \nabla\ell(\boldsymbol{w}_i) &= -\frac{1}{n}\sum_{j=1}^{n}\sigma(-\boldsymbol{w}_i^T\boldsymbol{z}_j)\boldsymbol{z}_j \qquad (4.1)\end{aligned}$$

4.2 Homomorphic Training

To adapt the training algorithm to an FHE-friendly setup (HAN et al., 2018a) used three main ingredients:

a) Mini-batch gradient descent: The gradient is calculated over a subset of the data instead of single elements or the whole dataset. Effectively, the sum in equation 4.1 doesn't range from 1 to n, but in a predetermined subset of indexes. Such a procedure can take advantage of the message encoding process described in the previous chapter, while limiting circuit depth.

b) Nesterov Accelerated Gradient (NAG) Optimizer: Instead of conventional gradient descent, we'll use the NAG variant:

$$\begin{aligned}\boldsymbol{w}_{i+1} &= \boldsymbol{v}_i - \gamma\nabla\ell(\boldsymbol{v}_i) \\ \boldsymbol{v}_{i+1} &= (1-\eta)\boldsymbol{w}_{i+1} + \eta\boldsymbol{w}_i,\end{aligned}$$

where γ and η are parameters that must be set by the user.

c) Polynomial approximation for sigmoid: It's impossible to homomorphically evaluate e^{-x} exactly, so the authors propose using a polynomial approximation for the sigmoid function, fitted using least squares minimization. The degree 3 estimate that will be used in our tests is $\sigma_2(x) = 0.5 + 0.15x + -0.0015x^3$, and it's the polynomial has the least square distance to the actual sigmoid in the interval $[-8, 8]$ (Figure 6). It's desirable to have a low-degree polynomial to save multiplications, and a cubic function provides a decent estimate.

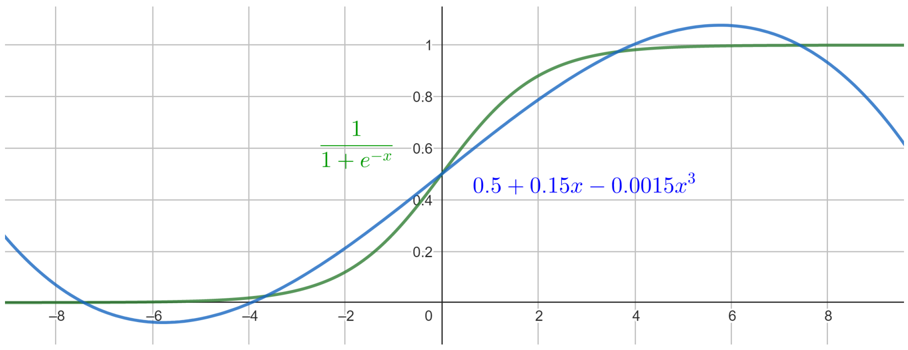


Figure 6 – Sigmoid polynomial approximation

Then the first version of the algorithm can be formalized:

Algorithm 1 Approximate mini-batch training through NAG (HAN et al., 2018a)

```
Input: Mini-batches $\{Z_i\}$ where $Z_i \in \mathbb{R}^{m\times(d+1)}$, parameter $\gamma, \eta$, number of iterations $K$
    and an approximate sigmoid $\sigma_2$
Output: Weight vectors $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^{d+1}$
 1: Initializes $\boldsymbol{w}, \boldsymbol{v} \leftarrow \mathbf{0}$;
 2: for $k$ in $1, \ldots, K$ do
 3:     Select a mini-batch $Z_i$ (in order or at random)
 4:     $\boldsymbol{a} = Z_i \cdot \boldsymbol{v}$
 5:     for $j$ in $1, \ldots, m$ do
 6:         $b_j = \sigma_2(a_j)$
 7:     end for
 8:     $\boldsymbol{\Delta} = \sum_{j=0}^{m-1} b_j \cdot Z_i[j]$
 9:     $\boldsymbol{w}^+ = \boldsymbol{v} - \gamma\boldsymbol{\Delta}$
10:     $\boldsymbol{v}^+ = (1-\eta)\boldsymbol{w}^+ + \eta\boldsymbol{w}$
11:     $\boldsymbol{w} = \boldsymbol{w}^+, \ \boldsymbol{v} = \boldsymbol{v}^+$
12: end for
```

Notice that the gradient Steps 4-7 calculate the sigmoidal part of the gradient, and step 8, effectively compute it. Steps 9-10 correspond to the NAG updates.

