A seminar report on:

"FULLY HOMOMORPHIC ENCRYPTION"

Prepared by : Shubham Agarwal

Roll. No : U19CS046

Class : B.Tech-III Computer Science and Engineering 5th Semester

Year : 2021-22

Guided by : Sachi Shah

Department of Computer Engineering

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

CERTIFICATE

This is to certify that the seminar report entitled **Fully Homomorphic Encryption** is prepared and presented by **Mr. Shubham Agarwal** bearing Roll No. : **U19CS046** 3rd Year of **B.Tech(Computer Science and Engineering)** and his work is satisfactory.

GUIDE	JURY	HOD
(Sachi Shah)		COED

Table of Contents

List of Figures

List of Abbreviations

AI Artificial intelligence
FHE Fully Homomorphic Encryption
IoT Internet of Things
msb most significant bit
PHE Partially Homomorphic Encryption
PII Personally Identifiable Information
SWHE Somewhat Homomorphic Encryption

Abstract

In the technical world, cloud computing, the internet of things (IoT), and blockchain technology have become the most powerful innovation. They pose a significant risk of information leakage, as this information can be targeted by cloud insiders or hackers, resulting in the computerization of privacy. There is a crucial need to create a solution that ensures privacy even if cloud security is breached. There is a solution for cryptography. Fully Homomorphic Encryption is an encryption system that allows any third party or provider to compute encrypted data and produce accurate results without revealing any information about the original material. Gentry [3] created the first such scheme in 2009, which was a big breakthrough, and since then, improvements have been made to the scheme to make it faster and more practical to employ in real-world applications. In this research, we look at what fully homomorphic encryption is and the significant issues that come with implementing it.

Keyword Keywords – Cryptography, cloud computing, Fully Homomorphic Encryption, blockchain, IoT

1 Introduction

1.1 OUTSOURCED COMPUTATION

In this era, information technology has made mobile phones, personal computers, and the internet necessary. Because these devices have low computing and storage capacity, they cannot use them for application servers that demand a large amount of processing and storage capacity and fault tolerance, which cheap commodity hardware cannot supply. It cannot be used to support current applications that are based on AI and Machine Learning. This type of need necessitates the deployment of expensive hardware and software infrastructure, which is prohibitively expensive for ordinary people and even small businesses. This condition opens up a new business opportunity to build an infrastructure that can serve all of these users simply by moving computation to high-end computer resources, a process known as outsourcing computation.

Outsource computing is defined as a scheme in which a party P provides data D to a party Q, who processes the data according to P's needs, produces results, and gives them back to P or another party. Outsourced computing problems may be solved in a variety of ways, but the most prevalent is the "Cloud." For network users, the cloud delivers extensive, flexible, and on-demand distant storage and processing capabilities. Data science is now being used by big apps to better serve their customers, which has made cloud so popular that all major technology companies across the world, including Microsoft, Amazon, IBM, and Google, are now in the cloud industry.

Most cloud-based applications require data owners to send private and sensitive data to the cloud for processing over the internet or through any other medium. This raises concerns about the security of outsourcing computation schemes because users lose control of their data when it is sent to the cloud. There is always the risk of data leakage from the cloud, which can create insecurity for businesses and individuals. If sensitive data such as medical records or financial statements is compromised, a firm can lose important data and insights that can damage their business. For an individual, if sensitive data such as medical records or financial statements is hacked, it can pose severe privacy problems.

Recent studies in cloud security have presented security approaches that are utilized in network security, such as storing data in encrypted form, which maintains secrecy even if there is a security breach on the cloud and an adversary can access stored data. However, when a user wishes to do a computation on the stored data, the data must first be decrypted in order to obtain acceptable results, and all computations must be performed on private data.The challenge of secure computing can be solved by providing computation providers with a computation scheme that allows them to calculate on encrypted data without decryption while yet producing the same output (encrypted) as if they were processing plain data.

Homomorphic encryption methods are a type of encryption that allows users to conduct out-

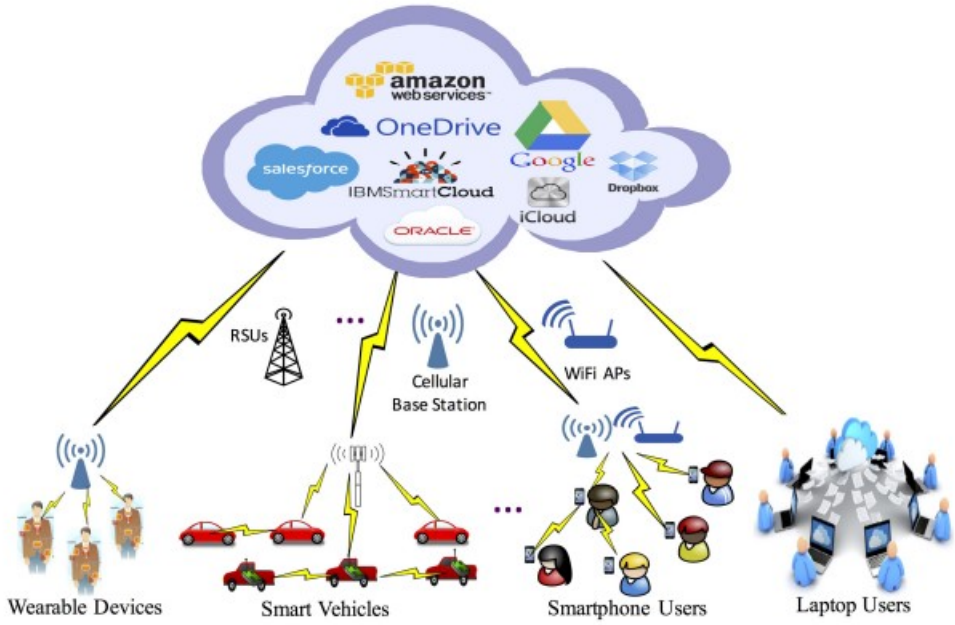

Figure 1.1: Cloud Computing

sourced computing without having to worry about trust.

