ANALYSIS AND COMPARISON OF FULLY HOMOMORPHIC ENCRYPTION APPROACHES OVER INTEGERS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CANSU BOZKURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

FEBRUARY 2022

Approval of the thesis:

ANALYSIS AND COMPARISON OF FULLY HOMOMORPHIC ENCRYPTION APPROACHES OVER INTEGERS

submitted by **CANSU BOZKURT** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Kestel
Dean, Graduate School of **Applied Mathematics** _____

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** _____

Assoc. Prof. Dr. Murat Cenk
Supervisor, **Cryptography, IAM, METU** _____

Dr. Cansu Betin Onur
Co-supervisor, **Cryptography, IAM, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Oğuz Yayla
Cryptography, IAM, METU _____

Assoc. Prof. Dr. Murat Cenk
Cryptography, IAM, METU _____

Assoc. Prof. Dr. Fatih Sulak
Department of Mathematics, Atılım University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: CANSU BOZKURT

Signature :

ABSTRACT

ANALYSIS AND COMPARISON OF FULLY HOMOMORPHIC ENCRYPTION APPROACHES OVER INTEGERS

BOZKURT, CANSU
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Murat Cenk
Co-Supervisor : Dr. Cansu Betin Onur

February 2022, 53 pages

The time period after the mid-20th century was named as information age or digital age. In that age, the world is being digitalized very fastly. The amount of data transferred and processed online is increasing rapidly. As a result, data protection became an essential topic for researchers. To process or make a computation on the encrypted data deciphering ciphertext first causes a security flaw. Homomorphic encryption (HE) algorithms were designed to make computations on data without deciphering it. However, HE algorithms are able to work for a limited amount of processing steps. Fully homomorphic encryption (FHE) algorithms are developed to solve this problem. It is feasible to apply any accurately calculable operation on encrypted data. This thesis presents definitions, properties, applications of FHE. Some constructions of FHE schemes based on the integers are also analyzed. Furthermore, the computational complexity of two algorithms, namely the DGHV scheme and Batch DGHV scheme has been computed and their efficiency are compared based on the complexities. While the DGHV scheme encrypts the one-bit message, the batch DGHV scheme encrypts an ℓ-bit message vector m at a time. The primary purpose is to research which option is more efficient for encrypting ℓ-bit messages. The first option is to use the DGHV scheme for ℓ-times. The second option is to use the batch DGHV scheme one time. We conclude that for message size ℓ when security parameter $\ell \leq \lambda^{3/2}$ using batch scheme is more efficient than using DGHV scheme.

Keywords: Fully Homomorphic, Batch Fully Homomorphcic, Homomorphic Encryption, Bootstrapping, Squashing

ÖZ

TAM SAYILAR ÜZERİNDEKİ TAM HOMOMORFİK ŞİFRELEME YAKLAŞIMLARININ ANALİZİ VE KARŞILAŞTIRILMASI

BOZKURT, CANSU
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Murat Cenk
Ortak Tez Yöneticisi : Dr. Cansu Betin Onur

Şubat 2022, 53 sayfa

20. yüzyılın ortalarından sonraki dönem bilgi çağı veya dijital çağ olarak adlandırılmıştır. Bu çağda dünya çok hızlı dijitalleşmiştir. Çevrimiçi olarak aktarılan ve işlenen veri miktarı hızla artmıştır. Sonuç olarak, veri koruma araştırmacılar için önemli bir konu haline gelmiştir. Şifreli bir veri üzerinde işlem veya hesaplama yapmak için şifrenin önce deşifre edilmesi bir güvenlik açığına neden olmaktadır. Homomorfik şifreleme algoritmaları, verileri deşifre etmeden hesaplama yapmak için tasarlanmıştır. Ancak biraz homomorfik şifrelemeler algoritmaları sınırlı miktarda işlem adımı için uygulanabilirler. Bu sorunu çözmek için tam homomorfik şifreleme algoritmaları geliştirilmiştir. Bu yöntemle şifrelenmiş verilere herhangi bir hesaplanabilir fonksiyon uygulanabilir. Bu tez içerisinde tam homomorfik şifrelemenin tanımları, özellikleri ve uygulamaları sunulmuştur. Bazı tamsayılara dayalı tam homomorfik şifreleme şemalarının yapıları da analiz edilmiştir. Ayrıca, DGHV şeması ve gruplu DGHV şeması (bir düz metin vektörünü tek bir şifreli metin halinde homomorfik olarak şifrelemeyi destekleyen bir şema) olmak üzere iki algoritmanın hesaplama karmaşıklığı hesaplanmış ve verimlilikleri karşılaştırılmıştır. DGHV şeması tek bitlik mesajı şifrelerken, gruplu DGHV şeması her seferinde ℓ bitlik mesaj vektörünü şifreler. Bu tezde Öncelikli olarak ℓ-bit uzunluğundaki bir mesajın şifrelenmesinde hangi şemanın kullanımının daha verimli olduğu araştırılmıştır. Şifreleme için DGHV şemasını ℓ kere çalıştırmak ile gruplu DGHV şemasını bir kere çalıştırmak arasındaki verimlilik farkı

hesaplanmıştır. Yapılan hesaplamalar sonucunda ℓ uzunluğundaki bir mesaj için güvenlik parametresi $\ell \leq \lambda^{3/2}$ olduğunda gruplu DGHV şemasını kullanmanın DGHV şemasına göre daha verimli olduğu sonucuna ulaşılmıştır.

Anahtar Kelimeler: Tam Homomorfik Şifreleme, Gruplu Tam Homomorfik Şifreleme, Homomorfik Şifreleme, Önyükleme Methodu

To my family

ACKNOWLEDGMENTS

I would like to express my very great appreciation to my thesis supervisor Assoc. Prof. Dr. Murat Cenk for his patient guidance, enthusiastic encouragement and valuable advices during the development and preparation of this thesis. His willingness to give his time and to share his experiences has brightened my path.

I would also thank to my co-supervisor Dr. Cansu Betin Onur for her guidance, support and motivation throughout this thesis.

Furthermore, I would like to thank my family and my friends for listening to my concerns, believing in me and their supports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

$\mathbb{Z}$	Integers
$\mathbb{Q}$	Rational Numbers
$\mathbb{R}$	Real Numbers
$AGCD$	Approximate Gratest Common Divisor
HE	Homomorphic Encryption
$SwHE$	Somewhat Homomorphic Encryption
FHE	Fuully Homomorphic Encryption

CHAPTER 1

INTRODUCTION

Cryptography is an essential tool for information security. People have continued to develop more complex cryptographic encryption systems, starting with simple encryption systems such as shift cipher and substitution cipher [19]. The most significant purpose of these developments is to communicate and store private information securely. Over time, cryptography has become an interdisciplinary science, and cryptographic algorithm design and development have come to an important place.

Cryptographic studies are grouped into two main categories as symmetric keys and asymmetric keys [26]. One of the differences between the two groups is the number of keys. While only one key is used in symmetric algorithms, two keys are used in the asymmetric version, private and public. These keys encrypt and decrypt the data respectively. Another significant difference is that symmetric systems are more efficient than asymmetric systems. Efficiency is an essential feature for cryptography because the low efficiency of a highly secure design renders the system useless. Today, with the development of technology, much more reliable and efficient algorithms are needed. People need these algorithms to store and process their private information. For example, being able to perform an operation on encrypted data without decrypting the data in the cloud system, which has been widely used, and delivering this data securely to the other party has become an important goal for studies. As seen in the example, the biggest problem is deciphering the encrypted data before working on it. It means putting the data we encrypted to protect it at risk again. Researchers have proposed Homomorphic encryption as a solid solution for this scenario.

Homomorphic encryption basically can be expressed mathematically as: $Enc\ (p_1 *$

$p_2) = Enc(p_1) * Enc(p_2)$, where p_1 and p_2 are two encrypted messages, and Enc is the encryption function. Here, homomorphic encryptions can be divided into three groups. These are; partially homomorphic (PHE), somewhat homomorphic (SwHE), and fully homomorphic encyrption (FHE). These groups are constructed according to the capability of encyrption algorithm [23]. Suppose that the scheme supports an unlimited number of one operation, namely, addition or multiplication on encrypted data. In that case, it is named partially homomorphic. If it enables both operations to up to a certain level of complexity, somewhat homomorphic, if it allows both operations an unlimited number of times, it is named as fully homomorphic.

FHE is a popular research subject. It mainly intends to compute unlimited arithmetic operations on encrypted data. The schemes that recommended so far are not practical enough. Calculating many functions on ciphertext is not easy because of the big ciphertext size. That is why this topic is ideal for studying and developing.

This thesis includes definitions, properties, applications of FHE. Some constructions of FHE schemes based on the integers are also studied. Furthermore, the computational complexity of two algorithms, namely the DGHV scheme and Batch DGHV scheme has been computed and their efficiency are compared based on the complexities. While the DGHV scheme encrypts the one-bit message, the batch DGHV scheme encrypts an ℓ-bit message vector m at a time. The essential purpose is to research which option is more efficient for encrypting ℓ-bit messages. The first alternative is to use the DGHV scheme for ℓ-times. The second one is to use the batch DGHV scheme one time. We conclude that for message size ℓ when security parameter $\ell \leq \lambda^{3/2}$ using batch scheme is more efficient than using DGHV scheme.

1.1 Historical Process and Literature Review

RSA is a public-key algorithm, adversity of the factorization problem is the key point for the security of RSA scheme. Two different prime numbers choosen in key generation step. They are named as p and q respectively. To improve security p and q randomly designated and should have close lengths. Compute $n = p.q$ and, n is the reduction value of private and public keys. Compute $\varphi(n) = (p-1)(q-1)$,

which is the totient of these numbers. Generate a positive integer e less then $\varphi(n)$ so, $gcd(e, \varphi(n))$ are equal to 1. Then define d, the $d.e \equiv 1$ in module $\varphi(n)$. To encrypt M and create C, compute $C = M^e$ in module n and decrypt C to obtain the message M, compute $M = C^d$ in module n. The RSA algorithm has multiplicative homomorphism property, so $C_1 = M^{e_1}$ and $C_2 = M_2^e$ are two encrypted messages, and

$$(C_1 \times C_2) = M_1^e \times M_2^e = (M_1 \times M_2)^e = (C_1) \times (C_2).$$

After a while, research named "privacy homomorphism" is led by designers of RSA [21], and this system provides only additive or only multiplicative homomorphism called as "Partially Homomorphic scheme." Goldwasser-Micali [15], Benaloh [1], Damgård–Jurik and Paillier [27] cryptosystems can be given as additive homomorphic examples for the Partially homomorphic scheme.

Later, researchers were interested in the question, "Can homomorphic scheme support both additive and multiplicative homomorphism?" Nearly 30 years later, BGN cryptosystem was presented in 2005 [3]. The scheme, while not the desired encryption system, is a significant improvement. Because it provides an unlimited number of homomorphic addition, it supports only one multiplication. This type of scheme that performs both operations limited times is named as SwHE. Gentry represented very first FHE scheme in his Ph.D. thesis. [12]. He found a way called "Bootstrapping", which can convert a SwHE scheme to a FHE scheme. Gentry uses Ideal Lattices to design a somewhat homomorphic scheme in his breakthrough work. According to his study, arbitrary computation can be operated on ciphertext without any decryption process. To understand the design behind Gentry's scheme, we can begin with the noise component on the ciphertexts. The biggest problem is noise component proliferates when one operates on ciphertexts. If the noise reaches some bound, which is dependent on the scheme, the decryption process can not be made correctly. Bootstrapping technique decreases the noise and achieves a FHE scheme. After Gentry's work, theoretically, anyone can perform unlimited homomorphic arbitrary operations on the ciphertext. The reason why it is called in theory is that there is no convenient implementation of this scheme efficiently.

After this groundbreaking discovery, new simpler FHE schemes are presented to increase efficiency and facilitate the implementation . These schemes basically can be

grouped as Lattice-based [13], over the integers [30], Learning with Errors Problem based [5], and Ring Learning with Errors Problem based [6]. Among these schemes, FHE over the integers is a remarkable candidate to work on since its arithmetic are more straightforward than the others.

1.2 Outline

In this thesis, the FHE over the integer (DGHV) scheme was studied. Generally, our focus is on analyses that will increase the scheme's efficiency. In chapter 2, necessary definitions are made to understand the subject better.