4.2.1 Ciphertext packing and data representation

Let $\boldsymbol{Z}$ be the $n \times (d+1)$ matrix where the columns are $\boldsymbol{z}_j = \boldsymbol{x}'_j \cdot y'_j$, and $x_{i,j}$ the i^{th} elements of $\boldsymbol{x}_j$. Denote $\boldsymbol{Z} = [\boldsymbol{y}',\ \boldsymbol{x}'_1,\ \boldsymbol{x}'_2,\ \dots,\ x'_d]$ or, more explicitly:

$$\boldsymbol{Z} = \begin{bmatrix} z[0][0] & z[0][1] & \dots & z[0][d] \\ z[1][0] & z[1][1] & \dots & z[1][d] \\ \vdots & \vdots & \ddots & \vdots \\ z[n-1][0] & z[n-1][1] & \dots & z[n-1][d] \end{bmatrix}$$

where $z[i][0] = y'_i$ and $z[i][j+1] = \boldsymbol{y}'_i \cdot x_{i,j}$, for $0 \le i \le n-1$ and $0 \le j \le d-1$

Instead of encrypting each element of $\boldsymbol{Z}$ separately, we partition the data in $m \times g$ submatrices and encrypt each block as a unique ciphertext:

$$Z_{i,j} = \begin{bmatrix} z[mi][gj] & \dots & z[mi][gj+(g-1)] \\ z[mi+1][gj] & \dots & z[mi+1][gj+(g-1)] \\ \vdots & \vdots & \vdots \\ z[mi+(m-1)][gj] & \dots & z[mi+(m-1)][gj+(g-1)] \end{bmatrix}, \tag{4.2}$$

for $0 \le i < n/m$ and $0 \le j < (d+1)/g$

Then we linearize (by row) the above matrix and transform it into a mg-dimensional real (so complex) vector $\boldsymbol{p}_{i,j}$. We then encrypt $\boldsymbol{p}_{i,j}$ using the scheme described in the previous chapter. The size of the blocks m and g are chosen such that $mg = N/2$, where $N/2$ is the dimension of encoding space $\mathbb{C}^{N/2}$.

Notice that this packing technique produces $n(d+1)/mg$ ciphertexts instead of $n(d+1)$ of a naive elementwise encoding approach.

The NAG weight vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ are also partitioned into $(d+1)/g$ sub-vectors and represented by a $m \times g$ matrix of m copies of the sub-vector. So the j^{th} sub-vector will corresponds to :

$$W_j = \begin{bmatrix} w[gj] & w[gj+1] & \dots & w[gj+(g-1)] \\ w[gj] & w[gj+1] & \dots & w[gj+(g-1)] \\ \vdots & \vdots & \vdots & \vdots \\ w[gj] & w[gj+1] & \dots & w[gj+(g-1)] \end{bmatrix},$$

and V_j is defined analogously. The encryption works as in $Z_{i,j}$

4.2.2 Batch Inner Product

Recall from Algorithm 1 that the training phase requires evaluating two inner products (steps 4 and 8). The two steps will take $O(g^2)$ multiplications and $O(g)$ additions. To make these steps more FHE-friendly and its multiplicative complexity (HAN et al., 2018a) introduced a batching technique that requires only two SIMD multiplications ($O(1)$) and $O(\log g)$ SIMD additions, using the matrix representation described before.

Given $Z \in \mathbb{R}^{m\times g}, \boldsymbol{v} \in \mathbb{R}^g$, our goal is to homomorphically calculate $Z \cdot \boldsymbol{v}$. Let $V \in \mathbb{R}^{m\times g}$ be the matrix of m copies of $\boldsymbol{v}$; if we encrypt Z and V in a row-linearization style and homomorphically multiply them, we get an encryption of the Hadamard product (elementwise):

$$Z \circ V = \begin{bmatrix} Z[1][1]\cdot v[1] & Z[1][2]\cdot v[2] & \dots & Z[1][g]\cdot v[g] \\ Z[2][1]\cdot v[1] & Z[2][2]\cdot v[2] & \dots & Z[2][g]\cdot v[g] \\ \vdots & \vdots & \ddots & \vdots \\ Z[m][1]\cdot v[1] & Z[m][2]\cdot v[2] & \dots & Z[m][g]\cdot v[g] \end{bmatrix}$$

Given this result, we just have to sum its columns, which can be done through rotations combined with additions. Let $\mathrm{Lrot}_i(A)$ be the matrix whose elements are rotated i places to the left (thinking in the linearized vector of A). Without loss of generality, assume that g is a power of two, if that's not the case, we can append zeros to it. Defines $A^{(1)} = Z \circ V$, the desired inner product is the first column of $A^{(g)}$ which is updated through the following $(\log g)$-step iterative process:

$$A^{(2^{k+1})} = A^{(2^k)} + \mathrm{Lrot}_{2^k}(A^{(2^k)}) \tag{4.3}$$

It's not difficult to prove that:

$$A_1^{(2^{k+1})} = \begin{bmatrix} \sum_{i=1}^{2^{k+1}} Z[1][i]\cdot v[i] \\ \sum_{i=1}^{2^{k+1}} Z[2][i]\cdot v[i] \\ \vdots \\ \sum_{i=1}^{2^{k+1}} Z[m][i]\cdot v[i] \end{bmatrix},$$

which directly implies $A^{(g)} = Z \cdot \boldsymbol{v}$ as wanted. To get a better idea of what 4.3 is doing, let's work in a 4-dimensional example: Denote $A^{(1)} = Z \circ V$ as $[\boldsymbol{a}_1,\ \boldsymbol{a}_2,\ \boldsymbol{a}_3,\ \boldsymbol{a}_4]$ for simplicity.