1.2 Fundamentals of Cryptography [7]

The security of encryption methods should not be dependent on code obfuscation but rather on the confidentiality of the key used in the encryption process. There are two types of en-cryption techniques: symmetric encryption schemes and asymmetric encryption methods. In the following sections, we will quickly go through each of these plans.

1.2.1 Symmetric encryption schemes

Before beginning any secure communication session, the sender and receiver agree on the key they will use in these schemes. As a result, such methods cannot be used directly by two people who have never met previously. This also indicates that we must have a separate key for each individual in order to connect with them. Because these schemes require a high number of keys, key creation and administration become substantially more complicated processes. However, symmetric systems have the benefit of being extremely fast, and they are employed in situations where speed of execution is critical.

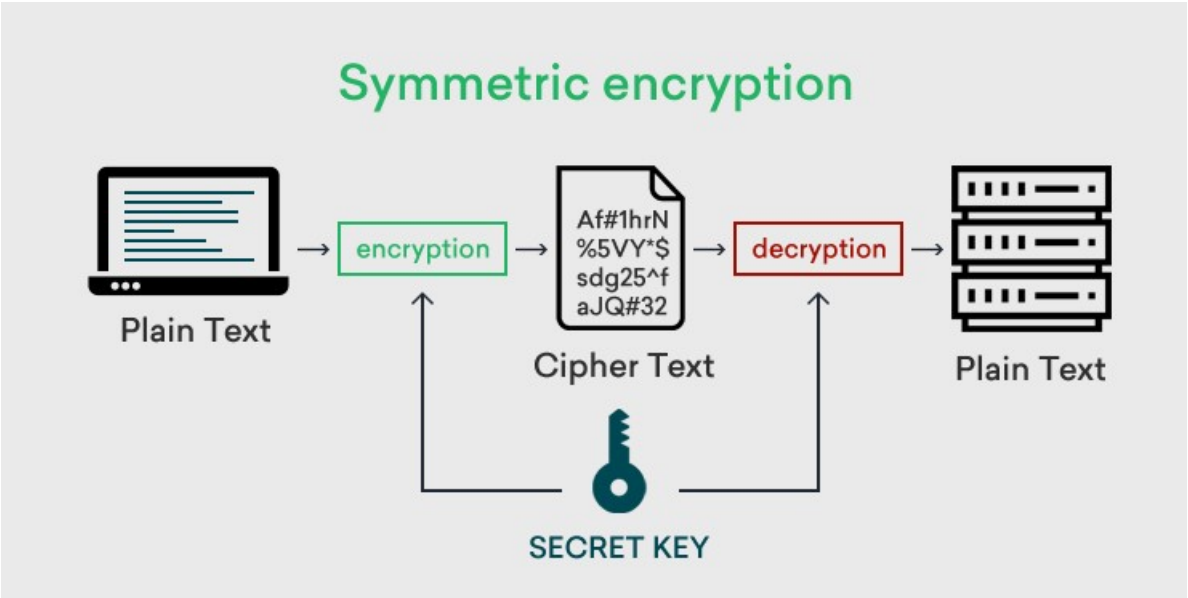

Figure 1.2: Symmetric Encryption

1.2.2 Asymmetric encryption schemes

Every participant in these systems has a pair of keys, one private and one public. While person's private key is known exclusively to her, the public key of each member is not confidential in the group. So schemes are more secure than their symmetric equivalents, and they do not require previous agreement on a shared key between the communicating parties before initiating a communication session.

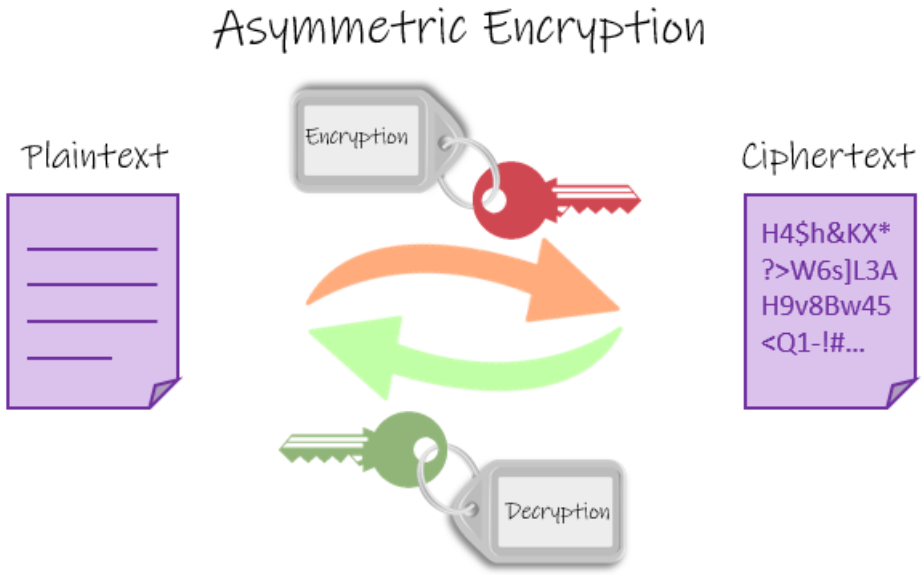

Figure 1.3: Asymmetric Encryption

1.3 Homomorphic Encryption

Before we go into homomorphic encryption, it's important to understand what "encryption" is. Encryption is like converting data from its original plaintext form to an encoded version known as ciphertext using secure keys held by only authorized parties. Decryption is the process of selecting keys in such a way that they may be decrypted. To utilize encryption and decryption, both parties must know the encryption method and the keys that will be used for encryption and decryption, which must be produced prior to encryption. An encryption method is made up of a trio of encryption, decryption, and key generation algorithms.