Chapter 3 demonstrates the structure of the functions that can be evaluated homomorphically using arithmetic circuits. And also, many real-life applications of Homomorphic encryption are described. These applications have been developed mainly in the fields of health, finance, advertising, and education.

Chapter 4 describes the symmetric and asymmetric key versions of the DGHV scheme. Then, we demonstrate why these schemes are homomorphic under addition and multiplication operations. Lastly, we give the general review of FHE schemes and a way to convert a SwHE scheme to an FHE scheme.

Chapter 5 mentions some efficiency problems of the DGHV scheme and ways to improve it. These are finding more efficient SwHE schemes, extending the message size or message space, increasing the speed of bootstrapping process, eliminating the bootstrapping procedure, eliminating the squashing process and shortening the existing public key. We generally focus on "extending message size and message space" solutions among them.

In chapter 6, we compute and compare the complexities of two algorithms, namely, DGHV scheme [30] and batch DGHV scheme [7].

CHAPTER 2

PRELIMINARIES

This section contains explanations of some concepts that will be used later. In chapter 3, we demonstrate the function's structures that can be operated homomorphically working with arithmetic circuits. In chapters 4 and 5, many terms and methods to construct a FHE scheme like 3 to 2 trick, sparse subset sum problem, knapsack problem, and approximate greatest common divisor problem. Finally PHE, SwHE, and FHE are used throughout the thesis.

2.1 Definition and Properties

In this part, some definitions and explanations are given to understand the upcoming chapters easier. Some of the definitions given in this thesis are based on Genry's work [13].

Definition 2.1.1. (Approximate Greatest Common Divisor Problem (AGCD) [30]) Given polynomially many samples from $D_{\gamma,\rho}(p) = \{$choose $q \xleftarrow{\$} \mathbf{Z} \cap [0, 2^{\gamma}/p]$, $r \xleftarrow{\$} \mathbf{Z} \cap (-2^{\rho}, 2^{\rho})$: output $x = pq + r\}$ p is an η-bit odd integer then, output p.

The difficulty of the SwHE scheme is based on this AGCD problem. For our scheme, one wants to find secret value p given many values of public key member which have the form $x_i = pq_i + r_i$ where $|r_i|$ is small noise.

As an example : Lets say our set is $TestSeq = [399, 710, 105, 891, 402, 102, 397]$ the optimum GCD is approximately 100.18867794375123.

Lemma 2.1.1. *(Simplified Leftover Hash Lemma [16]) F be a family of 2-universal*

hash functions from A to B. Assume that $f \xleftarrow{R} F$ and $a \xleftarrow{R} a$ are selected uniformly and independently. A hash function F from A to B is a family of 2-universal hash functions. Let's choose $f \xleftarrow{R} F$ and $a \xleftarrow{R} a$. Then, $(f, f(a))$ is $1/2\sqrt{|B|/|A|}$ - uniform over $F\chi B$.

AGCD is the main challenge behind the DGHV scheme [30].

Arithmetic circuits are an essential tool for both encryption and evaluation steps in FHE schemes. The following two-term, namely arithmetic circuit and circuit depth, should be understood well.

Definition 2.1.2. (Arithmetic circuits [29]) An arithmetic circuit C over the field F and the set of variables Y (usually, $Y = \{y_1, ..., y_n\}$) is a acyclic directed graph as follows. The vertices of C are called gates. All the gates in C of in-degree 0 is labeled by either a variable from Y or a field element from F. Every other gate in C is labeled by either $\times$ or $+$ and has in-degree 2.

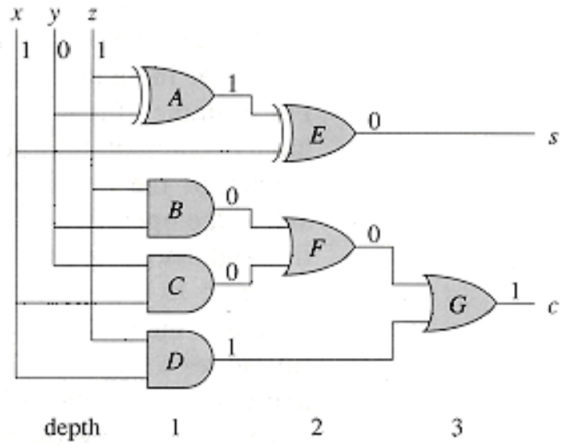

Figure 2.1: An Example of Circuit Construction [9]

In figure 2.1, we denote

- $s = parity(x, y, z) = x \oplus y \oplus z$,
- $c = majority(x, y, z) = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$.

Let $x = 1$, $y = 0$ and $z = 1$ and the summation of $x + y + z = 2$. To implement these problem to arithmetic circuit first we apply $c = (1 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 1) = 1$ and

then $s = 1 \oplus 0 \oplus 1 = 0$. One can see that $< c, s > = < 1, 0 >$, and it is interpreted as $(10)_2 = (2)$. Note that in the Figure 2.1, A and E are XOR-gates; B, C and D are AND-gates; F and D are OR-gates.

Definition 2.1.3. (Circuit depth) Depth of a circuit is defined as worst-case "running time" this circuit. The depth of an input wire is 0. If a combination item has inputs $a_1, a_2, ..., a_n$ at depths $dp_1, dp_2, ..., dp_n$ respectively, then its outputs have depth $max\{dp_1, dp_2, ..., dp_n + 1\}$. The depth of a combination item is the depth of its outputs. The maximum depth of any combination element is defined as depth of the combination circuit. We prevented the combination circuits from including cycles as a result the different impressions of depth are expressed explicitly.

The following definition is about a technique used for doing more efficient computation on circuit theory.

Definition 2.1.4. (Three for two Trick [20]) This method is used to more efficiently calculate the carry bits during the addition process. Given three integer a, b, c such that $a + b + c = m + n$. Here m is the bits formed by exclusive-OR of the bits of a, b, c and n is the bits formed by the carry.

$$m_i = a_i \ \oplus b_i \ \oplus c_i$$

$$\begin{aligned} n_0 = 0;\ n_{i+1} &= 1 \ if \ \{(a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i) == 1\} \\ &= 0 \ otherwise \end{aligned}$$

The following six definitions are most common terms used in thesis to explain what fully homomorphic encyption is.

Definition 2.1.5. (Correct Homomorphic Decryption) The scheme for a ciphertexts $\vec{c} = \langle c_1, \cdots, c_t \rangle$ satisfies $c_i \leftarrow Encrypt_\varepsilon(pk, m_i)$ is correct if the output of a decryption function, which takes evaluate function and secret key as input, is equal to outputs of t-input circuits C. It can be summarize as following

$$Decrypt(sk,\ Evaluate(pk, C, \vec{c})) = C(m_1, \cdots, m_t).$$

Definition 2.1.6. (HE [30]) ε is a scheme that includes four algorithms namely, KeyGen, Encrypt, Decrypt, Evaluate. Suppose that it is homomorphic for a class $\mathbb{C}$

of circuits if it is correct for all circuits $C \in \mathbf{C}$. ε is fully homomorphic if it is correct for all boolean circuits.

Definition 2.1.7. (Compact Homomorphic Encryption) ε scheme which has four component namely KeyGen, Encrypt, Decrypt, Evaluate is called compact if it is bounded a predefined boundary polynomial $bound(\lambda)$, and for any circuit C and ciphertext tuple $\vec{c} = \langle c_1, \cdots, c_t \rangle$ size of the output $Evaluate(pk, C, \vec{c})$ is also bounded with $bd(\lambda)$.

Definition 2.1.8. (Augmented Decryption Circuits [29]) ε is an encryption scheme that has four components KeyGen, Encrypt, Decrypt and Evalute, and its decryption scheme applied as a circuit. Augmented decryption circuit is a combination of two circuits and these two circuits accept secret key and two ciphertexts as inputs. First one carries out decryption process for two ciphertexts and adds outputs bits mod 2. Second one again decrypts these two ciphertexts than multiplies the resulting bits mod 2. As augmented decryption circuit is dependent only security parameter λ, it symbolized with $D_\varepsilon(\lambda)$.

Definition 2.1.9. (Bootstrappable Encryption) One say that the scheme ε is bootstrappable if $D_\varepsilon(\lambda) \subseteq C_\varepsilon(\lambda)$. Here, ε is the homomorphic encryption scheme, $C_\varepsilon(\lambda)$ is a circuits set which ε is correct, and $D_\varepsilon(\lambda)$ is the augmented decryption circuits.

Definition 2.1.10. (Knapsack Problem [22]) Define two sets, all elements of which are integers, namely $Val = \{v_1, ..., v_n\}$ and $Wt : \{w_1, ..., w_n\}$ which symbolize values and weights related with n elements respectively. Also, given integer W, which represents the capacity of knapsack, find the maximum value subset of Val and Wt such that knapsack can carry items without tearing. This means that sum of the chosen items' weight must be smaller or equal to knapsack weight capacity W. As an example: Let's say $Val : \{60, 100, 120\}$, $Wt : \{10, 20, 30\}$ and $W = 50$ then solution set is $S : \{30, 20\}$

Now, we may define the following three items to figure out how we categorize homomorphic encryption.

Definition 2.1.11. (Partially Homomorphic Encryption) Partially Homomorphic Encryption(PHE) only allows a single arithmetic operation, namely addition or multiplication, to be performed unlimited times on encrypted data.For example, while the

Paillier scheme [27] is additive homomorphic, RSA [21] and Elgamal [10] systems are multiplicative.

Definition 2.1.12. (Somewhat Homomorphic Encryption) The scheme permits for both addition and multiplication on ciphertext, but only up to a certain complexity. An example of this is the DGHV scheme [30], described in Chapter 4.

Definition 2.1.13. (Fully Homomorphic Encryption) The scheme permits limitless number of addition and multiplication on ciphertext.

CHAPTER 3

HOMOMORPHIC ENCRYPTION AND ITS APPLICATIONS

In our rapidly growing and developing world, storing and managing data in cloud systems plays an important role. In addition to processing and storing data, protecting this data is also a substantial point. Encryption of the data before it is sent to the cloud is a proper and logical process to keep it secure. However, deciphering the data for the operations to be performed on it during this journey endangers the data. Therefore, it should be encrypted homomorphically so that it can be processed even when the data is encrypted.

In this chapter, real-life applications of Homomorphic encryption are described. These applications have been developed mainly in the fields of health, finance, advertising, and education.

3.1 Practical Applications for Homomorphic Encryption

In this section, applications and functions of homomorphic encryption in fields like; health, finance, and advertisement are given. Detailed explanations of described encryption algorithms are provided in the following chapter. Today, many hospitals prefer to keep their electronic health records in the cloud instead of holding them on their local computers. The main reasons for this are the increased storage space due to the rising data density and the difficulty of calculating the growing data. Patient data are encrypted to securely transport, store, and process in the cloud. These data are often used to make statistical interpretations. Some of the most used functions for statistical operations are: mean, standard deviation, logistical regressions. If one

considers these algorithms specifically, it will be sufficient to use the SwHE scheme [24]. Note that while the averages algorithm has no multiplication, the standard deviation algorithm has just one multiplication also, the logistical regressions algorithm has more than one multiplications depending on the required accuracy.

3.2 Cloud Services

Even though cloud services reach a wide range of users, there are still questions about data protection. This section will describe some applications that use SwHE scheme to protect customers' confidentiality. In all these scenarios, the data that comes to the cloud in an encrypted way and is operated by the cloud to supply a service to the person is considered. Two ideas to consider here, the first is to encrypt the function itself, and the second is to encrypt the data. In the following sub-sections, the headers will contain information about whether the data itself, the function or both are encrypted.

3.2.1 In Healthcare: Secret data and Public functions

In the article [2], a system was suggested named Patient Controlled Encryption System. According to this article, patients decide who can see their results and who can not. While this approach helps protect data, it does not help to work on it safely in the cloud. Thanks to the implementation of FHE, it has become possible to operate on the encrypted data on the patient side. The latest updates, warnings, and recommendations will be securely transmitted to the patient with the FHE on encrypted data. The main functions used here are statistical functions, namely mean, standard deviation, and logistic regression. Encrypted entries usually consist of information about our health, such as test results and illness history. Finally, when it comes to public health, priority is given to the protection of data rather than the preservation of function.