- Iteration $k = 0$, Compute $A^{(2)}$:

$$\begin{aligned} A^{(2)} &= A^1 + \mathrm{Lrot}_1(A^{(1)}) \\ &= [\boldsymbol{a}_1,\ \boldsymbol{a}_2,\ \boldsymbol{a}_3,\ \boldsymbol{a}_4] \\ &\ + [\boldsymbol{a}_2,\ \boldsymbol{a}_3,\ \boldsymbol{a}_4, \boldsymbol{a}_1] \\ &= [\boldsymbol{a}_1 + \boldsymbol{a}_2,\ \boldsymbol{a}_2 + \boldsymbol{a}_3,\ \boldsymbol{a}_3 + \boldsymbol{a}_4,\ \boldsymbol{a}_4 + \boldsymbol{a}_1] \end{aligned}$$

- Iteration $k = 1$, Compute $A^{(4)}$:

$$\begin{aligned} A^{(4)} &= A^2 + \mathrm{Lrot}_2(A^{(2)}) \\ &= [\boldsymbol{a}_1 + \boldsymbol{a}_2,\ \boldsymbol{a}_2 + \boldsymbol{a}_3,\ \boldsymbol{a}_3 + \boldsymbol{a}_4,\ \boldsymbol{a}_4 + \boldsymbol{a}_1] \\ &\ + [\boldsymbol{a}_3 + \boldsymbol{a}_4,\ \boldsymbol{a}_4 + \boldsymbol{a}_1, \boldsymbol{a}_1 + \boldsymbol{a}_2,\ \boldsymbol{a}_2 + \boldsymbol{a}_3] \\ &= [\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3 + \boldsymbol{a}_4, \ldots] \end{aligned}$$

Notice that until now, only one multiplication was used. The complete result of $A^{(g)}$ from Recurrence 4.3 is actually a $m \times g$ matrix, where the columns after the first are not important. To clean it out, we use another SIMD multiplication by a mask matrix $[\mathbf{1}\ \mathbf{0}\ \ldots\ \mathbf{0}]$ and use a similar procedure of rotation-addition as the above one to replace the null entries by replicates of the first column. The total multiplicative cost of the batch inner product is then 2.

The procedure of summing columns, removing garbage by the zero mask, and propagating first column entries, is defined as a separated algorithm for (HAN et al., 2018a): SumColVec, returning :

$$\mathrm{SumColVec}(A) = \begin{bmatrix} \sum_j a_{1j} & \ldots & \sum_j a_{1j} \\ \sum_j a_{2j} & \ldots & \sum_j a_{2j} \\ \vdots & \ddots & \vdots \\ \sum_j a_{mj} & \ldots & \sum_j a_{mj} \end{bmatrix},$$

for $A \in \mathbb{R}^{m \times g}$. The desired inner product is then $\mathrm{SumColVec}(Z \circ V)$, actually a matrix of duplicates g duplicates of it. Similarly, the SumRowVec algorithm returns a matrix with m replicates of the sum over A rows, using an analogous process of SIMD operations.

Now we have all tools to redefine the loop of Algorithm 1 to its vectorized form described in Algorithm 2.

Notation: For a fixed iteration $k = i$, $\mathbb{Z}_j = Z_{i,j}$ as in equation 4.2. A bolded number or constant denotes the $m \times g$ matrix of its replicates. $M^{\circ n}$ is the result of n nested Hadamard products of M: its elements are $M^{\circ n}_{i,j} = M^n_{i,j}$

After batch choice in the first algorithm, Algorithm 2 starts. Steps 1-4 calculates the first inner product; Step 5 sums the results over j, since for a particular j, we are dealing with just a subset (of size g) of data columns. Step 6 evaluates the approximate sigmoid function, and finally, loop 7-12 calculates the gradient and updates weights for all columns batches j.