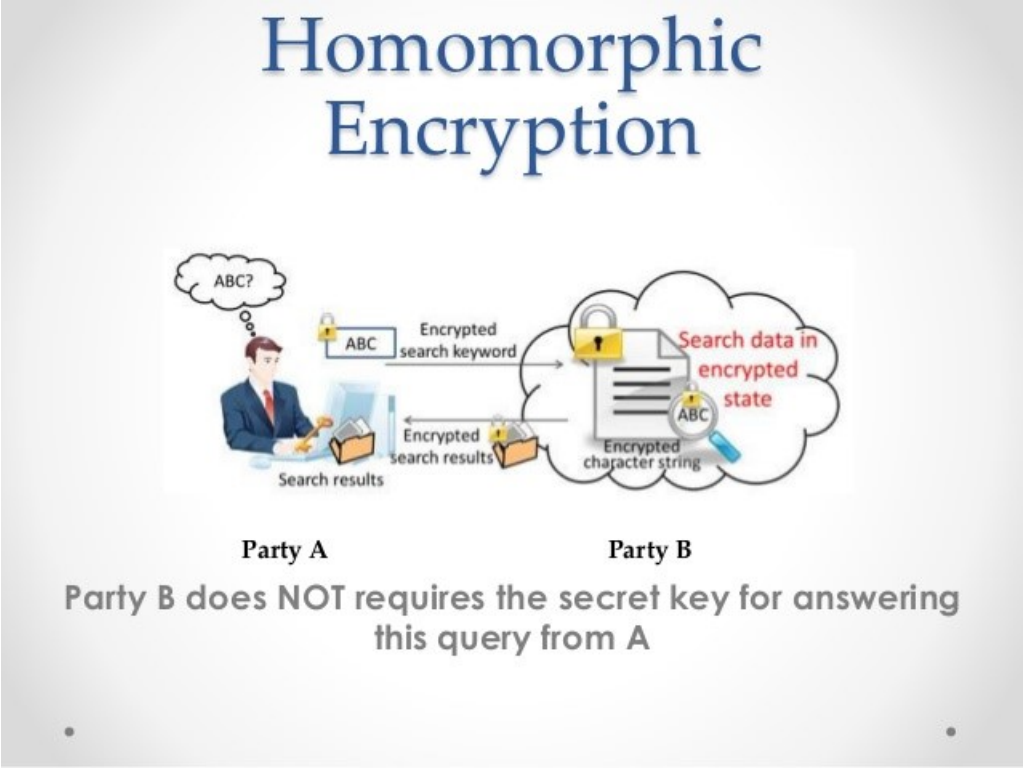

Figure 1.4: Homomorphic Encryption

Homomorphic encryption is considered an Encryption that allows you to compute over encrypted material without having to decode it. Consider a data object X that has Encryption $Encr(X)$ and a function $f()$ that has to be evaluated on X in a homomorphic scheme that contains an equivalent function f',.This ensures that $f'(Encr(X)) = Encr(f(X))$.
Homomorphic encryption systems are divided into three categories based on the quantity and fundamental functions they support: partly, somewhat, and completely homomorphic encryption methods.

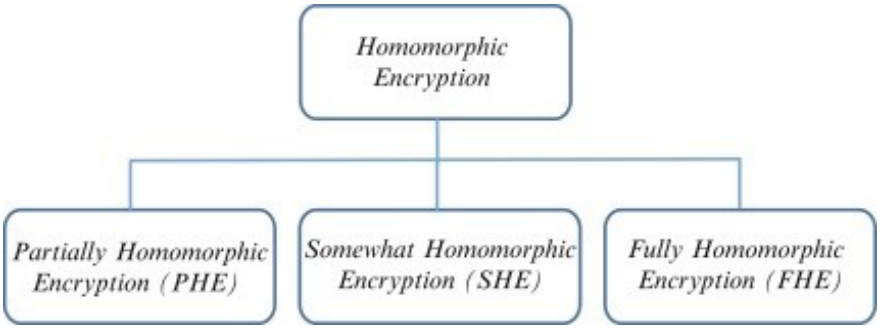

Figure 1.5: Types Of Homomorphic Encryption

• **Properties of Homomorphic Encryption**

Homorphic Encryption comes from abstratc algebra.Now assume the plain text $M1, M2$ and Encryption scheme E. It will be Homorphic Encryption or Fully Homomorphic Encryption if it follows below properties:

• **Additive Property**

$$\boldsymbol{Dec_{sk}(Enc_{pk}(M1) + Enc_{pk}(M2)) = M1 + M2} \tag{1.1}$$

• **Multiplicative Property**

$$\boldsymbol{Deck(C_k(M1)^* C_k(M2)) = M1^* M2} \tag{1.2}$$

1.3.1 Partially Homomorphic Encryption

Homomorphic multiplication or homomorphic addition are supported by partially homomorphic encryption systems. Multiplicative homomorphism exists in the RSA and ElGamal cryptosystems. Additive homomorphism is a well-known paillier cryptosystem. PHE systems can be beneficial in situations when just addition or multiplication is required, such as safe electronic voting, which only requires homomorphic addition. PHE methods now in use are unable to calculate the whole set of 6 Boolean operators, i.e., the NAND operator. Instead, they use a commutative ring that contains the Boolean set 0, 1 to enable either addition or multiplication. The NAND operator, on the other hand, necessitates both homomorphic addition and multiplication, as seen in the following expression [4].

$$x_1 NAND x_2 = 1 - x_1.x_2, x_1, x_2 \varepsilon \{0, 1\} \tag{1.3}$$

As a result, PHE schemes' usefulness is severely restricted.

1.3.2 Somewhat Homomorphic Encryption

Somewhat homomorphic encryption is a homomorphic technique that permits a limited number of homomorphic multiplication and addition to be computed (SWHE). The so-called

"Poly-Cracker" schemes, released in the early 1990s, were the first encryption systems that might be deemed SWHE. These approaches are based on the multivariate ring ideal remainder issue. Encryption is accomplished here by masking plaintext with a random element of a publicly known ideal. Regrettably, virtually all of these programs lack security assurances.