3.2.2 In Economy and Finance: Secret Data and Secret functions

In finance sector, unlike health, both data and the function that accepts encrypted data as input are encrypted [24]. The data in this area can be company-specific confidential information related to collaborations, warehouse activities, and investment decisions. The confidentiality of the functions used here is as vital as the confidentiality of the data. The information such as financial analysis and predictive product models obtained by the company by spending big money becomes predictable by knowing the function used. Thanks to the FHE, these functions can be calculated implicitly. For example, the client uploads the encrypted function to the system (such as a statistical program with encrypted data). Client's public key used to encrypt transferred data. Later the data transferred to the cloud. Finally, the cloud delivers the encrypted output to the client, and the data can be read by using its private key.

3.2.3 Advertising and Pricing

It is not a coincidence that when you pass by the coffee shop, you receive a discount coupon from that coffee shop, advertisements for different places on your phone on Friday nights, and if you are a well-groomed person, you are constantly sent cosmetics ads [18]. Today, advertising companies follow many different ways to attract the attention of the consumer. Devices we use in our daily lives transmit our location, e-mail addresses, and keywords we searched the Internet to the other party [28]. When such conceptual data is transformed to the cloud server, it becomes available to many companies. In this way, the advertising firm chooses the advertisement to be sent to you with a smart strategy, not arbitrarily.

The biggest problem in the scenario described in the first paragraph is that the application users give too much information to the other party, and FHE can be considered a solution. Namely, all private information is encrypted before passing to server. Here the outer function is encrypted, and encrypted advertisement matched with ser. Here the encryption of the function is not as important as in the financial sector. Consumer's public key is used for encrypting to whole related data and ads. Then operations on the cloud can be made, and the consumer can decipher the adver-

tisement taken to see it. If cloud service providers do not want to defraud advertisers, the system will function properly.

3.2.3.1 Some Functions with FHE

Three special functions are mentioned in the sections above. These are the mean function, the standard deviation, and the logistic regression function. The main feature of these functions is that they contain no or a limited number of multiplication operations, so they are functions that SwHE can be used.

- Average of k terms $\{p_i\}$: as a pair $(\sum_{i=1,\ldots,k} p_i, k)$, then avarage $m = \frac{\sum_{i=1,\ldots,k} p_i}{k}$
- Standard deviation: $\sqrt{\frac{\sum_{i=1,\ldots,k} (p_i - m)^2}{k}}$
- Logistic Regression: $l_r = \sum_{1,\ldots,k} b_i l_{r_i}$, here b_i is regression coeffficient for the variable l_{r_i} , and the prediction function is $g(x) = \frac{e^x}{1+e^x}$

There are important things to consider when using SwHE on functions. First, parameters must be explicitly selected for all encryption systems. These particular choices are determined by how many multiplications will be made in the function. Parameter selections are essential for safety and efficiency because a very safe system may not be efficient, and a very efficient system may not be secure. A balance must be established between the two. In other words, as the function used in the system changes, the selected parameters should also change.

CHAPTER 4

FULLY HOMOMORPHIC ENCRYPTION OVER THE INTEGER

The chapter will explain the FHE over the integer scheme named DGHV [30] designed by Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan in detail. In addition, it will be mentioned how to turn a SwHE scheme in to a FHE scheme in general.

4.1 Approximate Greatest Common Divisor Problem

Given a set of integers that are near-multiples of a secret integer, then the problem is finding approximate greatest common divisor of this set, namely secret integer. AGCD was first studied by Howgrave-Graham [17]. Further studies about AGCD was provided by the van Dijk, Gentry, Halevi and Vaikuntanathan's homomorphic encryption scheme [30]. The problem in this scheme is that given polynomially many samples from $D_{\gamma,\rho}(p) = \{\text{choose } q \xleftarrow{\$} \mathbf{Z} \cap [0, 2^{\gamma}/p],\ r \xleftarrow{\$} \mathbf{Z} \cap (-2^{\rho}, 2^{\rho}) : \text{output } x = pq + r\}$ for a randomly chosen η-bit odd integer p then, output p.

4.2 Somewhat Homomorphic Encryption Scheme Over the Integer

The section explains the SwHE scheme [30]. It includes many special parameters:

First parameter is γ. It represents bit-length of the integers in public key. Second one is η which stands for the bit-length of the secret key (it is the hidden approximate-gcd

of all the public-key integers). Next parameter is ρ, the bit-length of the noise. Lastly number of integers in public key expressed as τ.

The scheme has two different variants namely symmetric and asymmetric. Although we explain both scheme, generally we focus on public key version.

4.2.1 Symmetric Key Version of the Scheme

Here three algorithms of the symmetric key scheme, namely key generation, encryption, and decryption are explained. Moreover, we emphasize decryption algorithm to be sure decryption works correctly, we put some restrictions on parameters.

4.2.1.1 Key Generation

p, private key, is an odd integer in some interval $p \in [2^{\eta-1}, 2^{\eta})$.

4.2.1.2 Encryption

One can use a system $Enc(m) = pq + 2r + m$ for encryption one bit. In this system p is secret key that is greater than q. Also q and r are chosen as random integers. Note that when choosing r, be careful that "2r" must be less than half of the p in absolute value.

4.2.1.3 Decryption

One can decrypt ciphertext using the algorithm $Dec(c) = (c \ mod \ p) \ mod \ 2$. In the encryption phase, it noted that "$2r$" must be chosen less than half of the p in absolute value. The reason for this restriction is to ensure that the decryption is done correctly. In the decryption phase one, first calculate modular p. If $2r + m$ is greater than p, the decryption process is failed.

4.2.2 Asymmetric Key version of the Scheme

Turning symmetric key algorithm into public key algorithm is a simple process. Describe a set of integer such that these integers are generated by "encryptions of zero". Then choose a subset of this set and use it for public keys.

4.2.2.1 Key Generation

The private key is picked as an odd η-bit integer: p $\leftarrow (2\mathbf{Z}+1)\bigcap[2^{\eta-1}, 2^{\eta})$. To set public key, sample $D_{\gamma,\rho}(p) = \{$choose $q \xleftarrow{\$} \mathbf{Z} \cap [0, 2^{\gamma}/p]$, $r \xleftarrow{\$} \mathbf{Z} \cap (-2^{\rho}, 2^{\rho})$: output $x = pq + r\}$. Among them, x_0 should be assigned as the largest one. If it is not the largest, then relabel the x_i values. Restart the process until the remainder of x_0, denote as $r_p(x_0)$, is even and x_0 is odd. Then public key is a tuple of x_i's namely $pk = \langle x_0, x_1, ..., x_\tau \rangle$.

4.2.2.2 Encryption

S is a randomly chosen subset such that $S \subseteq \{1, 2, ..., \tau\}$, and integer $r \in (-2^{\rho'}, 2^{\rho'})$, then output $c \rightarrow [m + 2r + \sum_{i \in S} x_i]_{x_o}$

4.2.2.3 Evaluate

C_ϵ is a binary circuit. This circuit has t inputs and t ciphertexts c_i. C_ϵ performs whole operations like addition and multiplication over the integer and outputs the result as integer.

4.2.2.4 Decryption

Decryption process is similar with symmetric key scheme. It outputs m using the Dec function $Dec(c) = (c \ mod \ p) \ mod \ 2$.

4.2.3 Homomorphic operations

Both public key and symmetric key scheme have additive and multiplicative homomorphism. The homomorphic property defines for both version is same.

4.2.3.1 Additive homomorphism

Let c_1 and c_2 be two ciphertext .

$$\begin{aligned} c_1 &= q_1 p + 2r_1 + m_1 \\ c_2 &= q_2 p + 2r_2 + m_2 \\ c_1 + c_2 &= (q_1 + q_2).p + 2(r_1 + r_2) + (m_1 + m_2) \end{aligned}$$

Note that $c_1 + c_2$ is an encryption of $m_1 + m_2$, and it is equal to $m_1 \oplus m_2$ in mod 2.

Correctness of homomorphic addition:

When we try to decrypt ciphertext $c_1 + c_2$ we must obtain message $m_1 + m_2$ if the scheme additive homomorphic.

$$\begin{aligned} (c_1 + c_2 \ mod \ p)_2 &= (((q_1 + q_2).p + 2(r_1 + r_2) + (m_1 + m_2)) \ mod \ p)_2 \\ &= (2(r_1 + r_2) + (m_1 + m_2))_2 \\ &= m_1 + m_2 \end{aligned}$$

Note that additive homomorphic decryption works only the noise parameter $2(r_1 + r_2) < p$.

4.2.3.2 Multiplicative homomorphism

Let c_1 and c_2 be two ciphertext. If one take $r' = 2r_1r_2 + r_1m_2 + r_2m_1$ and $q' = q_1q_2p + 2q_1pr_2 + q_1pm_2 + 2r_1q_2p + m_1q_2p$ then, c_1c_2 can be expressed as following.

$$\begin{aligned} c_1 &= q_1 p + 2r_1 + m_1 \\ c_2 &= q_2 p + 2r_2 + m_2 \\ c_1.c_2 &= (q_1q_2p + 2q_1pr_2 + q_1pm_2 + 2r_1q_2p + m_1q_2p).p + 2(2r_1r_2 + r_1m_2 + r_2m_1) + (m_1.m_2) \\ c_1.c_2 &= q'p + 2r' + m_1m_2 \end{aligned}$$

Here, c_1c_2 is an encryption of m_1m_2. And note that noise becomes twice larger.

Correctness of homomorphic multiplication:

When we try to decrypt ciphertext $c_1.c_2$ we must obtain message $m_1.m_2$ if the scheme multiplicative homomorphic.

$$\begin{aligned}(c_1.c_2 \ mod \ p) \ mod \ 2 &= ((q'p + 2r' + m_1m_2) \ mod \ p) \ mod \ 2 \\ &= (2r' + m_1.m_2) \ mod \ 2 \\ &= m_1.m_2\end{aligned}$$

Note that multiplicative homomorphic decryption works only the noise parameter $2r' < p$.

4.3 General review of FHE scheme

In a FHE scheme, one can do limitless operations on encrypted data. Binary circuits are used for representing these operations (mainly AND, XOR, NAND gates). The inputs of the circuit are ciphertexts $\psi_1, \psi_2, ..., \psi_t$. Unlike the three traditional public-key algorithm elements, namely key generation, encryption, and decryption, a homomorphic scheme also needs an extra (possibly randomized) Evaluate algorithm. The inputs of Evaluate algorithm are public key, permitted circuit, and ciphertext tuples $\Psi = \langle \psi_1, \psi_2, ..., \psi_t \rangle$. These tuples of ciphertexts are used for input of the arithmetic circuit. The output of the Evaluate algorithm is a ciphertext ψ so that ψ is the ciphertext of $C(m_1, ..., m_t)$ under public key, where m_i is the message bit matching to the ciphertext of ψ_i

4.3.1 How to Construct a FHE Scheme

The primary problem operation on ciphertext is enormous noise. Every homomorphic addition and multiplication operations raise the noise significantly. The SwHE scheme supports a limited number of operations on encrypted text. The beginning step for constructing a FHE scheme is constructing a SwHE scheme. To illustrate, if the noise bit size is k let noise threshold as kt, in that situation it takes $log_2 t$ levels of

multiplication to caught the bound.

One can see that to construct a scheme letting unlimited operation on ciphertext, and we need a solution to minimize noise. Gentry suggests a procedure called "ciphertext refresh" in his Ph.D. thesis [12]. In this procedure, the decryption circuit has the input bits of secret key and ciphertext. Each of these bits is encrypted with a different public key. If the decryption circuit can handle the noise, the output of the same plaintexts is different than each other. When the decryption polynomial's degree is small enough, the noise of new ciphertexts will be enough to do the homomorphic operation again.

The augmented decryption circuit is a circuit augmented by logic gates. Generally, these gates are chosen as AND and XOR because these gates have functional completeness property [11]. This means it is possible to implement all logic gates with AND and XOR gates. To summarize, if the SwHE scheme can do arbitrary operations on encrypted plaintext, this means that the SwHE scheme can handle augmented decryption circuits.

The second question is, while SwHE can overcome AND and XOR gates, the decryption circuit can do the same? The answer is NO. To make it possible, one can use a step named bootstrapping. One must first use the "squashing method" to make the decryption circuit bootstarappable. This means that the decryption polynomial is small enough, and our SWHE can do many arbitrary operations on the ciphertext. Eventually, constructing a FHE scheme is completed successfully.

4.3.2 A General Explication

Now, in figure 4.1 one can see a general explanation of constructing a FHE scheme.