All the operations in Algorithm 2 can be efficiently computed over encrypted data, changing Hadamard products by encrypted multiplication over a linearized encrypted matrix, also we need to apply continuous apply bootstrapping after a given number of iterations to be able to correctly evaluates an arbitrary number of iterations. For further

Algorithm 2 Vectorized weights update (HAN et al., 2018a)

```
Input: Matrices $Z_j, W_j, V_j \in \mathbb{R}^{m \times g}$, for $0 \leq j < (d+1)/g$
Output: Matrices $W_j^+, V_j^+$, for $0 \leq j < (d+1)/g$
 1: for $0 \leq j < (d+1)/g$ do
 2:     $M_j = Z_j \circ V_j$
 3:     $M_j = \text{SumColVec}(M_j)$
 4: end for
 5: $M = \sum_j M_j$
 6: $S = \mathbf{0.5} + \mathbf{0.15} \circ M - \mathbf{0.0015} \circ M^{\circ 3}$
 7: for $0 \leq j < (d+1)/g$ do
 8:     $S_j = S \circ Z_j$
 9:     $\Delta_j = \text{SumRowVec}(\text{S}_\text{j})$
10:     $W_j^+ = V_j - \boldsymbol{\gamma} \circ \Delta_j$
11:     $V_j^+ = (\mathbf{1} - \boldsymbol{\eta}) \circ W_j^+ + \boldsymbol{\eta} \circ W_j$
12: end for
13: return $W_j^+, V_j^+$ for $0 \leq j < (d+1)/g$
```

optimization details of this algorithm and the exact FHE formulation, the reader can refer to (HAN et al., 2018a).

4.3 Data Applications

The HEAAN scheme is implemented in a C++ package also named HEAAN (CHEON; KIM, et al., 2018). It supports all cryptographic primitives and bootstrapping. The logistic regression algorithm we describe in the previous chapter was implemented by the authors in the C++ package HELR (HAN et al., 2018b), which uses HEAAN package as the FHE base.

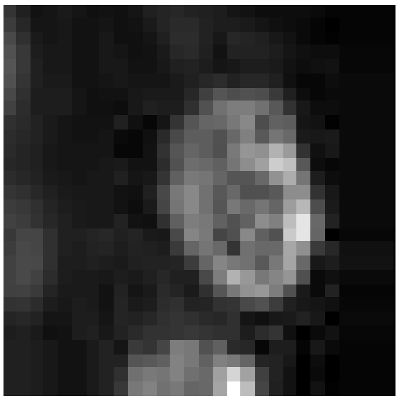
We tested the algorithm in the TissueMNIST dataset, which is part of the dataset collection from MedMNIST dataset (YANG et al., 2021). The original version has 28×28 images of kidney cortex cells labeled in eight classes. We adapt the dataset by compressing[1] the images to a smaller 14×14 set and transform it into a binary classification problem choosing only samples from two classes: "Collecting Duct" and "Proximal Tubule Segments" (Figure 7)

The images are matrixes of pixel values, so to train the model, we linearized them to 196-dimensional vectors. Subsetting just the above two classes, we got 92672 samples, with $\sim$ 90 MB

The CKKS ring parameters were set as:

$$N = 2^{15}$$
$$q = 2^{45}$$

[1] using the simple mean of the pixels


(a) Collecting Duct

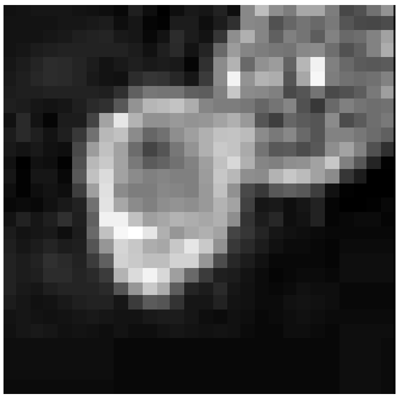

(b) Proximal Tubule Segments

Figure 7 – TissueMNIST

With a total number of 50 iterations, running bootstrapping every 3 iterations, using a computer with a 64-bit quad-core Intel Core i5-6200U 2.3GHz CPU, and 16GB of RAM, we got the following results:

KeyGen time	8.52 min
Encrypt time	19.17 min
Training time	6.19 hours
Public key size	2.62 GB

Table 1 – FHE perforamce

An important note is that 47% of training time was spent in bootstrapping.

In terms of the model, after 50 iterations, we got the following metrics:

	Encrypted	Unencrypted
Accuracy	64.6211%	64.6363%
AUC	81.7039%	81.6996%

Table 2 – Model Performance

By unencrypted, we mean Algorithm 2 applied to unencrypted data (with approximate sigmoid). Notice that encrypted/unencrypted metrics are very close; This result shows us that the noise introduced by encryption, homomorphic evaluation, and bootstrapping is negligible regarding model performance.

5 Conclusions and Further Work

Although the training time of 6.19 hours might appear a poor performance, considering that such a thing was impossible before 2009, the dataset has a considerable size, and the test was made on a personal computer, these results are actually encouraging, and reflects the evolution of FHE/ML research since Gentry's thesis.

On the practical use of the private logistic regression model, the use case of sending encrypted data to a cloud server and running the model there is well-mapped and could be done, even more efficiently since a cloud machine has usually better specifications than personal machines.