1.3.3 Fully Homomorphic Encryption

A homomorphic encryption technique with no restrictions on the types of functions that may be used, the number of assessments that can be performed, or the depth of processing. On the encrypted data, a completely homomorphic method can assess a circuit of any complexity. Gentry's [3] FHE system, proposed in 2009, was the first big advance for this type of method. We'll go over everything in depth in the following chapter.

1.4 Various Utilization

Homomorphic Encryption methods are more powerful than traditional encryption standards because they allow you to compute functions on encrypted data, allowing you to use any outsourcing computing solution without worrying about data security, which is a major benefit. With the increasing use of cloud-based solutions in real-world applications, homomorphic encryption schemes are becoming increasingly popular as a way to make clouds more safe and trustworthy.

1.4.1 Security for Cloud Processing

One of the finest use cases for FHE is secure data processing in clouds, and this is the use case that has driven research in this subject since the beginning. FHE methods can be used to process encrypted data in encrypted form without actually utilizing the data, ensuring privacy. [5]

1.4.2 Encrypted Search

Searching for encrypted data is one of the most important applications of homomorphic encryption because it protects users' privacy while browsing. When searching in encrypted mode, a third party cannot keep track of the user's browsing history, which could be used for advertising or malicious purposes. Users encrypt their queries using their public key in the encrypted search [6] method, and the query is performed on the encrypted data by the server, making the user's browsing history untraceable. Meng Shen introduced the SWHE technique, which encrypts graphs and allows for the calculation of Shortest Distance Queries across them.

Figure 1.6: Secure Cloud Processing

1.4.3 Implimentaion in Medical Science

Medical data is very confidential data, and it is frequently the case that an individual does not want to disclose this information. Many countries have placed limits on the use of healthcare data since it falls under the PII category; however, there are a variety of use cases in the healthcare industry that require medical records to train machine learning models in order to build next-generation healthcare solutions. This type of data may be utilized for research purposes utilizing the FHE technique since encrypted data does not represent a security risk to individuals, but it is valuable to researchers who can do computations on encrypted data.

1.4.4 For Secured E-voting

Homomorphic encryption may be used to create a secure electronic voting system in which all ballots with user selections are encrypted, and the results are calculated using homomorphic processes. It is difficult to obtain the plaintext of an individual ballot, which aids in the privacy of an individual voter.

1.4.5 To Prevent Phishing

Phishing attacks attempt to obtain data from users by simulating their experience in order to convince them that they are submitting credentials to a legitimate entity, which is then used for harmful reasons. By demanding encrypted credentials from users, which can be readily validated via homomorphic encryption, homomorphic methods may be utilized to thwart fishing

attacks. Even if a person offers his or her credentials to an opponent, in this case, it will be useless to him. There will be more and more uses of homomorphic encryption in the future as self-awareness of privacy protection grows, and homomorphic encryption develops.

1.5 Related work

Gentry [3] presented the first fully homomorphic encryption method in 2009, which was the first major advance in the realm of FHE research. It was built on the well-known ideal lattices issue. Despite the fact that it had a significant influence on the area of FHE, It was computationally costly, resulting in low efficiency. [2] Following the Gentry's scheme, researchers became very interested in lattice-based cryptosystems, and many FHE schemes were introduced to improve the efficiency of the Gentry's scheme. However, most of them had a complex key generation scheme, which, combined with the large size of the key and inline, rendered these schemes impractical for real-world use.

Following Gentry's approach in 2010, Dijk proposed an integer-based FHE technique. This scheme only works with integers; all ciphertext and keys are integers, making it much easier to understand and implement than previous FHE schemes based on ideal lattices. However, this scheme suffers from the problem of large ciphertext and key sizes, making it unsuitable for real-world applications.

The first LWE-based FHE scheme (BV) was introduced by Brakerski and Vaikuntanathan in 2011. In homomorphic multiplications, the re-linearization approach was introduced to regulate the ciphertext dimension. Furthermore, the dimension-modulus reduction approach was presented as a novel strategy for reducing the decryption algorithm in order to make the system bootstrap able and completely homomorphic.

A leveled FHE method was presented by Brakerski, Gentry, and Vaikuntanathan in 2012. (BGV). This method was built on re-linearization and dimension-modulus reduction techniques that were enhanced for key switching and modulus switching procedures in the BGV scheme to give it more control over the ciphertext's dimensions and noise level. Gentry, Sahai, and Waters proposed the GSW system in 2013. Approximate eigenvectors of matrices were used in this method. Every calculation in the GSW scheme is conducted as a matrix operation. The ciphertext and key are represented as matrices, and all homomorphic operations are performed as matrix operations. Due to matrix operations, the ciphertext size is nearly always limited, which eliminates the need for key flipping. The flatten approach may be used to make the GSW scheme scale-invariant, which also overcomes the problem of modulus switching. This system overcomes all of the key problems of its predecessor. However, the matrix operations are computationally costly, making this scheme a computational bottleneck.

Ducas and Miccianico proposed the DM method, which is based on homomorphic NOT and NAND gates that may be utilized to construct any digital circuit. All homomorphic operations in the DM scheme are ciphertext vector additions, which are very easy to compute and make this

scheme very simple to implement. The issue with scheme is that ciphertext must be refreshed after each homomorphic operation, which increases the amount of computation required in comparison to plain data evaluation and becomes a bottleneck for this scheme. Other research on the design of LWE-based FHE schemes focuses on increasing the efficiency of the bootstrapping process and optimizing it.

2 Fully Homomorphic Encryption Schemes

Genetry[3] released his crucial work in 2009, presenting a novel SHE method based on the ideal coset problem over ideal lattices. This is the first fully homomorphic encryption method ever devised. We'll go over Gentry's First scheme in the next part, as well as several novel homomorphic encryption techniques.