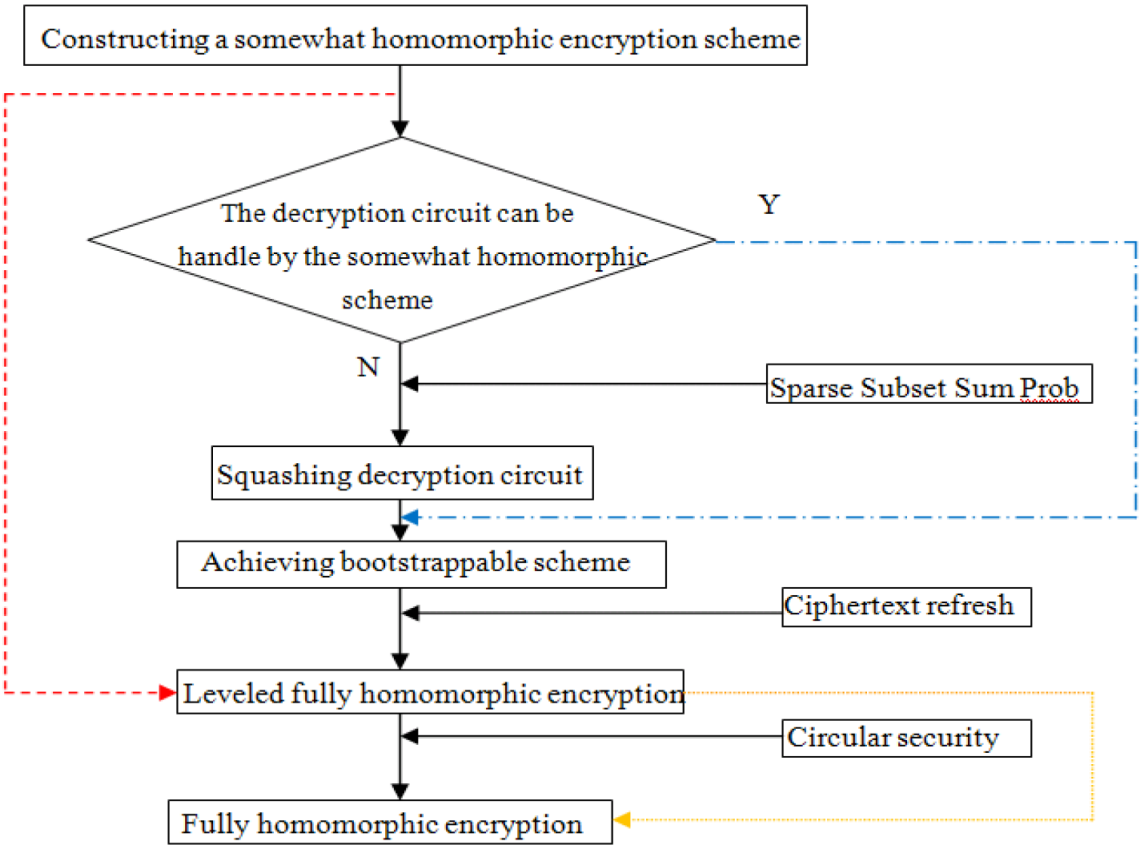

Figure 4.1: General Overview of Constructing FHE [31]

The explanation is formed as a module that was applied in Gentry's scheme. To generate a FHE scheme, one can follow all possible paths on the diagram. In the figure, while the solid lines show the Gentry scheme, dotted lines demonstrate alternative ways for constructing FHE. There are many challenges that will be explained in this figure.

- If decryption circuit can not be handled by SwHE then one must use a technique called "squashing". But this method include a very strong assumption namely "sparse subset sum problem" which is the most engrossing phase in Genry's solution.
- In a "leveled" FHE scheme, for refreshing the ciphertext, how many public keys to use depends on the depth of the circuit linearly unless one can choose to use same public key for all the levels. This is called "circular security".
- Gentry exhibited bootstrapping theorem to obtain FHE [12]. The theorem says that if the given a SwHE scheme can evaluate its own decryption circuit, then

the scheme can be converted it into a “leveled” FHE scheme. Operating the decryption function on ciphertext homomorphically using a encrypted secret key refreshes the ciphertext so decreases the noise.

4.4 Converting DGHV to FHE

The section includes the Gentry’s way [12] to construct a FHE scheme from SwHE scheme over the integer. As one remembers decryption function is represented as $Dec(c) = (c \, mod \, p)_2$, but it’s too hard to implement the modular system as a boolean circuit. Then one can use the trick $Dec(c) = [c - p\lfloor c/p \rfloor]_2$. Here, p is chosen as prime integer, so it is equal to 1 in modulo 2. However, the division is still a hard operation for the boolean circuit. Hence we use the "squash the decryption circuit." method suggested by Gentry [13]. Because of using the method, some additional information about the secret key are added to the public key. After the addition, new ciphertext is called "post-processed." One can decrypt these ciphertexts more accurately than the standard ones, making the scheme bootstrappable. However, this method has advantages as well as disadvantages. One of them is adding some information that can help the attacker break the scheme, and the other one is adding something that increases the size of the ciphertext.

4.4.1 Squashing Step

New parameters namely κ, θ and Θ were added. These are functions of λ (security parameter). To be more precise, we operate $\kappa = \gamma\eta/\rho$, θ=λ, and Θ= $\omega(\kappa log\lambda)$. Take secret key public key pk^* and $sk^* = p$ from the original SwHE schema ε^*, a set $y = \{y_1, ..., y_\theta\}$ of rational numbers in [0, 2) with κ bits of precision is added to public key. In this step we use the "Sparse Subset Sum Problem". There is a sparse subset $S \subset \{1, ..., \Theta\}$ of size Θ with $\sum_{i \in S} y_i \approx 1/p$ (mod 2). Finally, the secret key is changed with the indicator vector of the subset S.

4.4.1.1 Key Generation

Generate sk^* and pk^* as in the SwHE scheme, and set $x_p \leftarrow \lfloor 2^\kappa/p \rfloor$. A Θ-bit random vector $s = \{s_1, ..., s_\Theta\}$ is chosen such that its hamming weight is θ. Then, denote a set with $S = \{i : s_i = 1\}$. For $i = \{1, ..., \Theta\}$, random integers $u_i \in Z \cap [0, 2^{\kappa+1})$ are chosen, subject to the rule that $\sum_{i \in S} u_i = x_p \ (mod\ 2^{\kappa+1})$. Then, set $y_i = u_i/2^\kappa$ and $y = \{y_1, ..., y_\Theta\}$. Note that the each $y_i \in [0, 2)$, with κ bits of precision after the binary point. Also the chosen y_i's verifies that, $[\sum_{i \in S} y_i]_2 = 1/p - \triangle_p$ for some $|\triangle_p| < 2^{-\kappa}$. The outputs are secret key sk = s and public key $pk = (pk^*, y)$.

4.4.1.2 Encrypt and Evaluate

Ciphertext c^* is created first, and set $z_i \leftarrow [c^* y_i]_2$ for $i \in \{1, ..., \Theta\}$,also keep z_i as $n = [log\theta]$ + 3 bits of precision after the binary point.
The outputs are $c^*$and $z = \langle z_1, ..., z_\Theta \rangle$

4.4.1.3 Decryption

The output is $m' \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_2$

4.4.2 Bootstrapping

Bootstrapping is one of the important part of the FHE structure. As in the below theorem, thanks to bootstrapping part decryption circuit has been a subset of permitted circuits.

Theorem 4.4.1. *([13]) Assume that ε is the scheme above at the same time D_ε is the set of augmented (squashed) decryption circuits. Finally, let $D_\varepsilon \subset C(P_\varepsilon)$.*

Actually, this theorem says that ε is bootstrappable. The purpose is to state the modified decryption equation $m' \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_2$ as a permitted polynomial, and demonstrate that this polynomial is computed with a polynomial-size circuit. Remember that $z_i \in \mathbf{Q}$, and $0 \geq z_i > 2$, in binary representation with $n = \lceil log\theta + 3 \rceil$

bits of precision and also, $c^* \in \mathbf{Z}$. Moreover, s_i's can be chosen as 0 or 1, and weight of the vector $\{s_1, ..., s_\Theta\}$ is equal to θ. Note that our parameters $\sum s_i z_i$ is within 1/4 of an integer.

We separate operation phase into three parts:

1. Set $a_i \leftarrow s_i z_i$ for $i \in \{1, ..., \Theta\}$(i.e., if $s_i = 1$ then $s_i = a_i$ if not $a_i = 0$). One can see that because $z_i \in \mathbf{Q}$, and $0 \geq z_i > 2$ also $a_i \in \mathbf{Q}$, and $0 \geq z_i > 2$.

2. Depending the condition $\sum_i a_i = \sum_j w_j \ mod\ 2$, generate different $n + 1$ rational number $\{w_j\}_{j=1}^{n}$ with less than n bits of precision, from the set of $\{a_i\}_{i=1}^{\Theta}$.

3. Finally, the output is $c^* - \sum_j w_j \ mod\ 2$

For item one, one can easily express it using multiplication gates with one level sub-circuit. Nevertheless, it is too complicated to represent the other two items with one sub-circuit level. For the second part, one can use the three-for-two trick [20] with a constant-depth circuit. This method can be used to convert three numbers in to two. Converted two numbers can be maximum 1 bit longer. Here, note that to apply this trick, bit length of the inputs are not important. Eventually, one can say that summation of these three numbers is equal to two numbers. To obtain two integer s_1 and s_2 which is $[s_1 + s_2 = \sum_{i=1}^{k} r_i]$ this trick can be applied maximum of $\lfloor log_{3/2} k + 2 \rceil$ times and conclude that minimum depth is $d \leq 2^{\lceil log_{3/2} k + 2 \rceil} < 8k^{1/log(3/2)} < 8k^{1.71}$. Remember that our parameter $\sum s_1 + s_2$ is 1/4 of an integer and, our final sum is $\sum s_1 + s_2 \ mod\ 2$. This implies that one can compute this value using a 4-degree multivariate polynomial.Now, the total degree of a circuit that computes $[s_1 + s_2 = \sum_{i=1}^{k} r_i]$ can be maximum $4 \times 8k^{1.71} = 32k^{1.71}$.

CHAPTER 5

EFFICIENCY PROBLEM OF DGHV SCHEMA

In this chapter, one can find the main challenge about the FHE schema over Integers and current strategies to tackle them.

We illustrate the FHE schema over the ıntegers on previous chapters. When we look at the schema , it is include 3 phases.

1. Describe a SwHE schema
2. Squashing the schema
3. Bootstrapping

As one can see on the previous chapters these three stages involve repetitive and heavy processes that will reduce efficiency. We can group the studies carried out to increase efficiency as follows:

- finding more efficient SwHE schemes,
- extending the message size or message space,
- to speed up the bootstrapping process [8],
- eliminate the Bootstrapping procedure [4],
- eliminate the Squashing process [14],
- decrease the public key's size.

Now, we summarize the main studies about above headers, and we carry on with explaining the second item "extending the message size or message space" in detail.

5.1 Extending the message size or message space

The DGHV scheme suggested by Dijk, Gentry, Halevi and Vaikuntanathan [30] allows only 1 bit encryption for 1 encryption cycle and also use Z_2 for the message space. This means that one can encrypt only 0 or 1 for a encryption cycle.

There are mainly two approaches to extending the message spaces or message sizes over the DGHV scheme. First one is a process called Batching, i.e. a procedure that allows to process a input vector homomorphicly as a single output. The second one is using non-binary message spaces instead of Z_2.