The limitation (besides training time) is that the only stopping criterion is the maximum number of iterations, so the user should specify an ad-hoc number that is not too high to affect training time, nor too low to affect model performance. Further work would be studying how to apply classical stopping criteria such as checking gradient norm, or the weight difference, but those procedures involve comparisons which is not a natively supported operation in FHE schemes.

Another natural further work is to focus on different models, such as tree-based or even neural networks. There is a lot of research in the domain of machine learning prediction/evaluation, where there is a pre-trained model, and the goal is to just apply it to an encrypted dataset, using the FHE; But there are very few research papers in training algorithms such as the one we presented here.

ML prediction is generally an easier problem and requires much fewer operations than maximizing likelihoods or doing backpropagation. It might be an interesting application in image or text classification where pre-trained models are very common and available.

References

ALBRECHT, Martin et al. **Homomorphic Encryption Security Standard**. Toronto, Canada, Nov. 2018.

BRAKERSKI, Zvika; GENTRY, Craig; VAIKUNTANATHAN, Vinod. **Fully Homomorphic Encryption without Bootstrapping**. [S.l.: s.n.], 2011. Cryptology ePrint Archive, Paper 2011/277. https://eprint.iacr.org/2011/277. Available from: <https://eprint.iacr.org/2011/277>.

CHEON, Jung Hee; HAN, Kyoohyung, et al. Bootstrapping for Approximate Homomorphic Encryption. In________. **Advances in Cryptology – EUROCRYPT 2018**. Cham: Springer International Publishing, 2018. P. 360–384. ISBN 978-3-319-78381-9.

CHEON, Jung Hee; KIM, Andrey, et al. **HEAAN package**. [S.l.]: GitHub, 2018. https://github.com/snucrypto/HEAAN.

________. Homomorphic Encryption for Arithmetic of Approximate Numbers. In________. **Advances in Cryptology – ASIACRYPT 2017**. Cham: Springer International Publishing, 2017. P. 409–437. ISBN 978-3-319-70694-8.

CORN, P.; MANZOOR, A.; DASH, S. **Primitive roots of Unity**. [S.l.: s.n.], 2016. Brilliant.org. Visited on 17 Nov. 2022. Available from: <https://brilliant.org/wiki/primitive-roots-of-unity/>.

COUNCIL OF EUROPEAN UNION. **General Data Protection Regulation - Council Regulation no 679/2016**. [S.l.: s.n.], 2016. https://gdpr-info.eu/.

DIJK, Marten van et al. Fully Homomorphic Encryption over the Integers. In________. **Advances in Cryptology – EUROCRYPT 2010**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 24–43. ISBN 978-3-642-13190-5.

FAN, Junfeng; VERCAUTEREN, Frederik. **Somewhat Practical Fully Homomorphic Encryption**. [S.l.: s.n.], 2012. Cryptology ePrint Archive, Paper 2012/144. https://eprint.iacr.org/2012/144. Available from: <https://eprint.iacr.org/2012/144>.

GENTRY, Craig. **A fully homomorphic encryption scheme**. 2009. PhD thesis – Stanford University. crypto.stanford.edu/craig.

GENTRY, Craig; HALEVI, Shai. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In________. **Advances in Cryptology – EUROCRYPT 2011**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. P. 129–148. ISBN 978-3-642-20465-4.

GENTRY, Craig; SAHAI, Amit; WATERS, Brent. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In: CANETTI, Ran; GARAY, Juan A. (Eds.). **Advances in Cryptology – CRYPTO 2013**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. P. 75–92. ISBN 978-3-642-40041-4.

HAN, Kyoohyung et al. **Efficient Logistic Regression on Large Encrypted Data**. [S.l.: s.n.], 2018. Cryptology ePrint Archive, Paper 2018/662. https://eprint.iacr.org/2018/662. Available from: <https://eprint.iacr.org/2018/662>.

________. **HELR package**. [S.l.]: GitHub, 2018. https://github.com/KyoohyungHan/HELR.

HERSTEIN, I.N. **Abstract Algebra**. [S.l.]: Wiley, 1996. ISBN 9780471368793.

HUYNH, Daniel. **CKKS explained, Part 5: Rescaling**. [S.l.: s.n.], Dec. 2020. Available from: <https://blog.openmined.org/ckks-explained-part-5-rescaling/>.

________. **CKKS explained: Part 1, Vanilla Encoding and Decoding**. [S.l.: s.n.], Mar. 2021. Available from: <https://blog.openmined.org/ckks-explained-part-1-simple-encoding-and-decoding/>.

JIA, Yan-Bin. **Polynomial Multiplication and FFT, Computer Science 477/577 Notes)**. [S.l.]: Iowa State University, Sept. 2022.

LIU, Tianren. On Basing Search SIVP on NP-Hardness. In________. **Theory of Cryptography**. Cham: Springer International Publishing, 2018. P. 98–119. ISBN 978-3-030-03807-6.