2.1 Gentry's Homomorphic Encryption Scheme

Gentry's initial suggested design was based on the notion of using bootstrapping to overcome the problems created by noise in homomorphic evaluation high-depth circuits. We will not cover the exact mathematics involved in Gentry's approach due to the incredible complexity of Gentry's scheme and Lattice-based encryption. We'll go through the general issues with homomorphic encryptions before looking at how this approach addresses them.

2.1.1 Noise problem

Noise is a tiny random number that is introduced to a ciphertext during the encryption process to assure the cryptosystem's semantic security.
Consider a basic encryption system with a secret t and encryption - decryption as follows to see the issues created by additional noise in homomorphic operations.

$$Encr(X) = X.t + e \tag{2.1}$$

$$Decr(X) = (X - mod(X,t)) \tag{2.2}$$

Consider the following two ciphertexts, C1 and C2, which were both produced using the same technique. When these two ciphertexts are added together, the result is

$$C = C1 + C2 \tag{2.3}$$

$$C = X_1.t + e_1 + X_2.t + e_2 \tag{2.4}$$

$$C = (X_1 + X_2).t + (e_1 + e_2) \tag{2.5}$$

If $e_1 + e_2$ is less than t, the decryption of C will yield a result identical to $X_1 + X_2$. Otherwise, it will yield an incorrect result. Multiplication can exceed this limit of the permissible amount of noise in a few rounds of successive computation on a single ciphertext, resulting in inaccurate output. Multiplication is a very simple operation in real-world applications that often requires multiple rounds of complex processing, including exponentiation. This restriction on the quantity of noise imposes a limit on the number of times a ciphertext may be computed without losing its accuracy.

2.1.2 Levelled fully Homomorphic Scheme

Consider an encryption method that can evaluate d consecutive homomorphic operations on ciphertext without introducing noise; such techniques are referred to as Levelled Homomorphic Encryption.

The main disadvantage of such a scheme is that functions with depth within a predefined depth range can be evaluated accurately without decryption. However, such schemes are not useful in real-world applications because it is not always possible to know the depth of a function ahead of time, and functions exceeding the depth limit cannot be executed homomorphically.

2.1.3 Bootstrapping

Gentry suggested a technique called bootstrapping to overcome the problem of noise in levelled homomorphic encryption systems. Consider a homomorphic operations-only Levelled homomorphic Encryption scheme with a maximum depth of d.

Gentry suggested a system in which the intermediate ciphertext is decrypted using homomorphic decryption and the secret key, which is encrypted with the same public key, is delivered to the compute end; this key is known as a bootstrapping key. The assumption is that the bootstrapping key does not reveal any information about the secret key, which is known as the circular security assumption. This technique of refreshing is known as bootstrapping, and the schemes that are capable of doing so are known as bootstrap able.

Although bootstrapping renders Levelled homomorphic systems fully homomorphic, it involves a significant amount of computing, and bootstrapping may be necessary to assess a single function several times.

2.1.4 Squashing

Bootstrapping seems nice when it offers FHE, but it's not so simple since it needs homomorphic decryption, and decryption circuits (functions) in general have a large depth, requiring these schemes to perform decryption or encryption repeatedly, which is time and computationally expensive.

There is one solution to this problem is to reduce the depth of the decryption circuit and shift the complexity to the encryption phase, making decryption lightweight (with less depth). This approach is known as Squashing.

2.1.5 Developing FHE From SWHE

Gentry's method may be applied to any somewhat homomorphic encryption system in three phases to produce a fully homomorphic encryption scheme as follows:

I) Developing SWHE

II) SWHE to LHE conversion

III) Making LHE self-sustaining or bootstrappable.

• Squashing

2.2 The DM Scheme

DM is a symmetric encryption method based on an FHE algorithm. In this method, ciphertexts are represented as vectors, and homomorphic operations correspond to ciphertext vector additions. Because of its straightforward homomorphic operation, it is theoretically simpler than other LWE-based FHEs. The following is DM cryptosystem:

2.2.1 Preliminaries

• The LWE Problem

Regev proposed the LWE[1] issue, which is a well-known computation problem over lattices. Let $n = n(\lambda)$ and $q = q(\lambda)$ indicate the size and modulus of the vector, respectively, for a security parameter λ, and let $\chi = \chi(\lambda)$ denote the random distribution on Z for the random errors. By sampling $s \leftarrow Z$, the vector s is created. Output the following LWE instance $(a,b) = (a,(a.s+e)\,mod\ q) \in Z_q^{n+1}$ for vector $a \leftarrow Z_q^n$ and error $e \leftarrow \chi$. The LWE assumption is that the uniform distribution on Z_q^{n+1} created by distinct LWE instances is computationally indistinguishable from the distribution $\chi^{'}$ formed by different LWE instances.

2.2.2 DM Cryptosystem [1]

• $keygen(\lambda)$:λ specifies a security parameter t exceeds two the inline modulus is an integer. The ciphertext dimension $n = n(\lambda)$, modulus $q = q(\lambda)$, and ciphertext noise distribution $\chi = \chi(\lambda)$ have been configured to ensure a security level of λ. In this case,$x < q/2t$ for any $x \leftarrow \chi$. The parameter set $params = (n,q,t,\chi)$ will be denoted. The key is sampled evenly from: $Z_q^n : pk/sk \leftarrow Z_q^n$.