5.1.1 Batch FHE Over the Integers

In the DGHV scheme, one can encrypt a message m using an algorithm.

$$c = q.p + 2r + m$$

p is the private key and q and r are random integers and q is much bigger than r. Following decryption algorithm can be used to recover m.

$$m = [c \; mod \; p] \; mod \; 2.$$

To turn DGHV scheme into a Batch scheme, one can use Chinese Remainder Theorem [7]. Thanks to this theorem m_i which is a message contain multiple bits can be encrypted as a single ciphertext. One can see the form of ciphertext which is batched with using CRT as following

$$c = CRT_{q_0,p_0,\dots,p_{l-1}}(q, 2r_0 + m_0,\dots, 2r_{l-1} + m_{l-1}).$$

In above ciphertext $q_0, p_0,\dots, p_{l-1}$ are all co-prime integers. To Decrypt m_i message bit vector we use the same algorithm $m_i = [c \; mod \; p_i] \; mod \; 2$ for all $0 \leq i \leq l-1$. Note that we define ciphertext c as

$$c = CRT_{q_0,p_0,\dots,p_{l-1}}(q, a_0,\dots, a_{l-1}).$$

Here, u is chosen under the restriction with $0 \leq u < q_0 \times \prod_{i=0}^{l-1} p_i$ such that for all $0 \leq i < l$

$$u \equiv q \ (mod \ q_0),$$
$$u \equiv a_i \ (mod \ p_i).$$

Also, this technique help us not only encrypt a bit but also elements form Z_Q. One can choose public elements as $Q_0, \ldots, Q_{l-1}$ and, encrypt $(m_0, \ldots, m_{l-1})$ chosen from $Z_{Q_0} \times, \cdots, Z_{Q_{l-1}}$ respectively. Now packing l plaintext into a single ciphertext looks like

$$c = CRT_{q_0, p_0, \ldots, p_{l-1}}(q, Q_0 r_0 + m_0, \ldots, Q_{l-1} r_{l-1} + m_{l-1}).$$

We choose q as random integer modulo q_0 and r_i's small integers. As we indicate earlier, for decryption phase use same algorithm to find messages.

$$m_i = [[c \ mod \ p_i] \ mod \ Q_i]$$

One can pick Q_i's as pairwise co-prime and, ın this case isomorphism theorem may be used to view batch scheme as a SwHE scheme on Z_Q. To more specific, according to isomorphism theorem i.e. $Z_{\prod Q_i} \cong \prod Z_{Q_i}$, homomorhic encryption supporting arithmetic operation on Z_Q. We can think that Q is equal to production of all Q_i's. Turning this into a public key scheme is so similar with DGHV's phase.

$$c = \left[\sum_{i=0}^{l-1} m_i . x_i^{'} + \sum_{i \in S} x_i \right]_{x_0}$$

We choose two integers $x_i^{'}$ and x_i in addition to public keys. Note that $x_i^{'} \ mod \ p_j = Q_j r_{i,j}^{'} + \delta_{i,j}$ and $x_i \ mod \ p_j = Q_j r_{i,j}^{'}$ for all i, j. A FHE scheme can be obtained from given scheme using Gentry's technique if one choose all Q_i's are same and equal to 2 for $0 < i < l - 1$. To conclude message space Z_2 is extended as $(Z_2)^k$.

5.1.1.1 Key Generation

Generate η-bit p_j's, which are distinct primes, for $j \in [0, l)$ and compute π by multiplying p_j's. Then, define the public key parameter $x_0 = q_0 . \pi$ where q_0 is a 2^{λ^2}-rough integer chosen the set $q_0 \leftarrow \mathbf{Z} \cap [0, 2^{\gamma}/\pi]$. Note that 2^{λ^2}-rough means there is no prime factors of q_0 smaller than 2^{λ^2}. Be sure that $gcd(Q_j, x_0) = 1$ for $0 \leq j < l$.

Choose the integers x_i and x'_i with a quotient by π uniformly and independently distributed in $\mathbf{Z} \cap [0, q_0)$, and with the specified distribution modulo p_j for $j \in [0, l)$.

$i \in [1, \tau]$, $\quad x_i \; mod \; p_j \; = Q_j r_{i,j}$, $\quad r_{i,j} \leftarrow \mathbf{Z} \cap (-2^\rho, 2^\rho)$,

$i \in [1, \ell - 1]$, $\quad x'_i \; mod \; p_j \; = Q_j r'_{i,j} + \delta_{i,j}$, $\quad r'_{i,j} \leftarrow \mathbf{Z} \cap (-2^\rho, 2^\rho)$,

Finally $sk = (p_j)_{\{0 \le i \le l-1\}}$ and $pk = \{x_0, (x'_i)_{\{0 \le i \le l-1\}}, (Q_i)_{\{0 \le i \le l-1\}}, (x_i)_{\{0 \le i \le \tau\}}\}$

5.1.2 Encryption

For any $m = (m_0, \cdots, m_{l-1})$ with $m_i \in \mathbf{Z}_{Q_i}$, choose a random binary vector $b = (b_i)_{\{1 \le i \le \tau\}} \in \{0, 1\}^\tau$. Then, ciphertext can be found by using the folowing encryption algorithm.

$$c = \left[\sum_{i=0}^{l-1} m_i . x'_i + \sum_{i=1}^{\tau} b_i x_i \right]_{x_0}$$

5.1.3 Decryption

Output $m = (m_0, ..., m_{l-1})$ where $m_j \leftarrow [c \; mod \; p_j]_{Q_j}$.

5.1.4 Addition and Multiplication

$Add(pk, c_1, c_2)$: output $c_1 + c_2 \; mod \; x_0$.

$Mult(pk, c_1, c_2)$: output $c_1 * c_2 \; mod \; x_0$.

5.1.5 Making The Scheme Fully Homomorphic

Making a scheme Fully Homomorphic, Gentry's technique [30] is used with batch settings. The purpose of manipulating the decryption circuit is to obtain a polynomial with smaller degree.

Include a set $\vec{y} = \{y_1, \cdots, y_\Theta\}$ to the public key. Here $y_i \in \mathbf{Q}$, and $0 \le y_i < 2$ with κ bits of precision. The purpose is to find a sparse subset $S_j \subset [0, \Theta - 1]$ of size Θ

such that $\sum_{\{i \in S_j\}} y_i \approx 1/p_j \ (mod\ 2)$.

5.1.5.1 Key Generation

Addition of private key and public key set $x_{p_j} \leftarrow \lfloor 2^\kappa / p_j \rceil$. Pick at random a Θ-bit vector $\vec{s_j} = \langle s_{j,0}, \cdots, s_{j,\Theta-1} \rangle$ with hamming weight θ for $0 \leq j < l$. Choose a $u_i \in [0, 2^{\kappa+1})$, where $i \in [0, \Theta - 1]$ dependent to the condition that $\sum_{i=0}^{\Theta-1} u_i s_{j,i} = x_p \ (mod\ 2^{\kappa+1})$. Define $\vec{y_i} = \{y_0, \cdots, y_{\Theta-1}\}$ and $y_i = u_i / 2^\kappa$. Thus each y_i verifies that for some $|\epsilon_j| < 2^\kappa$,

$$1/p_j = \sum_{i=0}^{\Theta-1} s_{j,i}.y_i + \epsilon_j \ mod\ 2.$$

Finally output $sk = (s_0, ..., s_{l-1})$ and $pk = (pk^*,\ y_0, ..., y_{\Theta-1})$.

5.1.5.2 Expanding

Take ciphertext c^* generated in section 5.1.2. Then create $z_i \leftarrow [c^* \ .\ y_i]_2$ for $i \in [0, \Theta - 1]$, with $n = \lceil log_2(\theta + 1) \rceil$ bits of precision after the binary point for each z_i. Then, output the ciphertext c^* and $z = (z_i)_{\{i=0,\cdots,\Theta-1\}}$.

5.1.5.3 Decryption

Lastly output the message $m = (m_0, \cdots, m_{l-1})$ with

$$m_j \leftarrow \left[\lfloor \sum_{i=0}^{\Theta-1} s_{j,i}.z_i \rceil \right]_2 \ \oplus \ (c \ mod\ 2).$$

5.1.6 FHE Over the Integers for Non-Binary Message Spaces

As we know about previous chapters DGHV scheme has message space Z_2 for $m \in Z_2$ and ciphertext $c = pq + 2r + m$. In algorithm p is a private prime integer and r is a small noise. To find message m we use an easy decryption algorithm $m = (c \ mod\ p) \ mod\ 2 = c - p\ .\ \lfloor c/p \rceil \ mod\ 2$. In squashed scheme, we find message m

using the following algorithm.

$$m \leftarrow (c \ mod \ 2) \oplus \left(\left\lfloor \sum_{i=1}^{\Theta} s_i z_i \right\rceil \ mod \ 2 \right)$$

In this algorithm $(s_1, \cdots, s_\Theta) \in \{0,1\}^\Theta$ is private key, and $z_i = (z_{i,0}.z_{i,1}, \cdots, z_{i,L})_2 \in \mathbf{R}$, the equation

$$\sum_{i=1}^{\Theta} s_i z_i \approx c/p$$

As Gentry mentioned in his article [12], there are 2 circuits.

1. First circuits for $j \in [0, L]$ calculates $\mathrm{W}_j = \sum_{i=1}^{\Theta} s_i z_{i,j}$ as following.

$$\sum_{i=1}^{\Theta} s_i z_i = W_0 + 2^{-1} W_1 + \cdots + 2^{-L} W_L$$

2. Second one calculates a and b using 3-to-2 trick repetitively.

$$W_0 + 2^{-1} W_1 + \cdots + 2^{-L} W_L = a + b \ mod \ 2$$

   To calculate W_j one needs to use half adders for sum and carry. Polynomials are used for such calculations. Although finding polynomial is easy for sum part , it is too hard for carry part in non binary message spaces. This is the main explanation why DGHV scheme use Z_2 message space for turning into a FHE scheme. Because all W_j is smaller than securiy parameter λ, one can easily say first circuits multiplicative degree is at most λ. The second circuits has a multiplicative degree $\mathcal{O}((log\lambda)^2)$ which is exact degree of it.

5.1.6.1 Q-ary Half Adder

As mentioned above chapters, there is a way to extend message space Z_2 to Z_Q where Q is a prime. In article [25] the Batch DGHV for non-binary message space scheme includes a part of finding function that express carries.

While c is carry and s is sum for $x, y \in \ Z_Q$, then

$$x + y = (c, s)_Q = c \ \times \ Q + s.$$

One can observe that $s = x + y \ mod \ Q$, and $f_{carry,Q}(x, y)$ denoted as lowest degree polynomial for carry c.

Theorem 5.1.1. *$f_{carry,Q}(x,y)$ is a polynomial that defined over Z_Q as following.*

$$f_{carry,Q}(x,y) := \sum_{i=1}^{Q-1} \binom{x}{i}_Q \binom{y}{Q-i}_Q$$

This function has total degree Q. Then one can say that for $x, y \in Z_Q$

$$c = f_{carry,Q}(x,y) \ mod \ Q$$

Theorem 5.1.2. *The polynomial $f_{carry,Q}(x,y)$ has the lowest degree among all functions h(x,y) over Z_Q satisfying that the equation*

$$c = h(x,y) \ mod \ Q$$

5.1.6.2 Low-Degree Circuits for Sum of Integers

Define $a_i = (a_{i,1}, \cdots, a_{i,n})_Q$ for $i = 1, \cdots, m$, and a circuit which is computes

$$a_1 + \cdots + a_m \ mod \ Q^n.$$

$A = a_{(i,j)_{i,j}}$ is used as a matrix representation of $a_i = (a_1, \cdots, a_m)_Q$, so one can say that matrix A is an $m \times n$ matrix. Moreover, $StreamAdd_Q(x_1, \cdots, x_m)$ for $x_1, \cdots, x_m \in Z_Q$ is an algorithm that consists of the following steps.

Algorithm 1 $StreamAdd_Q(x_1, \cdots, x_m)$

$s_2 \leftarrow x_1 + x_2$ mod Q
$c_2 \leftarrow f_{carry,Q}(x_1, x_2)$mod Q {**note:**$(c_2, s_2)_Q = x_1 + x_2$}
for $i = 3, \cdots, m$ **do**
  $s_i \leftarrow s_{i-1} + x_i$modQ
  $c_i \leftarrow f_{carry,Q}(s_{i-1}, x_i)$ mod Q {**note:**$(c_i, s_i)_Q = s_{i-1}1 + x_i$}
end for
return $(s_m, (c_2, \cdots, c_m))$

As one can see that for $(s_m, (c_2, \cdots, c_m)) \leftarrow StreamAdd(x_1, \cdots, x_m)$, and

$$x_1 + \cdots + x_m = s_m + Q \times (c_2, \cdots, c_m).$$

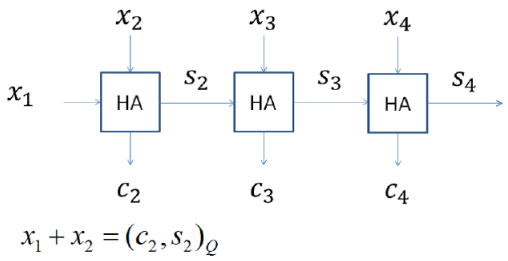

Figure 5.1: $StreamAdd_Q(x_1, \cdots, x_m)$

In above ,an algorithm StreamAdd which can compute sum of integers is defined. Next, MatrixAdd algporithm is shown for $m \times n$ matrix $A = (a_{i,j})_{i,j}$.