LYUBASHEVSKY, Vadim; MICCIANCIO, Daniele, et al. SWIFFT: A Modest Proposal for FFT Hashing. In________. **Fast Software Encryption**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. P. 54–72. ISBN 978-3-540-71039-4.

LYUBASHEVSKY, Vadim; PEIKERT, Chris; REGEV, Oded. A Toolkit for Ring-LWE Cryptography. In________. **Advances in Cryptology – EUROCRYPT 2013**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. P. 35–54. ISBN 978-3-642-38348-9.

________. On Ideal Lattices and Learning with Errors over Rings. In________. **Advances in Cryptology – EUROCRYPT 2010**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 1–23. ISBN 978-3-642-13190-5.

MCIVOR, James. **Ring Theory (Math 113)**. [S.l.]: University of California, Berkeley, Aug. 2016.

MICCIANCIO, Daniele. **Lattice Problems**. Theory of Cryptography Conference (TCC). 2007. Available from: <https://www.iacr.org/workshops/tcc2007/Micciancio.pdf>.

REGEV, Oded. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 56, n. 6, Sept. 2009. ISSN 0004-5411. DOI: 10.1145/1568318.1568324. Available from: <https://doi.org/10.1145/1568318.1568324>.

RIVEST, R L; ADLEMAN, L; DERTOUZOS, M L. On Data Banks and Privacy Homomorphisms. **Foundations of Secure Computation, Academia Press**, p. 169–179, 1978.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 120–126, Feb. 1978. ISSN 0001-0782. DOI: 10.1145/359340.359342. Available from: <https://doi.org/10.1145/359340.359342>.

STANIEWICZ, K.E. **A Fully Homomorphic Encryption scheme**. [S.l.]: University of Groningen, 2016.

SUN, Lawrence. Cyclotomic Polynomials in Olympiad Number Theory, 2013.

TENGAN, Eduardo; MARTINS, Sérgio Tadao. **Algebra: Highlights**. [S.l.]: American Mathematical Society, Dec. 2017.

TURNER, L Richard. **Inverse of the Vandermonde matrix with applications**. [S.l.], 1966.

YANG, Jiancheng et al. MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification. **arXiv preprint arXiv:2110.14795**, 2021.

Appendix

APPENDIX A – An ideal lattice scheme

A.1 Initial definitions

Definition A.1.1 (Relatively Prime Ideals)**.** Two ideals I and J of a ring R are considered relatively primes if $I + J = R$, where $I + J = \{i + j | i \in I, j \in J\}$.

Definition A.1.2 (Half-open parallelepiped)**.** For a given lattice basis $\mathbf{B}$, we define the half-open parallelepiped $\mathcal{P}(\mathbf{B})$ as:

$$\mathcal{P}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in [-1/2, 1/2) \right\}$$

Proposition A.1.1. *For a given* $\mathbf{t} \in \mathbb{R}^n$*, there exists an unique* $\mathbf{t}' \in \mathcal{P}(\mathbf{B})$*, such that,* $\mathbf{t} - \mathbf{t}' \in \mathcal{L}(\mathbf{B})$ *we define the operation of "reduction modulo the basis" as the map from* $\mathbf{t}$ *to* $\mathbf{t}'$*.*

$$\bmod \mathbf{B} : \mathbb{R}^n \to \mathcal{P}(\mathbf{B})$$
$$\bmod \mathbf{B} : \mathbf{t} \mapsto \mathbf{t} \bmod \mathbf{B} := \mathbf{t}'$$

This operation can be computed as:

$$\mathbf{t} \bmod \mathbf{B} = \mathbf{t} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{t} \rceil,$$

where $\lfloor . \rceil$ *round the entry to the closest integer. (STANIEWICZ, 2016)*

A.2 Abstract construction

Gentry (GENTRY, 2009) proposes an abstract scheme before making reference to ideal lattices. We need the following components:

- R - a fixed ring;
- $\mathbf{B}_I$ - a fixed basis of an ideal $I \in R$;
- IdealGen$(R, \mathbf{B}_I)$ - an algorithm that outputs $\mathbf{B}_J^{pk}, \mathbf{B}_J^{sk}$, public and secret basis of an ideal J, such that I and J are relatively prime ideals.
- Samp$(\mathbf{B}_I, x)$ - a function that samples from the coset $x + I$.

The scheme can be proved secure if we use the function $\text{Samp}(\mathbf{B}_I, x) = x + r \times s$, where r is chosen randomly from R, I is a principal ideal and s its generator.

Moreover, we assume that for every $\mathbf{t} \in R$, and a given basis $\mathbf{B}_M$ of an ideal $M \subset R$, there exists a unique representative $\mathbf{t} \bmod \mathbf{B}_M$, that can be efficiently computed.