• $Enc(m,pk,params)$:the plaintext and ciphertext spaces are Z_t, Z_q respectively.sample a,Z_q^n, $e \leftarrow \chi$, on input plaintext message $m \in Z_t$ and output ciphertext:

$$LWE_s^{t/q}(m) = (a, a.s + \frac{mq}{t} + e) \in Z_q^{n+1} \tag{2.6}$$

• $HomeNAND((a_1,b_1),(a_2,b_2))$:on input ciphertexts $c_i = (a_i,b_i), i \in 1,2$ and $c_i \in LWE_s^{4/q}(m_i, q/16)$ encrypts the plaintext message m_i, $output c = (a,b) \in LWE_s^{2/q}(1-m_1m_2, q/4)$.In particular,

$$(a,b) = (-a_1 - a_2, \frac{5}{8}q - b_1 - b_2) \tag{2.7}$$

The ciphertext (a,b) is a $1-m_1m_2$ ciphertext with noise magnitude smaller than $q/4$, ensuring accurate decoding. In DM, homomorphic NAND operations are performed by a few additions between ciphertext vectors, which are simpler and quicker than prior schemes' tensor products

or matrix operations. However, if the ciphertext magnitude is more than $q/4$ after another homomorphic operation, the ciphertext will no longer be successfully decrypted. To keep the noise magnitude low, ciphertext must be updated after each homomorphic operation.
To reduce ciphertext noise, an efficient ciphertext refreshing method based on Ring-GSW is presented in DM. The refreshing method takes as input ciphertext $(a,b) \in LWE_s^{2/q}(m,q/4)$ and the refreshing key K_{rf}, and base B_r is used to encode the ciphertext (a,b). K_{rf} consists of the following ciphertexts:

$$K_{i,c,j} = E\left(cs_i B_r^j \ mod\ q\right), c \in \{0,...,B_r-1\}, j = 0,...,d_r-1, i = 1,....,n \tag{2.8}$$

where $d_r = \lceil \log_{B_r} q \rceil$ and $E(.)$ is the encryption algorithm used in the ciphertext refreshing algorithm Method 1 depicts the ciphertext refreshing algorithm, where $Init(.)$ and $Incr(.)$ indicate the initialization and homomorphic addition of the accumulator *ACC*, respectively. *ACC* is initially configured as a $b+q/4$ encryption. When the main loop in 2.1 completes, the accumulator's underlying plaintext v satisfies

$$v - \frac{q}{4} = b + \sum_{i,j} a_{i,j} s_i B_r^j = b + \sum_i s_i \sum_j B_r^j a_{i,j} = b - \sum_i a_i s_i = \frac{q}{2} m + e \tag{2.9}$$

Where e is the amount of noise in the input ciphertext (a,b). As $mod\ e < q/4$. It is obvious

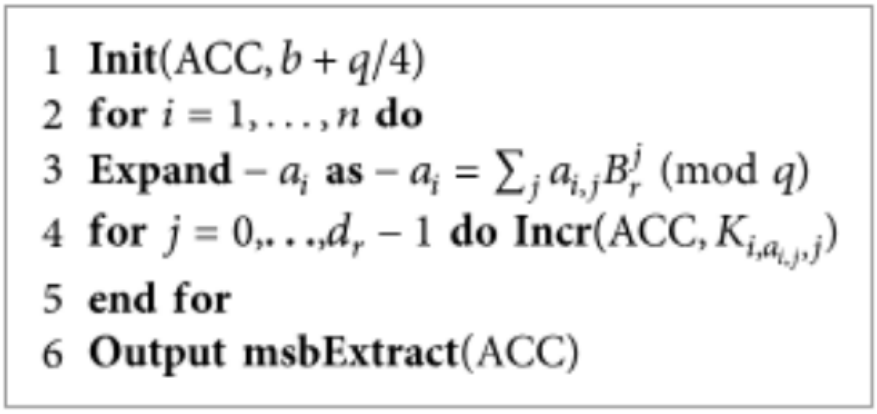

Figure 2.1: DM msbextract[1]

that $0 < v < q/2$ when $m = 1$. In other words, obtaining the plaintext m by extracting the most significant bit (msb) from v. The accumulator *ACC*, coupled with a switching key K_{ks} and a testing vector $t = -\sum_{i=0}^{\frac{q}{2}-1} CF(Y^i)$ is used as input during the *msbExtract* procedure in figure 2.1 . Here, $Y = X^{\frac{2N}{q}}$, where z is the secret key used in the ciphertext refreshing algorithm's encryption method. 2.2 shows the specifics of *msbExtract*.
The ciphertext c in the 2nd step of 2.2 is

$$c = (a, b_0 + u) = (a.CF(z) + t.e + 2u.msb(v)) \tag{2.10}$$

$$
\begin{aligned}
&1\ \ [\mathbf{a}^t, \mathbf{b}^t] = [\mathbf{0}^t, \mathbf{t}^t, \mathbf{0}^t, \ldots, \mathbf{0}^t] \cdot \mathbf{ACR}(\mathrm{ACC}) \\
&2\ \mathbf{c} = (\mathbf{a}, b_0 + u) \in \mathrm{LWE}^{t/Q}_{\mathbf{CF}(z)}(\mathrm{msb}(v)) \\
&3\ \mathbf{c}' = \mathbf{KeySwitch}(\mathbf{c}, K_{ks}) \in \mathrm{LWE}^{t/Q}_{\mathbf{s}}(\mathrm{msb}(v)) \\
&4\ \mathbf{c}'' = \mathbf{ModSwitch}(\mathbf{c}') \in \mathrm{LWE}^{t/q}_{\mathbf{s}}(\mathrm{msb}(v))
\end{aligned}
$$

Figure 2.2: DM Ciphertext Refresh [1]

Where $a = t^t.ACR(a), [a, b^{'}]$ is the 2nd row of ACC and $u = \lceil Q/2t \rceil$ *or* $\lfloor Q/2t \rfloor$. As $u/2t, c$ is an encryption of $msb(v) = m$. Thus $c^{t/Q}_{CF(z)}(msb(v))$.Following key and modulus switching, c is converted to ciphertext under key s modulo q. Under the right parameter settings, the noise amplitude of the refreshed ciphertext would be less than $q/16$, allowing for additional homomorphic computations.