Algorithm 1 $MatrixAdd_Q(A)$

for $j = 1, \cdots, n$ **do**

$(\alpha_j, (\beta_{2,j}, \cdots, \beta_{m,j})) \leftarrow StreamAdd_Q(a_{1,j}, \cdots, a_{m,j})$

{**note:**Apply $StreamAdd_Q$ to the jth column of A}

end for

for $i = 1, \cdots, n-1$ **do**

$(b_{1,j}, \cdots, b_{m,j}) \leftarrow (\alpha_j, \beta_{2,j+1}, \cdots, \beta_{m,j+1)}$

{**note:**Shift $\beta_{2,j+1}, \cdots, \beta_{m,j+1)^T}$ to the left}

end for

return $B = (b_{i,j} and \alpha_n$, where B is an $m \times (n-1)$ matrix

To more specify,

$$B = \begin{pmatrix} \cdots, & \alpha_{n-1} \\ \cdots, & \beta_{2,n} \\ . & . \\ . & . \\ \cdots, & \beta_{m,n} \end{pmatrix}, \alpha_n \leftarrow (StreamAdd_Q) \leftarrow \begin{pmatrix} \cdots, & a_{1,n} \\ . & . \\ . & . \\ . & . \\ \cdots, & a_{m,n} \end{pmatrix} = B$$

A matrix B shown as $(B = (b_{i,j}, \alpha_n) \leftarrow MatrixAdd_Q(A)$ for $i = 1, \cdots, m$ and $b_i = (b_{i-1}, \cdots, b_{i,n-1}, 0)_Q$, and also one can realized that

$$a_1 + \cdots + a_m \equiv (b_1 + \cdots + b_m) + \alpha_n (mod Q^n)$$

As a last part, an algorithm $FinalAdd_Q(A)$ is defined following.

Algorithm 1 $MatrixAdd_Q(A)$

for $j = 1, \cdots, n$ **do**

$(\alpha_j, (\beta_{2,j}, \cdots, \beta_{m,j})) \leftarrow StreamAdd_Q(a_{1,j}, \cdots, a_{m,j})$

{**note:**Apply $StreamAdd_Q$ to the jth column of A}

end for

for $i = 1, \cdots, n-1$ **do**

$(b_{1,j}, \cdots, b_{m,j}) \leftarrow (\alpha_j, \beta_{2,j+1}, \cdots, \beta_{m,j+1})$

{**note:**Shift $\beta_{2,j+1}, \cdots, \beta_{m,j+1)^T}$ to the left}

end for

return $B = (b_{i,j} and \alpha_n$, where B is an $m \times (n-1)$ matrix

Then, A is the matrix which is representation of sequence $(a_1, a_2, \cdots, a_m)$ and also $(d_1, d_2, \cdots, d_n) \leftarrow MatrixAdd_Q(A)$. One can see that below theorems hold due to $deg f_{carry,Q} = Q$.

Theorem 5.1.3. $(d_1, \cdots, d_n)_Q = a_1 + a_2 + \cdots + a_m \ mod \ Q^n$.

Theorem 5.1.4. *There is a polynomial $f_{Q,i}(x_{1,1}, \cdots, x_{m,n})$ defined in Z_Q field such that it satisfies two items.*

- $deg \ f_{Q,i} = Q^{n-i}$
- $d_i = f_{Q,i}(a_{1,1}, \cdots, a_{m,n}) \ mod \ Q$

5.1.7 Batch SwHE Over the Integers for Non-Binary Message Spaces

In previous section, fully homomorphic encrytion scheme for non-binary message space is demonstrated. In this one, Batch SwHE scheme over non binary message space $M = Z_{Q_1}^{h_1} \times \cdots \times Z_{Q_k}^{h_k}$ is explained. Here, $k \geq 1$, $h_j \geq 1$ and $Q_1, \cdots, Q_k$ are distinct primes. As demonstrate this with notations

$$I := \{(i,j) \mid i,j \in Z, 1 \leq i \leq k, \ 1 \leq j \leq h_i\}.$$

5.1.7.1 Key Generation

Select $p_{i,j}$ and $Q_{i'}$ to be distinct from each other such that $p_{i,j}$ for $(i,j) \in I$ is chosen random and uniformly, and q_o is co-prime to all $p_{i,j}$ and $Q_{i'}$.

$$q_0 \leftarrow \left[1, 2^{\gamma} / \prod_{(i,j)\in I} p_{i,j} \cap ROUGH(2^{\lambda^2}) \right]$$

In the upper equation, $ROUGH(2^{\lambda^2})$ represents a set does not have integers which have no prime factors less than 2^{λ^2}.

$$N := q_0 \prod_{(i,j)\in I} p_{i,j}$$

To continue, more parameters must be chosen. Select two parameters namely $e_{\varepsilon;0}$ and $e_{\varepsilon;i,j}$ for $\varepsilon \in \{1, \cdots, \tau\}$ and $(i,j) \in I$ by

$$e_{\varepsilon;0} \leftarrow [0, q_0) \cap \mathbf{Z}, e_{\varepsilon;i,j} \leftarrow (-2^{\rho}, 2^{\rho}) \cap \mathbf{Z}.$$

x_{ε} is integer that chosen from $(-N/2, N/2]$ such that

$$x_{\varepsilon} \equiv e_{\varepsilon;0} \ (mod q_0), \ x_{\varepsilon} \equiv e_{\varepsilon;i,j} \ Q_i \ (mod \ p_{i,j}) \ for \ (i,j) \in I.$$

Likewise, for $(i,j), (i',j') \in I$, select $e'_{i,j;0}$ and $e_{i,j;i',j'}$ by

$$e'_{i,j;0} \leftarrow [0, q_0) \cap Z, e_{i,j;i',j'} \leftarrow (-2^{\rho}, 2^{\rho}) \cap \mathbf{Z}.$$

$x'_{i,j}$ is integer that chosen from $(-N/2, N/2]$ such that

$$x'_{i,j} \equiv e'_{i,j;0} \ (mod q_0),$$

$$x'_{i,j} \equiv e'_{i,j;i',j'} \ Q_{i'} + \delta_{(i,j),(i',j')} \ \ (mod \ p_{i',j'}) \ for \ (i',j') \in I.$$

Here $\delta_{(i,j),(i',j')}$ is kronecker delta where for $i = i'$ and $j = j'$ its value is equal to 1, for the other cases its value is equal to 0. Lastly, publish public keys set contains $N, x_{\varepsilon}, x'_{i,j}$ and output the private key sk existing every $p_{i,j}$.

5.1.7.2 Encryption

There is a message $\vec{m} = (m_{i,j})_{(i,j)\in I} \in M$, to define encrypted message use below algorithm $Enc(pk, \vec{m})$. In this algorithm T denotes random subset of $\{1, 2, \cdots, \tau\}$ as

$$c := \sum_{(i,j)\in I} m_{i,j} x'_{i,j} + \sum_{\varepsilon \in T} x_{\varepsilon} \ Mod \ N \in (-N/2, N/2] \cap \mathbf{Z}.$$

5.1.7.3 Decryption

There is a ciphertext c, to output the message $\vec{m} \in M$ use below algorithm $Dec(sk, c)$.

$$\vec{m} := ((c \ mod \ p_{i,j})_{Q_i})_{(i,j)\in I}$$

5.1.7.4 Evaluation

Function $Eval(pk, f, c_1, \cdots, c_n)$, has an inputs public key pk,a polynomial f includes integer coefficient and ciphertext $c_1, \cdots, c_n$, seen below

$$c^* := f(c_1, \cdots, c_n) \ Mod \ N.$$

5.2 Batch FHE scheme: Bootstrapping for Large Plaintext

In this section bootstrapping procedure is explained for Batch SwHE scheme in 5.1.3.

5.2.1 Squashed Scheme

In this step, the primary purpose is to decrease the multiplicative degree of the decryption circuit for performing the bootstrapping phase correctly. As in the previous schemes, one can apply the same steps with DGHV scheme [30] to achieve the squashed method. The next session will explain how extra parameters κ_i, θ_i, Θ_i and L_i are chosen.

5.2.1.1 Key Generation

In this step, private and secret keys are produced by $KeyGen$ algorithm. Choose a subset Π of $S_{h_1} \times ... \times S_{h_k}$ that includes an id permutation, generating the group itself. Then, choose a Θ_i-bit vector which is chosen randomly with hamming weight θ_i, and for each $(i, j) \in I$ by $(s_{i,j;1}, ..., s_{i,j;\Theta_i}) \in \{0, 1\}^{\Theta_i}$. After that, define the $X_{i,j} := \lfloor {Q_i}^{\kappa_i}.(p_{i,j} \ Mod \ Q_i)/p_{i,j} \rceil$. To define a new parameter $u_{i,j}$, choose $i \in [1, k]$

and $1 \leq l \leq \Theta_i$, and it is expressed as $u_{i,l} \leftarrow [0, Q_i^{\kappa_i+1}) \cap \mathbf{Z}$ in such a way that for $1 \leq j \leq h_i$,

$$\sum_{l=1}^{\Theta_i} s_{i,j;l} u_{i,l} \equiv X_{i,j} \ \ (mod\ Q_i^{\kappa_i+1})$$

While $\sigma = (\sigma_1, ..., \sigma_k) \in \Pi$, produce

$$v_l^{\sigma} \leftarrow Enc(pk, \vec{m}_l{}^{\sigma})\ for\ 1 \leq l \leq \Theta_{max},$$

here the message $\vec{m}_l{}^{\sigma} = (m_{l;i,j}^{\sigma})_{(i,j)\in I} \in M$ is described by

if $l \leq \Theta_i\ m_{l;i,j}^{\sigma} = s_{i,\sigma_i(j);l}$, otherwise $m_{l;i,j}^{\sigma} = 0$.

Finally, demonstrate public key pk* including $pk, \Pi, u_{i,l}, v_l^{\sigma}$ and output the private key sk* formed with each $s_{i,j;l}$.

5.2.1.2 Encrytion and Evaluation

$Enc^*(pk^*, \vec{m})$ and $Eval^*(pk^*, f, c_1, ...c_n)$ functions are same in the Batch SwHE scheme.

- $Encryption$: There is a plaintext $\vec{m} = (m_{i,j})_{(i,j)\in I} \in M$, to define ciphertext c use below algorithm $Enc(pk, \vec{m})$. In this algorithm T denotes random subset of $\{1, 2, \cdots, \tau\}$.

$$c := \sum_{(i,j)\in I} m_{i,j} x'_{i,j} + \sum_{\varepsilon \in T} x_{\varepsilon}\ Mod\ N \in (-N/2, N/2] \cap Z$$

- $Decryption$: Function $Eval(pk, f, c_1, \cdots, c_n)$, has an inputs public key pk,a polynomial f includes integer coefficient and ciphertext $c_1, \cdots, c_n$, seen as following.

$$c^* := f(c_1, \cdots, c_n)\ Mod\ N$$

5.2.1.3 Decryption

To find the message bits first compute $z_{i,l}$ for $1 \leq i \leq k$ and $1 \leq l \leq \Theta_i$ as following.

$$z_{i,l} := ((c\ .\ u_{i,l}/Q_i^{\kappa_i}\ mod\ Q_i)\ mod\ L_i)$$

Hence, output the message bits $m_{i,j}$ for $(i,j) \in I$

$$m_{i,j} := c - \left\lfloor \sum_{l=1}^{\Theta_i} s_{i,j;l} z_{i,l} \right\rceil \ mod\ Q_i$$

Observe that, the main difference between Batch SwHE scheme and Batch FHE is due to the $u_{i,l}$ parameter included in the new public key. Also these parameter subjects to parameter $s_{i,j;l}$.

CHAPTER 6

TIME COMPLEXITIES OF ALGORITHMS

In this chapter, one can find the calculation of time complexities of two algorithms, namely the DGHV scheme [30] and Batch DGHV scheme [7], in detail.

Some of the parameters used during the calculations and their explanations are given below and also, all parameters are expressed in terms of the security parameter λ. First parameter is γ. It represents bit-length of the integers in public key. Second one is η which stands for the bit-length of the secret key (it is the hidden approximate-gcd of all the public-key integers). Next parameter is ρ, the bit-length of the noise. Lastly number of integers in public key expressed as τ. These parameters should be chosen under the following limitations:

$\rho = \omega(log\ \lambda)$, in order to thwart brute-force attacks on the noise;

$\eta \geq \rho.\Theta(\lambda log^2\ \lambda)$, to compute the "squashed decryption circuit" to provide homomorphism for deep enough circuits.