Then we can construct a (somewhat) homomorphic encryption scheme, using the functions KeyGen, Enc, Dec, and Eval:

- $\text{KeyGen}(R, \mathbf{B}_I)$ - sets public and secret basis $(\mathbf{B}_J^{pk}, \mathbf{B}_J^{sk}) \leftarrow \text{IdealGen}(R, \mathbf{B_I})$. Then the public key is $\text{pk} = (R, \mathbf{B}_I, \mathbf{B}_J^{pk}, \text{Samp})$ and secret key includes the secrete basis $\text{sk} = (R, \mathbf{B}_I, \mathbf{B}_j^{pk}, \text{Samp}, \mathbf{B}_J^{sk})$
- $\text{Enc}(\text{pk}, \pi) = \text{Samp}(\mathbf{B}_I, \pi) \bmod \mathbf{B}_J^{pk}$
- $\text{Dec}(\text{sk}, \psi) = (\psi \bmod \mathbf{B}_J^{sk}) \bmod \mathbf{B_I}$
- $\text{Eval}(\text{pk}, F, \Psi)$ - takes a circuit F of additions and multiplications mod-$\mathbf{B}_I$ and invoked Add and Mult below, in the proper order.

$$\text{Add}(pk, \psi_1, \psi_2) = \psi_1 + \psi_2 \bmod \mathbf{B}_J^{pk}$$
$$\text{Mult}(pk, \psi_1, \psi_2) = \psi_1 \times \psi_2 \bmod \mathbf{B}_J^{pk}$$

More generally, given a circuit F of addition and multiplications mod-$\mathbf{B}_I$, the Eval function is:

$$\text{Eval}(\text{pk}, F, \Psi) = g(F)(\Psi) \bmod \mathbf{B}_J^{pk},$$

where $g(F)$ is the generalized circuit that replaces $\text{Add}_{\mathbf{B_I}}$ and $\text{Mult}_{\mathbf{B_I}}$ by "+" and "×" operations of the ring R.

It's not trivial that correctness property (3.2). In fact, it holds only for a specific group of circuits.

Let's define X_{Enc} as the image of Samp and $X_{Dec} = R \bmod \mathbf{B_J^{sk}}$, i.e, the set of indistinguishable representatives of cosets of J.

We call $\mathcal{F}_\mathcal{E}$ a set of **permitted circuits** if its a subset of

$$\{F : \forall (x_1, \ldots, x_n) \in X_{Enc}, g(F)(x_1, \ldots, x_n) \in X_{Dec}\},$$

i.e., the set of mod-$\mathbf{B}_I$ circuits such that its generalization is in X_{Dec} when the input is in X_{Enc}. For the permitted circuits, one can prove that homomorphic decryption works correctly.

A.3 Concrete construction using ideal lattices

Let $f \in \mathbb{Z}[x]$ be a monic polynomial of degree n and define the ring:

$$R = \mathbb{Z}[x]/(f)$$

whose elements can be uniquely represented by:

$$\Big\{\overline{\sum_{i=1}^{n} \alpha_i x^{i-1}} | \alpha_i \in \mathbb{Z}\Big\},$$

where $\overline{P(x)} = P(x) \bmod f(x)$, i.e, the remainder in the polynomial division of P by f. Since $\deg(f) = n$, the polynomials in the above set have degrees up to $n-1$. So, there is a bijective map $\varphi : R \to \mathbb{Z}^n$, that takes $r = \sum_{i=1}^{n} \alpha_i x^{i-1} \in R$ and makes $\varphi(r) = (\alpha_1, \ldots, \alpha_n)^T$

Let $v \in R$ be a generator of an ideal J. We can construct an ideal lattice using the so-called *rotation basis* $\mathbf{B_J}$ with columns:

$$b_i = \varphi(b_i^*),$$
$$\text{where } b_i^* = v \times x^{i-1} \bmod f(x)$$

One can easily prove that there is a bijection between J and $\mathcal{L}(\mathbf{B_J})$.

Lemma A.3.1. Let $r \in \mathbb{Z}^n,\ j \in \mathcal{L}(\mathbf{B_J})$ where J is an ideal of $R = \mathbb{Z}[x]/(f)$. Then:

$$(r + j) \bmod \mathbf{B_J} = r \bmod \mathbf{B_J}$$

Gentry's original scheme, starts with R, an ideal I, and $\mathbf{B_I}$ a basis of $\mathcal{L}(\mathbf{B_I})$ (not necessarily the rotation one). Then the algorithm IdealGen described in previous section, chooses another ideal J such that I and J are relatively primes, and outputs $\mathbf{B_J^{pk}}, \mathbf{B_J^{sk}}$ basis of $\mathcal{L}_J$. A possible choice for the secret basis in the rotation one, and for the public basis, one can use the hermite normal form of $\mathbf{B_J^{sk}}$.