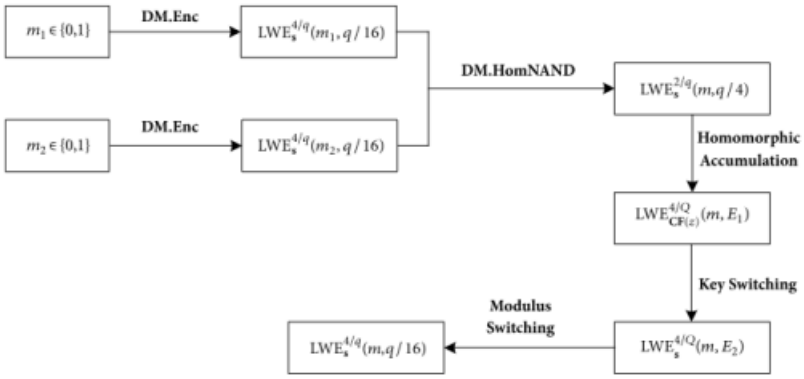

Figure 2.3: Flow of DM Homomorphic Scheme

2.3 The GSW Scheme[1]

The GSW method is based on matrices' estimated eigenvectors. All ciphertext is represented as a matrix, and all homomorphic operations are just ciphertext matrix operations. In comparison to earlier LWE-based FHE systems that required either keyswitching or relinearization, the GSW method is more natural and succinct. In the next part, we will first go over the mathematical requirements for the GSW scheme, followed by a walkthrough of the GSW cryptosystem.

2.3.1 Preliminaries

- **Cyclotonic Ring**

If N is a power of 2, the $2N-th$ cyclotonic polynomial is$\emptyset_{2N}(X) = X^N + 1$, and the polynomial ring is $R = Z[X]/X^N + 1$. The residue ring of R modulo an integer Q is denoted by $R_q = R/QR$. Each element in R is a polynomial with integer coefficients whose degree is at most$N-1$, and each element in R_Q is an element R with all of the polynomial's coefficient vector. Let $ACR(r)$ denote the matrix shown below: As illustrated in, the first column is $CF(r)$, while the other columns are anti-cyclic rotations of $CF(r)$ with the cycled entries ignored.

$$ACR(r) = \begin{vmatrix} r_0 & -r_{N-1} & \cdots & -r_1 \\ r_1 & r_0 & \cdots & -r_2 \\ \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & -r_{N-2} & \cdots & r_0 \end{vmatrix} \tag{2.11}$$

•BitDecomp and Flatten Techniques

Let $BD(.)$ represent the BitDecomp operation, and $a,b \in Z_q^k, l = |\log q| + 1, N = kl$. The operation Bitdecomp is defined as follows:

$$BD(a) = (a_{1,0},...,a_{1,l-1},...,a_{k,0},...,a_{k,l-1}) \tag{2.12}$$

where $a_{i,j}$ denotes the j-th bit in a_i's binary representation, from lowest to highest. The upper bound of a's $l1$ norm is reduced from nq to $n\log q$ after BitDecomp. Let $BD^{-1}(.)$ indicate the inverse operation of $BD(.)$;given a vector $a\prime = (a_{1,0},...,a_{1,l-1},...,a_{k,0},...,a_{k,l-1}) \in Z_q^N$, $BD^{-1}(.)$ is defined as follows:

$$BD^{-1} = (\sum_{j=0}^{l-1} 2^j a_{1,j},...,\sum_{j=0}^{l-1} 2^j a_{k,j}) \in Z_q^k \tag{2.13}$$

Let $FL(.)$ indicate the flatten operation; $FL(.)$ is defined for a vector $a^{'}Z_q^k$ as follows.

$$PT(b) = (b_1, 2b_1, ..., b_k, 2b_k, ..., 2^{l-1}b_k) \in Z_q^N \tag{2.14}$$

The following is an evident trait shared by $BD(.)$ and $PT(.)$:

$$< a^{'}, PT(b) > = < BD^{-1}(a^{'}, b > = < FL(a^{'}, PT(b) > \tag{2.15}$$

The above equation shows that an essential feature of $FL(.)$ is that it reduces the coefficients of a vector without altering its inner product with the vector $PT(b)$. When the aforementioned operations are performed on a matrix, they are done for each row of the matrix.

2.3.2 GSW Cryptosystem

• $KeyGen(\lambda, L)$: The security parameter and multiplicative depth are denoted by $\lambda.L$. The ciphertext dimension $n = n(\lambda, L)$, modulus $q = q(\lambda, L)$, and noise distribution $\chi = \chi^(\lambda, L)$ are all adjusted to ensure a security level of λ. Let $m = O(n\log q, l = \lfloor \log q \rfloor + 1, N = (n+1)l$, and

the parameter set $params = (n, q, \chi, m)$. $t \leftarrow Z_q^n$ is sample, $s = (1, -t) \in Z_q^(n+1)$, and the secret key $sk = v = PT(s)$ is produced. $B \leftarrow Z_q mn, e \leftarrow \chi^m$, is sample,let $b = B.t + e, A = [b||B]$, and output public key $pk = A$.

• $Enc(params, pk, m)$:For plaintext message $m \in Z_q$,sample $R \leftarrow \{0,1\}^{Nm}$; output ciphertext:

$$C = FL(m.I_n + BD(R.A)) \in Z_q^{NN} \tag{2.16}$$

• $HomeNAND(C_1, C_2)$:For input ciphertext pair $C_1, C_2 \in Z_q^{NxN}$,output will be

$$C_{NAND} = FL(I_N - C_1 C_2) \tag{2.17}$$

After homomorphic NAND operation,C_NAND satisfies the below property:

$$C_{NAND} v = (1 - m_1.m_2)v - m_2 e_1 - C_1 e_2 \tag{2.18}$$

In above equation m_1, m_2 are the plaintext messages for C_1, C_2 and e_1, e_2 are noises.Now we assume B_0 is the upper bound of the noise amplitude in C_1, C_2,which is upper bound for l_∞ norms of e_1, e_2.Obviously $max\{||e_1||_\infty, ||e_2||_\infty\} < B_0$.for the result of faltten operation $C_1, C_2 \in \{0,1\}^{NxN}$.As demonstrated in the preceding equation, the noise in C_NAND is upper limited by $(N+1)B_0$.