$\gamma = \omega(\eta^2 log\lambda)$, to protect against many types of lattice-based attacks on the underlying AGCD problem.

$\tau \geq \lambda + \omega(log\ \lambda)$, to apply the leftover hash lemma in the reduction to AGCD.

Moreover, a appropriate element set to remember during computations is $\rho = \lambda$, $\rho' = 2\lambda$, $\eta = \tilde{\Theta}(\lambda^2)$, $\gamma = \tilde{\Theta}(\lambda^5)$ and $\tau = \gamma + \lambda$.

Note that to create public keys following distribution is used as following.

$D_{\gamma,\rho}(p) = \{\text{choose } q \leftarrow \mathbf{Z} \cap [0, 2^\gamma/p], r \leftarrow \mathbf{Z} \cap (-2^\rho, 2^\rho) : \text{output } x = pq + r\}$

One can easily see that the distribution is efficiently sampleable.

6.1 Time complexity of DGHV scheme

This section includes the time complexity calculations of all parts of the fully homomorphic DGHV schemes, namely Key Generation, Encryption, Squashing respectively.

6.1.1 Key Generation

The private key p is chosen randomly in $p \leftarrow (2\mathbf{Z}+1) \cap [2^{\eta-1}, 2^{\eta}]$. For the public key, sample $x_i \leftarrow D_{\gamma,\rho}(p)$ for $i = 1, \cdots, \tau$.

$$x_i = p * q_i + r_i$$

First, we compute complexities of τ multiplications. We multiply η- bit secret key with $(\gamma - \eta)$-bit q_i's. The cost can be expressed as $\mathcal{O}(\eta\tau(\gamma - \eta))$. Then, we compute complexities of τ addition between $(p * q_i)$'s and noise parameter r_i. Note that, after the multiplication we can set length of $(p * q_i)$ as γ-bit. The cost of τ addition can be expressed as $\mathcal{O}(\tau\gamma)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{12})$.

6.1.2 Encryption

Choose a bit $m \in \{0, 1\}$ and a random subset S such that $S \subseteq \{1, 2, ..., \tau\}$ and a random integer $r \in (-2^{\rho'}, 2^{\rho'})$, then output the ciphertext c as

$$c \rightarrow [m + 2r + \sum_{i \in S} x_i]_{x_0}.$$

In this step we compute complexities of τ additions. We can take size of the set S as τ because we thought the worst case scenarios. We add γ-bits x_i's with each others. Note that, while first, we add two γ-bits x_i, for the second addition we add γ-bit and $(\gamma + 1)$-bit, and we add γ-bit and $(\gamma + \tau - 1)$-bit for the τ th one. Then, cost can be expressed as $\tilde{\mathcal{O}}(\gamma\tau)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{10})$. Note that, after the reduction ciphertext is smaller than x_0 .

6.1.3 Squashing

In this step, extra tree parameters κ, θ, Θ are added to scheme. Note that all of them are functions of λ. Here parameters used as $\kappa = \gamma\eta/\rho'$, $\theta = \lambda$, and $\Theta = \omega(\kappa.log\lambda)$. To use the size-reduction optimization, it is sufficient to use $\kappa = \gamma + 2$. Thanks to this choice, Θ is much more smaller. A private key $sk^* = p$ and public key pk^* are taken from the original scheme described in section 6.1.1. Then a set $\vec{y} = \{y_1, \cdots, y_\Theta\}$ is added to the public key. Here $y_i \in \mathbf{Q}$, and $0 \leq y_i < 2$ with κ bits of precision. The purpose is to find a sparse subset $S_j \subset [0, \Theta - 1]$ of size Θ such that $\sum_{\{i \in S_j\}} y_i \approx 1/p_j \ (mod\ 2)$.

6.1.3.1 Key Generation

Along with the private key $sk^* = p$ and public key pk^*, an adition item x_p should be set as following.

$$x_p \leftarrow \lfloor 2^\kappa / p \rceil$$

To generate x_p we need just 1 division. As we mentioned in section 6.1.3, using $\kappa = \gamma + 2$ is sufficient. Then our complexity is $\mathcal{O}(\eta(\gamma+2))$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^7)$, and new length of x_p is equal to $(\gamma - \eta + 2)$. Then, a Θ-bit vector $\vec{s} = \langle s_1, \cdots, s_\Theta \rangle$ randomly chosen with hamming weight θ and a set denoted as $S = \{i : s_i = 1\}$. For $i = \{1, ..., \Theta\}$, random integers $u_i \in Z \cap [0, 2^{\kappa+1})$ are chosen, according to the following rule.

$$\sum_{i \in S} u_i = x_p \ (mod\ 2^{\kappa+1})$$

For satisfying the condition, we need to do Θ addition. We add $(\kappa + 1)$ bits u_i's with each others. Note that, while first we add two $(\kappa + 1)$-bits u_i, for the second addition we add $(\kappa + 1)$-bit and $(\kappa + 2)$-bit, and we add $\kappa + 1$-bit and $(\kappa + \Theta)$-bit for the Θ th one. Than cost can be expressed as $\tilde{\mathcal{O}}(\Theta(\kappa + 1))$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{10})$. it should be noted that no need to compute cost of reduction because reduction in $2^{\kappa+1}$ is just taking first two bits of result of summation. Finally, set $y_i = u_i/2^\kappa$ and $\vec{y} = \{y_1, \cdots, y_\Theta\}$. To compute y_i's, we need $\kappa+1$ addition and $\kappa+1$ multiplications. (Because the y_i is a positive number between

[0,2) with κ bits of precision after the binary point). Then, cost can be evaluated as $\mathcal{O}(\Theta(\kappa+1))$. Furthermore, after using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{10})$.

6.1.3.2 Encrypt and Evaluate

Take ciphertext c^* generated in section 6.1.2. Then create z_i for $i \in \{1, \cdots, \Theta\}$,,also keep z_i as $n = [log\theta]$ + 3 bits of precision after the binary point.

$$z_i \leftarrow [c^* \ . \ y_i]_2$$

To create z_i's, we need to do Θ multiplications. Note that the each $y_i \in [0, 2)$, with κ bits of precision after the binary point. It means we can take length of the y_i's as $(\kappa+1)$-bit. Furthermore, length of the c^* is equal to γ-bit as implied in the section 6.1.2. Thus the complexity can be expressed as $\mathcal{O}(\Theta * (\kappa+1) * \gamma)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{15})$.

6.1.3.3 Decryption

Lastly, output the message as following.

$$m' \leftarrow [c^* - \lfloor \sum_{i \in S} s_i.z_i \rceil]_2$$

In this step, main cost originates from $(\Theta+1)$ addition. Here length of z_i is taken as n+1 bit for $n = \lceil log\theta \rceil + 3$. Then, cost can be expressed as $\mathcal{O}(\Theta * (n))$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^5)$.

6.2 Time complexity of Batch DGHV scheme

This section includes the time complexity calculations of all parts of the fully homomorphic batch DGHV schemes, namely Key Generation, Encryption, Squashing, respectively.

6.2.1 Key Generation

Generate η-bit p_j's, which are distinct primes, for $j \in [0, l)$ and compute π by multiplying p_j's, $\pi \leftarrow p_i.\pi$. To compute π, we need to do $\ell - 1$ multiplication. As mentioned in beginning of the chapter 6, length of π equal to η. Than,cost can be expressed as $\mathcal{O}(\eta^2(\ell-1)^2)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^4(\ell-1)^2)$. Let us define the public key element $x_0 = q_0.\pi$ where q_0 is chosen the following set $q_0 \leftarrow \mathbf{Z} \cap [0, 2^\gamma/\pi]$. To compute x_0, we need to do just 1 multiplication with $(\ell-1)\eta$-bit π and $(\gamma - (\ell-1)\eta)$-bit x_0. Our cost is equal to $\mathcal{O}(\eta^2(\ell-1)(\gamma-(\ell-1)))$. After using the convenient parameter set, total cost can be expressed as $\tilde{\mathcal{O}}(\lambda^7(\ell-1) - \lambda^4(\ell-1)^2)$.

For the next step of key generation, choose the integers x_i and x_i' with a quotient by π uniformly and independently distributed in $\mathbf{Z} \cap [0, q_0)$, and with the following distribution modulo p_j for $j \in [0, l)$

$$i \in [1, \tau], \qquad x_i \bmod p_j = Q_j r_{i,j}, \qquad r_{i,j} \leftarrow \mathbf{Z} \cap (-2^\rho, 2^\rho),$$

$$i \in [1, \ell-1], \qquad x_i' \bmod p_j = Q_j r_{i,j}' + \delta_{i,j}, \qquad r_{i,j}' \leftarrow \mathbf{Z} \cap (-2^\rho, 2^\rho),$$

Choose x_i and x_i' according to the CRT algorithm.

Algorithm 1 Calculation of x_i using CRT algorithm

$N \leftarrow 1$
for $j = 0, \cdots, \ell - 1$ **do**
  $N \leftarrow N.p_j$
end for
for $i = 1, \cdots, \tau$ **do**
  $x_i \leftarrow 0$
  for $j = 0, \cdots, \ell - 1$ **do**
    $N_j \leftarrow N_j/p_j$
    $M_j \leftarrow N_j^{-1} \bmod p_j$
    $x_i \leftarrow [x_i + Q_j.r_{i,j}.M_j.N_j] \bmod N$
  end for
  return (x_i)
end for

Let us start with the evaluation of cost N. As remembered in the beginning of the section $N = \pi$, and its cost is $\tilde{\mathcal{O}}(\lambda^4(\ell-1)^2)$. Then, for calculating N_j, we need to do $(\tau\ell)$ division with $(\eta(\ell-1))$-bit N_j and η-bit p_j. Hence our cost is equal to $\mathcal{O}(\tau\eta^2\ell(\ell-1))$. After using the convenient parameter set, total cost is $\tilde{\mathcal{O}}(\lambda^9(\ell^2-\ell)$. The next step is the calculating M_j. To evaluate the cost of M_j, we need to compute $\tau\ell$ number of inversion with division. The result is equal to $\mathcal{O}(\eta^3+\ell\tau\eta^2(\ell-1))$ .If we use the appropriate parameter set, total cost is $\tilde{\mathcal{O}}(\ell\lambda^{11}+\lambda^9(\ell^2-\ell))$. Later, to compute x_i the number of multiplication is $(3\ell\tau)$, the number of addition is $(\tau\ell)$, and finally the number of reduction is $(\tau\ell)$. We do the computation in three steps These are the followings,

$$M_j * N_j,$$

$$r_{i,j} * M_j * N_j,$$

$$Q_{i,j} * r_{i,j} * M_j * N_j.$$

First, cost of to compute $M_j * N_j$ is equal to $\mathcal{O}(\eta^2\tau\ell(\ell-1))$, and after the multiplication new length of $M_j * N_j$ is changed as $(\eta\ell)$. Moreover, If we use the convenient transformations, total cost is $\tilde{\mathcal{O}}((\ell^2-\ell)\lambda^9)$. Secondly, cost of to compute $(r_{i,j} * M_j * N_j)$ is equal to $\mathcal{O}(\eta\rho\tau\ell^2)$ and length of $(r_{i,j}*M_j*N_j)$ is altered as $(\eta\ell+\rho)$. After using the convenient parameter set, it can be stated as $\tilde{\mathcal{O}}(\ell\lambda^9)$. Eventually, cost of the compute $(Q_{i,j}*r_{i,j}*M_j*N_j)$ is equal to $\mathcal{O}((\eta\ell+\rho)\tau\ell)$, and length of $(Q_{i,j}*r_{i,j}*M_j*N_j)$ is changed as $(\eta\ell+\rho+\mu)$. It should be noted that μ taken as 2 to allow FHE transformation. After using the convenient parameter set, total cost is $\tilde{\mathcal{O}}(\ell^2\lambda^7)$. Finally $sk = (p_j)_{\{0\le i\le l-1\}}$ and $pk = \{x_0, (x'_i)_{\{0\le i\le l-1\}}, (Q_i)_{\{0\le i\le l-1\}}, (x_i)_{\{0\le i\le \tau\}}\}$

6.2.2 Encryption

For any $m = (m_0, \cdots, m_{l-1})$ with $m_i \in \mathbf{Z}_{Q_i}$, choose a random binary vector $b = (b_i)_{\{1\le i\le\tau\}} \in \{0,1\}^\tau$. Then, ciphertext can be found by using the folowing encryption algorithm.

$$c = \left[\sum_{i=0}^{l-1} m_i.x'_i + \sum_{i=1}^{\tau} b_i x_i\right]_{x_0}$$

In this step we compute complexities of $l + \tau$ additions with 1 reduction. We know that length of the x_i and x'_i is equal to $(\mu + \eta\ell + \rho)$, and also length of the x_i is equal to γ from section 6.2.1. Then, cost can be expressed as $\mathcal{O}(l\tau(\mu + \eta\ell + \rho))$ for multiplication, and $\mathcal{O}(\ell\gamma(\mu + \eta l + \rho))$ for reduction. After using the convenient parameter set, total cost is $\tilde{\mathcal{O}}(\lambda^7(\ell^2 + \ell))$.