The algorithms Enc, Dec and Eval works as described before.

Using ideal lattices, we can redefine X_{Enc} and X_{Dec} here in order to obtain a geometric interpretation. At first, notice that $X_{Dec} = R \bmod \mathbf{B_J^{sk}} = \mathcal{P}(B_J^{sk})$.

Let $\mathcal{B}(r)$ be the origin centered ball in $\mathbb{R}^n$ with radius r. Then, defines:

- r_{Enc} - a value s.t. $\mathcal{B}(r_{Enc})$ is the smallest ball satisfying $X_{Enc} \subseteq \mathcal{B}(r_{Enc})$
- r_{Dec} - a value s.t. $\mathcal{B}(r_{Dec})$ is the greatest ball satisfying $\mathcal{B}(r_{Dec}) \subseteq X_{Dec} = \mathcal{P}(\mathbf{B_J^{sk}})$

Now the set of permitted circuits become:

$$\mathcal{F_E} = \{F : \forall (x_1, \ldots, x_n) \in \mathcal{B}(r_{Enc}), g(F)(x_1, \ldots, x_n) \in \mathcal{B}(r_{Dec})\}$$

Fixed r_{Enc}, r_{Dec} we want to know what is $\mathcal{C}_{\mathcal{E}}$. We can bound $||g(F)(x_1, \ldots, x_n)||$ by bounding $||u + v||$ and $||u \times v||$. We know that $||u + v|| \leq ||u|| + ||v||$, by the triangle inequality. One can explicit a factor $\gamma_{Mult}(R)$, depending only on the ring, s.t. $||u \times v|| \leq \gamma_{Mult}(R)(||u|| \times ||v||)$. Then, Gentry shows that the scheme correctly evaluates any circuit F with depth at most

$$\log \log r_{Dec} - \log \log(\gamma_{Mult}(R) \cdot r_{Enc})$$

Then the author dedicates a lot of chapters show tweaks to simplifying the decryption circuit complexity, such that the scheme can evaluate it, and a bootstrappable scheme is achieved.

It's desirable to maximize r_{Dec}, and minimize r_{Enc} and $\gamma_{Mult}(R)$. Gentry's work show that:

$$r_{Dec} = \frac{1}{2||\mathbf{B_J^{sk*}}||}, \text{ where}$$
$$\mathbf{B_J^{sk*}} = ((\mathbf{B_J^{sk}})^{-1})^T$$

Also, by choosing $R = \mathbb{Z}[x]/(f)$ with $f(x) = x^n - 1$, then it can be proved that:

$$\gamma_{Mult}(R) \leq \sqrt{n}$$

Now we'll prove that our SHE scheme correctly decrypts a *valid cyphertext* (a ciphertext that can be outputed by Enc or Eval).

Theorem A.3.1 (Identity correctness). *Consider $\pi \in \Pi$, where $\Pi = \{\pi | \pi \in \mathcal{P}(\mathbf{B_I})\}$ is the plaintext space. If $\mathrm{Samp}(\pi) \in \mathcal{B}(r_{Dec})$, then*

$$\mathrm{Dec}(sk, \mathrm{Enc}(pk, \pi)) = \pi$$

Proof. The cyphertext is

$$\begin{aligned}\psi &= \mathrm{Samp}(\mathbf{B_I}, \pi) \bmod \mathbf{B_J^{pk}} \\ &= (\pi + i) \bmod \mathbf{B_J^{pk}},\end{aligned}$$

for some $i \in \mathcal{L}_I$. We can write $\psi = \pi + i + j$, for some $j \in \mathcal{L}_J$ because:

$$\begin{aligned}\psi &= (\pi + i) \bmod \mathbf{B_J^{pk}} \\ &= (\pi + i) - \mathbf{B_J^{pk}} \times c,\end{aligned}$$

where $c = \lfloor(\mathbf{B_J^{pk}})^{-1}(\pi + i)\rceil$. Since $c \in \mathbb{Z}^k$, if $j = \mathbf{B_J^{pk}} \times (-c)$, then $j \in \mathcal{L}_J$, by definition of lattice.

Then, by using Lemma A.3.1 and the fact that $(\pi + i) \in \mathcal{B}(r_{Dec}) \subseteq \mathcal{P}(\mathbf{B}_J^{sk})$:

$$\begin{aligned}
&\mathrm{Dec}(sk, \mathrm{Enc}(pk, \pi)) \\
&= \mathrm{Dec}(sk, \psi) \\
&= ((\pi + i + j) \bmod \mathbf{B}_J^{sk}) \bmod \mathbf{B}_I \\
&= ((\pi + i) \bmod \mathbf{B}_J^{sk}) \bmod \mathbf{B}_I \\
&= (\pi + i) \bmod \mathbf{B}_I \\
&= \pi
\end{aligned}$$

□