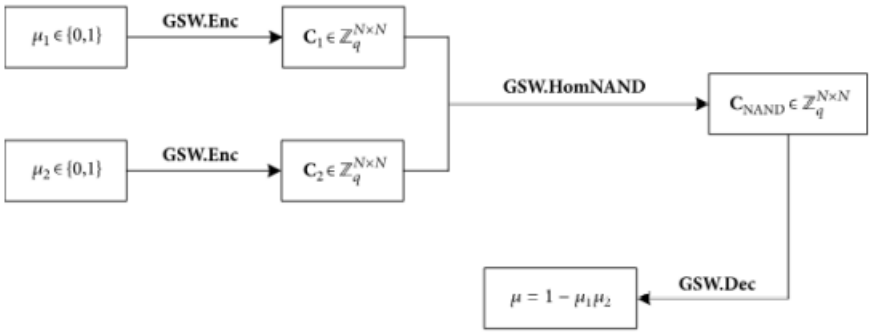

Figure 2.4: Flow of GSW Homomorphic Scheme

2.4 Properties of Homomorphic Encryption Schemes [8]

2.4.1 Strong Homomorphism

Strong homomorphism necessitates that evaluated ciphertexts resemble fresh ciphertexts in appearance.

2.4.2 Compactness

The fundamental flaw in the dull realization in which Evaluate just appends the description of the ciphertexts is that we want the decryption process to be the same whether the decrypted ciphertext is fresh or evaluated. This is obviously true if the scheme is substantially homomorphic, although being extremely homomorphic is frequently overkill. Compactness is a lesser concept that captures a lot of the power of homomorphic computing. It simply demands that the size of the ciphertext does not expand with the complexity of the evaluated circuit.

2.4.3 Circuit Privacy

Circuit privacy implies that the ciphertext created by Evaluate does not expose anything about the circuit that it evaluates other than its output value, even to the entity that supplied the public and secret keys. To put it another way, we see Evaluate's operation as a protocol between a client that generates keys and encrypts its input and a server that evaluates a function on that input and provides the result to the client. We next characterize circuit privacy as the server's normal input privacy feature, requiring that the client be simulated using just the output value that it learns.

3 Conclusion and Future Work

3.1 Conclusion

As the usage of outsourced compute grows, so does the need for safe computation on third-party machines, which is the primary motivating factor in the study of FHE methods. Since the very first FHE plan presented by the gentry, research in FHE has advanced at a rapid speed, with numerous schemes proposed; however, most of the FHE schemes available right now have performance difficulties that make them unsuitable with actual applications. Many industry heavyweights, like Amazon and Microsoft, are also working on the topic, and some of them have even begun releasing open source projects on it, such as Microsoft's SEAL. As the world becomes more concerned about the security and privacy of data, security schemes like FHE gain traction. Although currently existing FHE methods have proven good outcomes in terms of security and accuracy, the majority of them are computationally highly costly and difficult to implement in real-world situations.

3.2 Future Work

Here we will discuss the things that need to be done to improve the FHE.
• The majority of current effective FHE methods are based on RLWE, which is a structured version of LWE over algebraic rings. The algebraic structure of RLWE is yet unknown in terms of how resilient it is. It has recently been demonstrated that this structure leads to more effective attacks on SVP-type issues. It is necessary to investigate how secure these techniques are.
•The majority of FHE techniques are based on hard lattice problems that deal with huge integer vectors or polynomials. The study of these mathematical objects is time-consuming, which impedes the study and application of FHE. FHE methods based on novel computing problems, such as LWE, would be fantastic.
• Most FHE schemes are specified for numerical data; however, in order to fully apply FHE on outsourced computing, FHE methods that can handle other data types, such as strings and floating points, must be developed.
• There is a lack of comparative study that indicates the calculation capabilities of currently existing schemes in terms of ciphertext size handled, amount of time spent on computation, and simplicity of implementation. Working on benchmarking and comparison studies can make it easier for those who wish to deploy FHE-based solutions that meet their needs without delving too far into each scheme.
•Assume sender send his encrypted data to the server, which will evaluate a function f. After receiving the encrypted results, the data owner may inquire whether these ciphertexts are the true output of the function f. In reality, without the use of new cryptographic primitives, such verification is unachievable in present FHE methods.

4 References

[1] T. L. Xun Wang and J. Li, “A more efficient fully homomorphic encryption scheme based on gsw and dm schemes,” Security and Communication Networks, vol. 2018, 2018.

[2] C. C. K. G. A. J. C. A. R. Frederik Armknecht, Colin Boyd and Martin, A Guide to Fully Homomorphic Encryption.

[3] C. Gentry, A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[4] I. Iliashenko, Optimisations of fully homomorphic encryption. PhD thesis, Podunk IN, 2019.

[5] Z. H. Mahmoo and M. K. Ibrahem, “New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing,” AICIS, vol. 2018, 2018.

[6] J.-L. H. Jian Liu and Z.-L. Wang, “Searchable encryption scheme on the cloud via fully homomorphic encryption,” International Conference on Instrumentation Measurement, Computer, Communication and Control, vol. 6, 2016.

[7] Jaydip Sen "Homomorphic Encryption — Theory and Application" Submitted: May 3rd 2012Reviewed: May 23rd 2013Published: July 17th 2013 DOI: 10.5772/56687

[8] Shai Halevi (IBM Research) "Homomorphic Encryption" April 2017.

[9] Pratibha Chaudhary and Ritu Gupta and Abhilasha Singh and Pramathesh Majumder "Analysis and Comparison of Various Fully Homomorphic Encryption Techniques" International Conference on Computing, Power and Communication Technologies (GUCON),2019

[10] Wikipedia, Homomorphic Encryption "https://en.wikipedia.org/wiki/Homomorphic encryption"

Acknowledgement

I would like to thank the institute to give me this opportunity to write seminar report on Fully Homomorphic Encryption. I would like to thank Sachi Shah for valuable guidence.