6.2.3 Squashing

A set $\vec{y} = \{y_1, \cdots, y_\Theta\}$ is added to the public key. Here $y_i \in \mathbf{Q}$, and $0 \leq y_i < 2$ with κ bits of precision. The purpose is to find a sparse subset $S_j \subset [0, \Theta - 1]$ of size Θ such that $\sum_{\{i \in S_j\}} y_i \approx 1/p_j \ (mod\ 2)$.

6.2.3.1 Key Generation

Addition to the private key $sk^* = (p_0, \cdots, p_{l-1})$ and public key pk^*, an adition item x_p should be set as following.

$$x_{p_j} \leftarrow \lfloor 2^\kappa / p_j \rceil$$

To generate x_p we need just l division. As we mentioned in section 6.1.3, using $\kappa = \gamma + 2$ is sufficient. Then our complexity is $\mathcal{O}(\eta\ell(\gamma + 2))$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\ell^2\lambda^7)$, and new length of x_p is equal to $(\gamma - \eta + 2)$. Than, choose at random a Θ-bit vector $\vec{s_j} = \langle s_{j,0}, \cdots, s_{j,\Theta-1} \rangle$ with hamming weight θ for $0 \leq j < l$. Choose at random integers $u_i \in [0, 2^{\kappa+1})$, where $i = \{0, \cdots, \Theta - 1\}$ which is dependent to the following rule.

$$\sum_{i=0}^{\Theta-1} u_i s_{j,i} = x_p \ (mod\ 2^{\kappa+1})$$

For satisfying the condition, we need to do Θ addition. We add $(\kappa + 1)$ bits u_i's with each others. Note that, while first we add two $(\kappa + 1)$-bits u_i, for the second addition we add $(\kappa+1)$-bit and $(\kappa+2)$-bit, and we add $\kappa+1$-bit and $(\kappa+\Theta)$-bit for the Θ th one. Than cost can be expressed as $\tilde{\mathcal{O}}(\Theta(\kappa+1))$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{10})$. it should be noted that no need to compute cost of reduction because reduction in $2^{\kappa+1}$ is just taking first two bits of result of summation. Finally, set $y_i = u_i/2^\kappa$ and $\vec{y} = \{y_0, \cdots, y_{\Theta-1}\}$. To compute y_i's, we need $\kappa + 1$ addition

and $\kappa + 1$ multiplications. (Because the y_i is a positive number between [0,2) with κ bits of precision after the binary point). Then, cost can be evaluated as $\mathcal{O}(\Theta(\kappa + 1))$. Furthermore, after using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{10})$.

6.2.3.2 Expanding

Take ciphertext c^* generated in section 6.2.2. Then create z_i for $i \in \{0, \cdots, \Theta - 1\}$, with $n = \lceil log_2(\theta+1) \rceil$ bits of precision after the binary point for each z_i.Then, output the ciphertext c^* and $z = (z_i)_{\{i=0,\cdots,\Theta-1\}}$.

$$z_i \leftarrow [c^* \, . \, y_i]_2$$

To create z_i's, we need to do Θ multiplications. Note that $y_i \in \mathbf{Q}$, and $0 \leq y_i < 2$ with κ bits of precision. It means that we can take length of the y_i's as $(\kappa + 1)$-bit. Furthermore, length of the c^* is equal to γ-bit as implied in the section 6.1.2. Thus the complexity can be expressed as $\mathcal{O}(\Theta(\kappa + 1)\gamma)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^{15})$.

6.2.3.3 Decryption

Lastly output the message $m = (m_0, \cdots, m_\ell)$ with

$$m_j \leftarrow \left[\lfloor \sum_{i=0}^{\Theta-1} s_{j,i} . z_i \rceil \right]_2 \oplus (c \; mod \; 2)$$

In this step, main cost originates from Θ addition. Here length of z_i is taken as n+1 bit for $n = \lceil log_2(\theta + 1) \rceil$. Then, cost can be expressed as $\mathcal{O}(n\Theta)$. After using the convenient parameter set, the total cost is $\tilde{\mathcal{O}}(\lambda^5)$.

6.3 Comparison and Result

In this section, time complexity of DGHV scheme[30] and batch DGHV scheme [7] are compared. As we mention in first chapter while the DGHV scheme encrypts the one-bit message, the batch DGHV scheme encrypts an ℓ-bit message vector at a

time. The main purpose is to find out which encryption system is more efficient for encrypting l-bit messages. The first choice is to use the DGHV schema for l-times. The second choice is to use the batch DGHV scheme one time. We conclude that for message size l when security parameter $\ell \leq \lambda^{3/2}$ using batch scheme is more efficient than using DGHV scheme.

Table 6.1: Time Complexity Comparison

	DGHV scheme	Batch DGHV scheme
Key Generation of SwHE	$\tilde{\mathcal{O}}(\lambda^{12})$	$\tilde{\mathcal{O}}(l\lambda^{11} + ((l^2 - l)\lambda^9)$
Encyption	$\tilde{\mathcal{O}}(\lambda^{10})$	$\tilde{\mathcal{O}}(l^2\lambda^7)$
Squahing-Key Generation	$\tilde{\mathcal{O}}(\lambda^{10})$	$\tilde{\mathcal{O}}(\lambda^{10})$
Squahing- Evaluation and Encryption	$\tilde{\mathcal{O}}(\lambda^{15})$	$\tilde{\mathcal{O}}(\lambda^{15})$
Squahing - Decryption	$\tilde{\mathcal{O}}(\lambda^5)$	$\tilde{\mathcal{O}}(\lambda^5)$

CHAPTER 7

CONCLUSION

Today, with the development of technology, more reliable and efficient algorithms are provided. People need these algorithms to store and process their private information.To operate on encrypted data without decrypting it, and delivering this data securely to the other party has become an important goal for studies.

In this thesis, we focus on the FHE scheme over the integers and batch FHE scheme. We give some definition and properties of FHE. Then, we demonstrate the structure of the functions that can be evaluated homomorphically using arithmetic circuits. We also describe real-life applications of homomorphic encryption. These applications have been developed mainly in the fields of health, finance, advertising, and education. Furthermore, we present symetric and asymetric key version of the DGHV scheme. Then, we demonstrate why this schemes are homomorphic under addition and multiplication operations. Lastly, we give the general review of FHE schemes and a way to convert a SwHE scheme to FHE scheme. Moreover, we mention some efficieny problem of DGHV scheme and ways to improve it. These are finding more efiicient SwHE schemes, extending the message size or message space, speeding up the bootstrapping process, eliminating the bootstrapping procedure and the squashing process, and reducing the public key size of the existing scheme. We mainly focus on "extending message size and message space" solution among them. Finally, we compare time complexities of DGHV scheme and batch DGHV scheme. We show that for message size ℓ when security parameter $\ell \leq \lambda^{3/2}$ using batch scheme is more efficient than using DGHV scheme.

REFERENCES

[1] J. Benaloh, Dense Probabilistic Encryption, pp. 120–128, 1994.

[2] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, Patient controlled encryption: ensuring privacy of electronic medical records, in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 103–114, 2009.

[3] D. Boneh, E.-J. Goh, and K. Nissim, Evaluating 2-DNF Formulas on Ciphertexts, in D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and J. Kilian, editors, *Theory of Cryptography*, volume 3378, pp. 325–341, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, ISBN 978-3-540-24573-5 978-3-540-30576-7, series Title: Lecture Notes in Computer Science.

[4] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, ACM Transactions on Computation Theory (TOCT), 6(3), pp. 1–36, 2014.

[5] Z. Brakerski and V. Vaikuntanathan, Efficient Fully Homomorphic Encryption from (Standard) LWE, in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 97–106, IEEE, Palm Springs, CA, USA, October 2011, ISBN 978-0-7695-4571-4 978-1-4577-1843-4.

[6] Z. Brakerski and V. Vaikuntanathan, Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages, in D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and P. Rogaway, editors, *Advances in Cryptology – CRYPTO 2011*, volume 6841, pp. 505–524, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ISBN 978-3-642-22791-2 978-3-642-22792-9, series Title: Lecture Notes in Computer Science.

[7] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, Batch fully homomorphic encryption over the integers, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 315–335, Springer, 2013.

[8] J. H. Cheon, K. Han, and D. Kim, Faster bootstrapping of fhe over the integers, in *International Conference on Information Security and Cryptology*, pp. 242–259, Springer, 2019.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, MIT press, 2009.

[10] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE transactions on information theory, 31(4), pp. 469–472, 1985.

[11] H. B. Enderton, *A mathematical introduction to logic*, Elsevier, 2001.

[12] C. Gentry, *A fully homomorphic encryption scheme*, Stanford university, 2009.

[13] C. Gentry, Fully homomorphic encryption using ideal lattices, in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, p. 169, ACM Press, Bethesda, MD, USA, 2009, ISBN 978-1-60558-506-2.

[14] C. Gentry and S. Halevi, Fully homomorphic encryption without squashing using depth-3 arithmetic circuits, in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 107–109, IEEE, 2011.

[15] S. Goldwasser and S. Micali, Probabilistic encryption, Journal of Computer and System Sciences, 28(2), pp. 270–299, April 1984, ISSN 00220000.

[16] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, A pseudorandom generator from any one-way function, SIAM Journal on Computing, 28(4), pp. 1364–1396, 1999.

[17] N. Howgrave-Graham, Approximate integer common divisors, in *International Cryptography and Lattices Conference*, pp. 51–66, Springer, 2001.

[18] A. Juels, Targeted advertising... and privacy too, in *Cryptographers' Track at the RSA Conference*, pp. 408–424, Springer, 2001.

[19] D. Kahn, *The Codebreaker: The Story of Secret Writing*, Macmillam, 1976.

[20] R. M. Karp and V. Ramachandran, A survey of parallel algorithms for shared-memory machines, 1989.

[21] L. Adleman ,R. L. Rivest and M. L. Dertouzos, On data banks and privacy homomorphisms, Foundations of secure computation, 4(11), pp. 169–180, 1978.

[22] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*, John Wiley & Sons, Inc., 1990.

[23] P. Martins, L. Sousa, and A. Mariano, A survey on fully homomorphic encryption: An engineering perspective, ACM Computing Surveys (CSUR), 50(6), pp. 1–33, 2017.

[24] M. Naehrig, K. Lauter, and V. Vaikuntanathan, Can homomorphic encryption be practical?, in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 113–124, 2011.

[25] K. Nuida and K. Kurosawa, (batch) fully homomorphic encryption over integers for non-binary message spaces, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 537–555, Springer, 2015.

[26] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*, Springer Science & Business Media, 2009.

[27] P. Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, in J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592, pp. 223–238, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, ISBN 978-3-540-65889-4, series Title: Lecture Notes in Computer Science.

[28] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, Quantifying location privacy, in *2011 IEEE symposium on security and privacy*, pp. 247–262, IEEE, 2011.

[29] A. Shpilka and A. Yehudayoff, Arithmetic Circuits: a survey of recent results and open questions, p. 123.

[30] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully Homomorphic Encryption over the Integers, in D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and H. Gilbert, editors, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110, pp. 24–43, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, ISBN 978-3-642-13189-9 978-3-642-13190-5, series Title: Lecture Notes in Computer Science.

[31] C. Zhigang, W. Jian, C. Liqun, and S. Xinxia, Review of how to construct a fully homomorphic encryption scheme, International Journal of Security and Its Applications, 8(2), pp. 221–230, 2014.